

A white coiled telephone cord is shown against a yellow background. The cord is coiled in a series of loops, starting from the bottom left and moving towards the top right. The background is a solid, light yellow color. The top and bottom of the cover are black.

# Information Security

Policies and Actions in Modern Integrated  
Systems

Mariagrazia Fugini  
and Carlo Bellettini

# **Information Security Policies and Actions in Modern Integrated Systems**

Maria Grazia Fugini, Politecnico di Milano, Italy

Carlo Bellettini, Università degli Studi di Milano, Italy



**IDEA GROUP PUBLISHING**

Hershey • London • Melbourne • Singapore

Acquisitions Editor: Mehdi Khosrow-Pour  
Senior Managing Editor: Jan Travers  
Managing Editor: Amanda Appicello  
Development Editor: Michele Rossi  
Copy Editor: Alana Bubnis  
Typesetter: Sara Reed  
Cover Design: Lisa Tosheff  
Printed at: Yurchak Printing Inc.

Published in the United States of America by  
Idea Group Publishing (an imprint of Idea Group Inc.)  
701 E. Chocolate Avenue, Suite 200  
Hershey PA 17033  
Tel: 717-533-8845  
Fax: 717-533-8661  
E-mail: [cust@idea-group.com](mailto:cust@idea-group.com)  
Web site: <http://www.idea-group.com>

and in the United Kingdom by  
Idea Group Publishing (an imprint of Idea Group Inc.)  
3 Henrietta Street  
Covent Garden  
London WC2E 8LU  
Tel: 44 20 7240 0856  
Fax: 44 20 7379 3313  
Web site: <http://www.eurospan.co.uk>

Copyright © 2004 by Idea Group Inc. All rights reserved. No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without written permission from the publisher.

Library of Congress Cataloging-in-Publication Data

Information security policies and actions in modern integrated systems  
/ Mariagrazia Fugini, editor, Carlo Bellettini, editor.

p. cm.

ISBN 1-59140-186-0 (hardcover) -- ISBN 1-59140-291-3 (pbk.) -- ISBN  
1-59140-187-9 (ebook)

1. Computer security. 2. Integrated circuits. I. Fugini,  
Mariagrazia, 1957- II. Bellettini, Carlo, 1967-

QA76.9.A25I541545 2004

005.8--dc22

2003022606

British Cataloguing in Publication Data

A Cataloguing in Publication record for this book is available from the British Library.

All work contributed to this book is new, previously-unpublished material. The views expressed in this book are those of the authors, but not necessarily of the publisher.

# **Information Security Policies and Actions in Modern Integrated Systems**

## **Table of Contents**

<b>Preface .....</b>	<b>vi</b>
----------------------	-----------

### **Section I: Cryptography**

#### **Chapter I**

<b>Architectures for Advanced Cryptographic Systems .....</b>	<b>1</b>
---	----------

*Guido Bertoni, Politecnico di Milano, Italy*

*Jorge Guajardo, Infineon Technologies AG, Germany*

*Christof Paar, Ruhr Universität Bochum, Germany*

#### **Chapter II**

<b>Digital Certificates and Public-Key Infrastructures .....</b>	<b>64</b>
--	-----------

*Diana Berbecaru, Politecnico di Torino, Italy*

*Corrado Derenale, Politecnico di Torino, Italy*

*Antonio Lioy, Politecnico di Torino, Italy*

#### **Chapter III**

<b>Smart Card Applications and Systems: Market Trend and Impact on Other Technological Developments .....</b>	<b>98</b>
---	-----------

*Gerald Maradan, STMicroelectronics, France*

*Pierre Cotte, STMicroelectronics, France*

*Thierry Fornas, STMicroelectronics, France*

## Section II: Authorization Frameworks

### Chapter IV

#### **A Flexible Authorization Framework ..... 149**

*Duminda Wijesekera, George Mason University, USA*

*Sushil Jajodia, George Mason University, USA*

### Chapter V

#### **Enforcing Privacy on the Semantic Web ..... 177**

*Abdelmounaam Rezgui, Virginia Tech, USA*

*Athman Bouguettaya, Virginia Tech, USA*

*Zaki Malik, Virginia Tech, USA*

## Section III: Data Distribution and Dissemination on the Net

### Chapter VI

#### **Secure Data Dissemination ..... 198**

*Elisa Bertino, Università degli Studi di Milano, Italy*

*Barbara Carminati, Università degli Studi di Milano, Italy*

*Elena Ferrari, Università degli Studi dell'Insubria, Italy*

### Chapter VII

#### **Multimedia Security and Digital Rights Management Technology.. 230**

*Eduardo Fernandez-Medina, Universidad de Castilla-La Mancha,  
Spain*

*Sabrina De Capitani di Vimercati, Università di Milano, Italy*

*Ernesto Damiani, Università di Milano, Italy*

*Mario Piattini, Universidad de Castilla-La Mancha, Spain*

*Pierangela Samarati, Università di Milano, Italy*

## Section IV: Service Oriented Computing Frameworks

### Chapter VIII

#### **Data and Application Security for Distributed Application Hosting Services ..... 273**

*Ping Lin, Arizona State University, USA*

*K. Selçuk Candan, Arizona State University, USA*

**Chapter IX**

**Comparing the Security Architectures of Sun ONE and Microsoft**

**.NET ..... 317**

*Eduardo B. Fernandez, Florida Atlantic University, USA*

*Michael Thomsen, Florida Atlantic University, USA*

*Minjie H. Fernandez, Florida Atlantic University, USA*

**About the Authors ..... 331**

**Index ..... 339**

# Preface

We are witnessing a rapid growth of technological and business interest towards distributed computing environments, also named Global or Integrated Computing environments. The features of these environments include mobility, open-endedness, heterogeneity of data and applications, mobility of computing entities on a variety of devices, network systems, and logical and physical interconnections channels. In these systems, various resources (applications, data and repositories, as well as user contexts scenarios) demand to be merged under common interoperability paradigms, possibly based on standards and common policies, as well as on enabling technologies for interconnection, such as web Services, internetworking connection technologies, or distributed code and data management. Heterogeneity is a key factor in interconnection: the ability to manage it means the possibility to manage distributed and differently structured and managed resources under unifying paradigms for computing resources and networks.

In this scenario of integrated environments, the ability to create and manage large shared systems in a secure manner is an area that has received attention in various ways, from formal research to practical approaches, methods, and products. A comprehensive, systems approach to security is required if security consolidation is to succeed. This book serves as a forum to describe security threats, technologies, methodologies and deployment in the area of systems integration. The book collects submissions from academia and industry presenting research and development on theoretical and practical aspects related to designing, building and managing secure distributed systems.

The included topics range from Cryptographic Algorithms to Key Management and Public Keys Infrastructures for Security Management, from Authorization Frameworks for Security and Trust management to Models of Security in Federated Systems and Security for Internet Service Oriented Architectures and web-managed data formats.

## BOOK AIMS

The basic aim of the volume is to state the point of recent achievements in the field of security related to the interconnection of computers and applications through internetworking. In fact, the Internet is a worldwide collection of networks that are accessible by individual computing hosts in a variety of ways, including gateways, routers, dial-up connections, wireless networks and Internet service providers. In principle, the Internet is easily accessible to any person endowed with a computer and a network connection; individuals and organizations worldwide can reach any point on the network without regard to national or geographic boundaries or time of day. However, along with the convenience and easy access to information, new risks for security of information and personal data arise. First, the general risk is that valuable information will be lost, stolen, corrupted, or misused and that the computer systems, that is, its valuable data and applications, will be corrupted or damaged. It is well known and accepted that information electronically managed and available on networked computers is vulnerable: intruders do not need to enter an office or a building, and may be located anywhere. They can steal or tamper information without touching a paper or a computing device, creating new or altered files, run their own programs, and, particularly, they can easily hide evidence of their unauthorized activity, practically leaving no trace of their actions.

### Concepts

Just to briefly review the basic security concepts, important to *information on internetworking applications*, we remind confidentiality, integrity, and availability. Concepts relating to *information users* are instead authentication, authorization, and non-repudiation.

When information is read or copied by someone not authorized to do so, the result is known as loss of confidentiality. For some types of information or organizations and companies, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications, and corporate investment strategies. In some locations, there may be a legal obligation to protect the privacy of individuals. This is particularly true for banks and loan companies; debt collectors; businesses that extend credit to their customers or issue credit cards; hospitals, medical doctors' offices, and medical testing laboratories; individuals or agencies that offer services such as psychological counseling or drug treatment; and agencies that collect taxes.

Besides, information can be corrupted when it is available on an insecure network. When information is modified in unexpected ways, the result is known as loss of integrity. This means that unauthorized changes are made to information, whether by human error or intentional tampering. Integrity is particularly important for critical safety and financial data used for activities such as electronic funds transfers, air traffic control, and financial accounting.

Information can be erased or become inaccessible, resulting in loss of availability. This means that people who are authorized to get information cannot get what they need.

Availability is often the most important attribute in service-oriented businesses that depend on information (e.g., airline schedules and online inventory systems). Availability of the network itself is important to anyone whose business or education relies on a network connection. When a user cannot get access to the network or specific services provided on the network, they experience a denial of service.

To make information available to those who need it and who can be trusted with it, organizations use authentication and authorization. Authentication, which is the way to prove that a user is whom he/she claims to be, needs involved parties to provide a “proof.” This may involve something the user knows (such as a password), something the user has (such as a “smart card”), or something about the user that proves the person’s identity (such as a fingerprint). Authorization is the act of determining whether a particular user (or computer system) has the right to carry out a certain activity, such as reading a remote file or running a program. It is nowadays well accepted that authentication and authorization go hand in hand. Moreover, security is strong when the means of authentication cannot later be refuted — the user cannot later deny that he or she performed the activity. This is known as non-repudiation.

## QUESTIONS

The book addresses several questions, such as: What kind of crypto algorithms are being used and are emerging? What kind of certificate and public key infrastructure is the appropriate one for the global Internet? What is the state-of-the-art of certificates in solving authentication problems, and how can authorization problems be solved in a distributed environment? What security problems are emerging in distributed federated databases and document management systems or in distributed code? For example, the information industry is heading rapidly towards adopting Java as the programming language for the Internet. Will the Java Sandbox approach be sufficient to fulfill users’ and industrial needs for building the global information infrastructure? For smart cards, industry studies indicate that there could be some 2.5 to 3 billion smart cards in use by the turn of the century. Major users of these joint smart card technologies are forecast to include telecommunications, banks, hotels, airlines and insurance companies, as well as healthcare providers and governments. Smart cards, however, are limited in their processing power and storage capacity, which most security algorithms require. How will the industry solve these shortages and what impact smart cards have on enhancing individuals’ security and privacy?

Finally, recent proposals suggest a peer-to-peer relation between the client and the server, where the server will be able to push some content to client(s). Will push technology invade individuals' privacy and create a flooded Internet; or should it be regulated to become subscription based?

Will governments open the national boundaries and stop regulating the encryption technology as well as giving common rules and standards for designing lightweight security algorithms and trusted systems? Will it be possible to build Intranet/Extranets and Virtual Private Networks with full security? Finally, will the Internet and e-commerce rise to the expectation and flourish in the global village? The answers to these questions are yet to be seen.

## SECURITY AS A BUSINESS PROBLEM

Information security is today also a business problem, besides a technological problem. With the focus on information security in the media, and in legislatures around the world, organizations are facing complex requirements to comply with security and privacy standards and regulations. This is forcing the discussion of information security into boardrooms, as more executives and boards of directors understand their responsibility and accountability in information security governance. Current topics of discussion in the security field are driven mainly by the following issues:

- *Focus on information security:* The awareness of the challenges and issues to be faced in information security has grown. Through the media, government, cyber-attacks/crimes, and proliferation of vulnerabilities in products, information security continues to receive increased focus.
- *Technology to protect information:* As a result of successful attacks (such as Code Red and Nimda), the organizations have acknowledged that security products are not a complete solution to security problems, but rather security is a business and organizational problem.
- *Standards, regulations and legislation:* Companies and organizations have to face complex standards and regulations. Even within very specific, vertical areas, such as banking services, the complexity to meet security requirements is driven by the presence of different regulations (e.g., the U.S. Gramm-Leach-Bliley Act of 1999 (GLBA), Basel Accords, Securities and Exchange Commission (SEC) requirements, U.S. Patriot Act). A complex set of requirements is emerging while organizations cross the international boundaries.
- *Legal liability:* In 2002, legal liability from security has been stated. Organizations and software vendors are being pushed towards a higher degree of accountability for security by their customers.

- *Business partners demanding security:* Organizations have to prove that they are managing data and applications in a secure way, that is, they are applying security to a level that can satisfy their business partners. This goes beyond discussing what security products are installed; this requires that organizations be able to communicate compliance and management practice of security. For example, in the U.S., the National Strategy to Secure Cyberspace released by the White House recommends that organizations disclose their security audit and compliance status.

As a consequence, organizations can expect to see increased regulation, specifically in industry areas that are considered as critical, such as finance, transportation, communication, health care, energy, and utilities. Furthermore, regulatory requirements will come up from governments seeking to boost information security. Luckily, *information security* has turned into a well founded and defined *profession*, and many common security services will continue to be valuable and a real necessity — vulnerability management, secure communications, penetration testing, policy design, intrusion detection — while others may be changing or evolving radically.

Currently, information security is turning from awareness into action in many corporate environments, building on what we have seen in the last decade. Boards of directors and executive management are paying closer attention to their responsibilities in the protection of their information assets. Moreover, we are assisting at an increased focus on certification and accreditation with continuous assessment: as organizations face compliance obligations, or standards and best practices to manage information protection plans, we observe a focus on the certification and accreditation of system security before production implementation. This will be followed by the development of a continuous assessment process to manage risk and compliance with standards and regulations. For example, ISO17799 and BS7799 have become the de facto standards for defining (at a high level) an information security program/architecture. Also, the Common Criteria product certification is more and more widely pursued and recognized: with the mandate that security products be Common Criteria certified in order to be purchased, at least by U.S. Department of Defense Agencies, a significant increase is obtained in the adoption of Common Criteria certification.

## INTEGRATION TECHNOLOGIES

In the *integration of different systems* into a distributed global system, issues of data dissemination, access to a variety of applications, disseminated users of various typologies and with heterogeneous needs bring about new security needs. To mention just a few aspects, an integrated system must be able to:

- Authenticate users and applications in a distributed way;
- Selectively grant and revoke access rights to users of the distributed system;
- Encrypt data in an effective and yet efficient way both when data are transmitted and when data are stored;
- Treat in a confidential and privacy respectful way the large amount of structured and unstructured data present in various formats: database data, documents, web pages, links, and so on;
- Manage in a uniform way heterogeneous policies regarding data protection;
- Preserve data from sophisticated attacks that exploit the presence of inter-networked databases and systems, such as Trojan Horses attacks, backdoors, distributed denial of service, or statistical inference.

Current trends in *security technology* go towards the improvement of existing security systems, based on new requirements and customer dissatisfaction. For example, signatures and vulnerability management, or Intrusion Detection Systems (IDS) are being redesigned and improved to decrease the number of “false positives” and improving reliability, while preparing for integration of IDS systems with firewalls, crypto systems, authentication and authorization schemes and models, and security management tools. Also, security models in operating systems, in databases and in web-based systems are being improved by adding standardized ACLs, capability modes, and other security features.

Distributed and peer-to-peer systems, e.g., in service oriented architectures, are starting to gain commercial acceptance; in the meanwhile, architectures for distributed services are emerging with reliable hosting, communication security, backups, secure computation, secure content management, and so on. With trends towards distribution, secure communications are increasingly relevant, since hijacking scenarios, or mis-authenticated agents and servers become possible. A key to securing distributed and cryptography-employing systems will be the strong authentication of any message or data, by employing reliable Public Key exchanges and trusted cryptographic identification. Strong authentication may act as foundation for flexible and availability — assuring accounting and billing systems, such as service management, integrated with Quality of Service Standards on the IP level, in combination with wireless authentication.

Luckily, as long as the technology is evolving, the *legislation on the themes of security* of electronically managed information is becoming stricter and demanding at the national and international levels. Rules and laws are requiring producers and manufacturers, application developers, and service providers to adequate their systems to existing and emerging security levels and standards. As a consequence, companies and Public Administrations are more and more constrained both for business problems and for regulation and legislative prob-

lems to protect their data and application patrimony. Not the last are the problems of industrial espionage, of citizens' personal data discovery and of business and image losses due to attacks and to denial of service or discontinuity of services.

This book wants to put a head forward in the direction of integration of technologies tackling the use of new paradigms, such as web access, (wireless) internetworking, web services, and service oriented computing. These have lead to the urgent need for companies, agencies, and administrations to endow their systems with security measures for physical, logical and organizational security.

## BOOK ORGANIZATION

Many research and development efforts worldwide are currently focused on providing stronger and more efficient cryptographic algorithms and infrastructures. **Section I** of this volume is devoted to presenting an overview of recent progresses of cryptography and of its usability, as well as advances in one technological area, the one of smart cards.

Mathematical progresses are presented and crypto applications are illustrated, such as in digital signatures, digital certificates, secure data formats, secure communication protocols, time stamping, insurance of Trust, PKI Servers and Wireless LAN certificates. Aspects of authentication mechanisms based on traditional passwords schemes and on truly complex systems, such as smart card based systems, biometrics authentication, and dedicated software systems are then presented. For networks, the basic of techniques remains cryptography, which is able to ensure a large set of security properties; the problem with cryptography lies in the need to produce more and more sophisticated encryption algorithms and in the need to have manageable systems that reveal to be of treatable complexity in front of encryption/decryption cost and time. Hence the problems are to set up efficient cryptographic systems able to generate and handle crypto keys and certificates and to reach a corporate-level structure of cryptography such as PKI systems, requiring an acceptable effort by managers to operate all the aspects related to the presence of a cryptographic system.

Then, the market orientation of a vendor of security products is inserted, providing an insight on security challenges, on a particular, vertical application, i.e., smart cards, but also providing an interesting overview of the security market perspectives.

Another set of efforts in research and development in the security area are devoted to authorization frameworks, which are aimed at ensuring selective access to information. **Section II** of this volume contains articles devoted to models and systems that allow a precise definition of *access rights*, based on a wide range of paradigms, such as DAC and MAC controls, role based access controls, or credential and reputation based controls. These frameworks

are presented with focus on Internet data and applications, such as Internet-based transactions, workflow systems, federated databases, and information exchange in the semantic web, where machines are regarded as “intelligent agents” able to process and make inferences on a large variety of data.

On federated and distributed system, data distribution and dissemination in the networks resources brings about problems of individual privacy, of selective and dynamic authorization to access portions of the data and to deal with data in various formats, such as web-compliant, XML-based data and images, semantic web oriented information, and voice. **Section III** of this volume contains two articles dealing with confidentially and privacy respectful way the large amount of structured and unstructured data present in various formats: database data, documents, web pages, links, video and voice streams, images, and so on.

Subsequently, the book moves to new applications deployed on internetworked environments, tackling security problems in data and application distribution and web services frameworks based on peer-to-peer environments. Some interesting comparisons among distributed development environments are provided, discussing enabling security technologies. **Section IV** of the book examines distributed application hosting services (DAHs) and next offers an evaluation of claimed security features in some popular products oriented to web service management.

In more detail, the book sections have the following contents.

## **Section I: Cryptography and Technology**

This first section of the book sets the basis for cryptography with some details on algorithms and on crypto-analysis techniques and tools.

Chapter I by Bertoni et al., “Architectures for Advanced Cryptographic Systems,” is intended to give an overview of recent developments in modern cryptography. In the last few years, modern cryptography has been dominated by traditional systems, such as DES and RSA. Such systems have provided a secure way for storing and transmitting information, and are nowadays incorporated in many network protocols and secure storage media. However, more recently, the increasing power of crypto-analysis techniques and tools, and the emergence of new applications, such as wireless communications and mobile computing, service-oriented architectures, and integrated systems have stimulated the research and development of innovative cryptographic algorithms. New integrated systems require a more detailed and sophisticated mathematical formalization of cryptographic techniques. This chapter aims at giving the reader a comprehensive understanding of innovative crypto-systems, of their basic structure, of the alternative hardware architectures to implement them, of the application fields, and of their performance requirements and characterizations. Focus is put, among the others, on Advanced Encryption Standard and Elliptic Curve Cryptosystem.

Chapter II by Berbecaru et al., “Digital Certificates and Public Key Infrastructures,” is an exhaustive overview of PKI basics, architectures, and applications. Digital Certificates are signed objects containing a set of data bound together by a digital signature. Currently, Digital Certificates can be divided into three classes, based on the data they are bound to: identity certificates (often referred as public-key certificates, PKC) attribute certificates, and authorization certificates. The chapter explores the various possibilities of certificate structures, standards and usages.

Chapter III by Maradan et al., “Smart Card Applications and Systems: Market Trend and Impact on Other Technological Developments,” tackles a theme that is strategic in all fields of security and for all types of organizations. The chapter evidences the new needs and requirements for this important authentication support. The diffusion of new communication technologies has pushed smart cards to a very urgent need of applications, although GSM has been the market driver of this authentication means. In this chapter, our will is to provide ideas and trails to explain what make the strengths of smart cards, exploring both technical security issues and market trends. At the same time, the chapter explores the state-of-the-art of hacking techniques, and reveals some counter measures that could redesign modern security platforms. The chapter illustrated the evolution of smart card platforms and their impact on integrated systems, focusing on content protection, e-commerce, and pay TV systems. It finally presents a case study: the e-content protection and Smartright proposal.

## **Section II: Authorization Frameworks**

Chapter IV by Wijesekera et al., “A Flexible Authorization Framework,” gives advances in application areas, such as Internet-based transactions, cooperating coalitions, and workflow systems, which have brought new challenges to access control. In order to meet the diverse needs of emerging applications, it has become necessary to support multiple access control policies in one security domain. This chapter describes an authorization framework, referred to as the Flexible Authorization Framework (FAF) that is capable of doing so. FAF is a logic-based framework in which authorizations are specified in terms of a locally stratified rule base. FAF allows permissions and prohibitions to be included in its specification. FAF specifications can be changed by deleting and inserting its rules. We also describe FAF’s latest additions, such as revoking granted permissions, provisional authorizations, and obligations.

Chapter V by Rezgui et al., “Enforcing Privacy on the Semantic Web,” presents a reputation-based system for web environments aimed at an automatic process of privacy enforcement in a semantic web. Since web services and software agents exchange a large amount of semantically correlated information, the chapter presents a model for assigning a reputation to services. Reputation is a set of attributes built upon “how well” the services has per-

formed in its life cycle using a common perception on the service behavior. Reputation attributes are assigned to services using five criteria defined in the chapter related to security (permeability, authentication-based disclosure of information, correct delivery of data to authorized users, use of cryptography, and seniority seen as period of correct behavior). The architecture presented for a reputation management system sees a Reputation Manager, a set of Probing Agents, and a set of Service Wrappers. The system is distributed to enable support of peer-to-peer applications. Examples are provided in the field of e-government applications.

### **Section III: Data Distribution and Dissemination on the Net**

Chapter VI by Bertino et al., “Secure Data Dissemination,” considers the development of a new class of information-centered applications focused on the Selective Dissemination of Information (SDI). The purpose of these applications is the delivery of data to a large user community. “Selective” means that each user should not receive all the data but he/she may receive only specific portions of them. Such portions can be determined according to several factors, such as the user interests and needs, or the access control policies that the data source has in place. Additionally, SDI services can be classified by taking into account additional aspects, such as for instance the adopted distribution mode, the events starting up the distribution, and the network and architecture supporting the service. Due to these reasons, the chapter provides a taxonomy of SDI services and presents a detailed overview of the current approaches. Then, the chapter focuses on security issues for selective data dissemination services. In particular, focus is on four security properties: authenticity, integrity, confidentiality, and completeness, in the scope of Secure SDI applications is wide and heterogeneous. For instance, a relevant scenario for such kinds of applications is related to electronic commerce or digital libraries or electronic news (e.g., stock price, sport news, etc.). In such a case, users subscribe to a source and they can access information on the basis of the fee they have paid. Additionally, the service must ensure that contents are not eavesdropped on during transmission. Another important scenario for Secure SDI applications is data dissemination within an organization or community, where the delivery is controlled by security rules defined by system administrator(s), for instance, documents containing sensitive information about industrial projects.

Chapter VII by Fernandez-Medina et al., “Multimedia Security and Digital Rights Management Technology,” considers the crucial topic of multimedia content delivery applications on high bandwidth networks. It considers the pervasiveness of XML as a data interchange format, which has given origin to a number of standard formats for multimedia, such as SMIL for multimedia presentations, SVG for vector graphics, VoiceXML for dialog, and MPEG-21 and MPEG-7 for video. Innovative programming paradigms (such as the one of

web services) rely on the availability of XML-based markup and metadata in the multimedia flow in order to customize and add value to multimedia content distributed via the Net. In such a context, a number of security issues around multimedia data management need to be addressed. First of all, it is important to identify the parties allowed to use the multimedia resources, the rights available to the parties, and the terms and conditions under which those rights may be executed; this is fulfilled by the Digital Rights Management (DRM) technology. Secondly, a new generation of security and privacy models and languages is needed, capable of expressing complex filtering conditions on a wide range of properties of multimedia data. In this chapter, the general problem of multimedia security is analyzed, summarizing the most important XML-based formats for representing multimedia data; a language for expressing access control policies is presented and finally, the most important concepts of the DRM technology are discussed.

## **Section IV: Service Oriented Computing Frameworks**

Chapter VIII by Lin et al., “Data and Application Security for Distributed Application Hosting Services,” considers web and Internet services and the emergence of enabling techniques, such as J2EE and .NET, which have led to a trend toward distributed application hosting services (DAHSSs). Such hosting services, using rented Internet, computation power, and data storage space to clients are relatively a cheap and effective solution for achieving data and service availability, a balanced load on the servers, and increased scalability. However, these DAHSSs, implemented within the Internet environment, introduce many security concerns for the content and application owners. This chapter discusses security concerns for DAHSSs, the available security technologies and protocols at different tiers in the Internet information management hierarchy, and the open challenges.

Chapter IX by Fernandez et al., “Comparing the Security Architectures of Sun ONE and Microsoft .NET,” is an evaluation of claimed security features in a couple of products oriented to web service management. In fact, several companies have announced strategies for supporting web services, all using two basic reference architectures: Microsoft .NET or Sun ONE. Barely these architectures mention security, while the authors rightly point out that this aspect can be one of the fundamental success factors of the products. Therefore, the chapter examines the security features in .NET and ONE web services architectures, in particular, on how web service programs on specialized, shared systems are stored, on how user’s data are managed on these shared systems, e.g., on repositories or catalogs. Examined security features are confidentiality and integrity of the web service data, control on the code actions, access control (only paying subscribers can use the service), and service availability.

# Acknowledgments

The editors are thankful to partners in National and International projects dealing with security, such as the Demostene project for security in Public Administrations funded by Italian NRC, the MAIS (Multichannel Information Systems) Project funded by The Ministry of Education and Research, and to the WIDE and DEAFIN Projects funded by the UE.

*Maria Grazia Fugini, Politecnico di Milano, Italy*

*Carlo Bellettini, Politecnico di Milano, Italy*

**SECTION I:**

**CRYPTOGRAPHY**

## Chapter I

# Architectures for Advanced Cryptographic Systems

Guido Bertoni, Politecnico di Milano, Italy

Jorge Guajardo, Infineon Technologies AG, Germany

Christof Paar, Ruhr Universität Bochum, Germany

### ABSTRACT

*In the last 20-30 years, the world of modern cryptography has been largely dominated by traditional systems such as the Data Encryption Standard and the RSA algorithm. Such systems have provided a secure way for storing and transmitting information and they are nowadays incorporated in many network protocols and secure storage media. More recently, the increasing advance of crypto-analytical techniques and tools and the emergence of new applications, for example wireless communications and mobile computing, have stimulated the research and development of innovative cryptographic algorithms. These newer systems require a more detailed and sophisticated mathematical formalization and operations, which are not normally supported by general-purpose processors. For example, many basic operations required to implement recently proposed cryptographic algorithms, such as the Advanced Encryption Standard or Elliptic Curve Cryptosystems, are based on arithmetic in finite fields (or Galois fields). This chapter is, thus, intended to give an overview of such developments in modern cryptography. In particular, it aims at giving the*

*reader a comprehensive understanding of innovative cryptosystems, their basic structure, alternative existing hardware architectures to implement them, and their performance requirements and characterizations. Emphasis will be made throughout on two important cases: the Advanced Encryption Standard and Elliptic Curve Cryptosystems.*

## INTRODUCTION

It is widely recognized that data security will play a central role in the design of future IT systems. Although the PC had been the major driver of the digital economy until a few years ago, recently there has been a shift towards IT applications realized as embedded systems, and it is expected that this trend will continue as we advance into the 21st century. In addition, many of those applications either rely heavily on security mechanisms, including security for wireless phones, faxes, wireless computing, pay-TV, and copy protection schemes for audio/video consumer products and digital cinemas, or they will require security mechanisms to protect data, communications and our privacy. Thus, it is a pressing need to implement security measures and, in particular, cryptographic algorithms on platforms that are part of embedded systems.

Traditionally, ASICs have been common components in the design of embedded systems by providing the high performance, low power dissipation, and lower price per unit cost that many systems require. Furthermore, ASIC implementations of cryptographic algorithms are more secure than software ones because they cannot be as easily read or modified by an outside attacker. Nevertheless, ASIC implementations suffer from several drawbacks. Among those we can mention: (i) higher development costs and longer design cycles and (ii) lack of flexibility with respect to algorithm and parameter switching in fielded devices. These drawbacks are especially prominent in security applications, which are designed using new security protocol paradigms. Many of the new security protocols decouple the choice of cryptographic algorithm from the design of the protocol. Users of the protocol negotiate, on the fly, the choice of algorithm to use for a particular secure session. Thus, it would be desirable for the devices that will support these applications not only to support a single cryptographic algorithm and protocol, but also to be “algorithm agile”; that is, able to select from a variety of algorithms. For example, IPsec (the security standard for the Internet) allows applications to choose from a list of different symmetric and asymmetric ciphers.

In the mid-90s the use of reprogrammable components, in particular FPGAs (Field Programmable Gate Array), was introduced. FPGAs allowed for faster design cycles than ASICs because they enabled early functionality testing. Nonetheless, the performance and size of FPGAs did not permit them to substitute ASICs in most applications and thus, they were mainly used to

prototype embedded chips small enough to fit in the FPGA. In recent years, however, FPGA manufacturers have come closer to filling the performance gap between FPGAs and ASICs (Application Specific Integrated Circuit), enabling them not only to serve as fast prototyping tools, but also to become active players as components in embedded systems (Wong et al., 2002). The trend in both industry (see Altera Corporation, 2000; Altera Corporation, 2002a; Altera Corporation, 2002b; Chameleon Systems; Triscend Corporation; Xilinx Inc., 2002; Xilinx Inc., 2003) and academia (see Bondalapati & Prasanna, 2002; Hauser & Wawrzynek, 1997) is to develop chips which include either embedded components in them, such as memory, I/O controllers, and multiplier blocks, or both system reconfigurable components and programmable cores. The resulting processors/chips, which are not anymore a single part of an embedded system but rather can be used to develop the *whole* system, are known by various names ranging from hybrid architectures to Systems-on-Chip (SoC), Configurable System-on-Chip (CSoC), Reconfigurable Systems-on-Chip (RSoC), and Systems on Programmable Chip (SoPC), among others (Bondalapati & Prasanna, 2002). Thus, FPGAs and, in particular, reconfigurable devices are also integral parts in embedded system design.

From the above discussion, one can see that the security engineer is faced with the challenge of implementing cryptographic algorithms on both custom hardware and reconfigurable platforms. This chapter provides the reader with a self-contained overview of both traditional (DES and RSA) and newly introduced (AES and ECC) cryptographic algorithms and of the latest trends and architectures used to implement them on hardware platforms such as ASICs and FPGAs. We notice that the implementation of cryptographic systems presents several requirements and challenges. First, the performance of the algorithms is often crucial. One needs encryption algorithms to run at the communication link transmission rates or at fast enough rates that customers do not become dissatisfied. Second, in order to achieve such satisfactory performance, it is imperative to have a good understanding and knowledge of: (i) the encryption algorithms, (ii) the algorithms underlying their implementation (not necessarily the encryption algorithm but algorithms which are used to implement them, such as algorithms for finite field arithmetic), and (iii) the hardware platform. Finally, the security engineer also has to be aware of the latest trends in the design of encryption schemes as well as the latest attacks. This chapter makes emphasis on the implementation aspects. We provide several implementation approaches and compare them, thus allowing the reader to have a wide range of options for different applications. In other words, some applications might require the fastest possible implementation of the AES, without regard to power consumption and/or area, whereas others might want to be optimized for the last two parameters as long as an acceptable performance level is still achievable. Finally, we also hint at possible attacks on implementations and some solutions presented in the literature.

## Chapter Organization

The remainder of this chapter is organized as follows. We begin with a brief introduction to cryptography and the mathematical background needed to understand the encryption schemes described in latter sections. We make emphasis on definitions and when appropriate give relevant examples and refer the reader to other bibliographical sources for the proofs of theorems that we might use. The next two major sections involve discussions of symmetric and asymmetric cryptosystems. In particular, we discuss DES and AES as prime examples of symmetric schemes and RSA and ECC for the asymmetric case, as these are the most widely deployed algorithms in practical applications. We end this chapter with a short overview of attacks against the presented cryptographic schemes and their implementations as well as possible countermeasures.

## MATHEMATICAL BACKGROUND

This section should probably more appropriately be called “An Introduction to Finite Fields,” since these are, by far, the most widely used algebraic structure in the construction of cryptographic schemes. Examples include: the AES, the Diffie-Hellman key exchange protocol and those systems based on solving the difficulty of Discrete Logarithm (DL) problem, and elliptic curve cryptosystems. We refer the reader to Lidl and Niederreiter (1997) for a comprehensive treatment of finite fields.

**Definition 1.** Let  $S$  be a set. Then, the mapping from  $S \times S$  to  $S$  is called a binary operation on  $S$ . In particular, a binary operation is a rule that assigns ordered pairs  $(s, t)$ , with  $s, t \in S$ , to an element of  $S$ . Notice that under this definition the image of the mapping is required to be also in  $S$ . This is known as the *closure property*.

### Groups

**Definition 2.** A *group* is a set  $G$  together with a binary operation  $*$  on the set, such that the following properties are satisfied:

- (i) The group operation is associative. That is  $\alpha * (\beta * \gamma) = (\alpha * \beta) * \gamma$ , for all  $\alpha, \beta, \gamma \in G$ .
- (ii) There is an element  $\pi \in G$ , called the identity element, such that  $\pi * \alpha = \alpha * \pi = \alpha$  for all  $\alpha \in G$ .
- (iii) For all  $\alpha \in G$ , there is an element  $\alpha^l \in G$ , such that  $\alpha * \alpha^l = \alpha^l * \alpha = \pi$ . The element  $\alpha^l$  is called the inverse of  $\alpha$ . If the group also satisfies  $\alpha * \beta = \beta * \alpha$  for all  $\alpha, \beta \in G$ , then the group is said to be *commutative* or *abelian*. In the

Table 1: Notation for common group operations, where  $\alpha \in G$  and  $n$  and  $m$  are integers

Multiplicative Notation	Additive Notation
$\alpha^n = \alpha * \alpha * \alpha * \dots * \alpha$ ( $\alpha$ multiplied by itself $n$ times)	$n\alpha = \alpha + \alpha + \alpha + \dots + \alpha$ ( $\alpha$ added to itself $n$ times)
$\alpha^n = (\alpha^{-1})^n$	$-n\alpha = n(-\alpha)$
$\alpha^n * \alpha^m = \alpha^{n+m}$	$n\alpha + m\alpha = (n+m)\alpha$
$(\alpha^n)^m = \alpha^{nm}$	$n(m\alpha) = (nm)\alpha$

remainder of this chapter, we will only consider abelian groups unless we explicitly say something to the contrary. Note that we have used a multiplicative group notation for the group operation. If the group operation is written additively, then we talk about an additive group, the identity element is often associated with the zero ( $0$ ) element, and the inverse element of  $\alpha$  is written as  $-\alpha$ . Notation conventions are shown in Table 1.

**Example 1.** (i) The set of integers  $\mathbf{Z}$  forms an additive group with identity element  $0$ . (ii) The set of reals  $\mathbf{R}$  forms a group under the addition operation with identity element  $0$  and under the multiplication operation with identity element  $1$ . (iii) The integers modulo  $m$ , denoted by  $\mathbf{Z}_m$ , form a group under addition modulo  $m$  with identity element  $0$ . Notice that the group  $\mathbf{Z}_m$  is not a group under multiplication modulo  $m$ , since not *all* its elements have multiplicative inverses.

**Definition 3.** A group  $G$  is finite if the number of elements in it is finite, i.e., if its *order*, denoted  $|G|$ , is finite.

**Definition 4.** For  $n \geq 1$ , let  $\phi(n)$  denote the number of integers in the range  $[1, n]$  which are relatively prime (or co-prime) to  $n$  (i.e., an integer  $a$  is co-prime to  $n$  if  $\text{gcd}(a, n) = 1$ ). The function  $\phi(n)$  is called the *Euler phi function* or the *Euler totient function*. The Euler phi function satisfies the following properties:

- (i) If  $p$  is prime then  $\phi(p) = p - 1$ .
- (ii) The Euler phi function is multiplicative. In other words, if  $\text{gcd}(p, q) = 1$ , then  $\phi(pq) = \phi(p) \phi(q)$ .
- (iii) If  $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ , is the prime factorization of  $n$ , then  $\phi(n)$  can be computed as:

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

**Example 2.** Let the set of integers modulo  $m$  which are co-prime to  $m$  be denoted by  $\mathbf{Z}_m^*$ . Then, the set  $\mathbf{Z}_m^*$  under the operation of multiplication modulo  $m$  forms a group of order  $\phi(m)$  with identity element  $1$ . In particular, if  $m$  is prime then  $\phi(m) = |\mathbf{Z}_m^*| = m-1$ .

**Definition 5.** A group  $G$  is *cyclic* if there is an element  $\alpha \in G$  such that for each  $\beta \in G$ , there is an integer  $i$  such that  $\beta = \alpha^i$ . Such an element is called a generator of  $G$  and we write  $G = \langle \alpha \rangle$ . The order of  $\beta \in G$ , denoted  $\text{ord}(\beta)$ , is defined to be the least positive integer  $t$  such that  $\beta^t = \pi$ , where  $\pi$  is the identity element in  $G$ .

Notice the difference between the order of an element  $\alpha \in G$  ( $\text{ord}(\alpha)$ ) and the order of the group  $G$  ( $|G|$ ).

**Example 3.** (i) The multiplicative group of integers modulo  $11$ ,  $\mathbf{Z}_{11}^*$ , is a cyclic group with generators  $2$ ,  $2^3 = 8 \pmod{11}$ ,  $2^7 = 7 \pmod{11}$ , and  $2^9 = 6 \pmod{11}$ . Notice that the powers of two, which result in generators are co-prime to the order of  $\mathbf{Z}_{11}^*$ , i.e.,  $10$ . In fact, it can be shown that given a generator  $\alpha \in \mathbf{Z}_m^*$ ,  $\beta = \alpha^i \pmod{m}$  is also a generator if and only if  $\text{gcd}(i, \phi(m)) = 1$ . (ii) The additive group of integers modulo  $6$ ,  $\mathbf{Z}_6$ , has generators  $1$  and  $5$ .

## Rings and Fields

**Definition 6.** A ring,  $(R, +, *)$ , is a set  $R$  together with two binary operations on  $R$ , arbitrarily denoted  $+$  (addition) and  $*$  (multiplication), which satisfy the following properties:

- (i)  $(R, +)$  is an abelian group with identity element denoted by  $0$ .
- (ii) The operation  $*$  is associative, that is,  $\alpha^*(\beta*\gamma) = (\alpha*\beta)*\gamma$ , for all  $\alpha, \beta, \gamma \in R$ .
- (iii) There is a multiplicative identity element denoted by  $1$ , with  $0 \neq 1$ , such that for all  $a \in R$ ,  $\alpha*1 = 1*\alpha = \alpha$ .
- (iv) The operation  $*$  is distributive over the  $+$  operation. In other words,  $\alpha^*(\beta+\gamma) = (\alpha*\beta) + (\alpha*\gamma)$  and  $(\alpha+\gamma)*\alpha = (\beta*\alpha) + (\gamma*\alpha)$  for all  $\alpha, \beta, \gamma \in R$ .

If the operation  $*$  is also commutative, i.e.,  $\alpha*\beta = \beta*\alpha$ , then the ring is said to be commutative.

**Example 4.** (i) The set of integers  $\mathbf{Z}$  with the usual addition and multiplication operations is a commutative ring. Similarly, the set of rational numbers  $\mathbf{Q}$ , the set of reals  $\mathbf{R}$ , and the complex numbers  $\mathbf{C}$  are all examples of commutative rings with the usual addition and multiplication operations. (ii) The set  $\mathbf{Z}_m$  of integers modulo  $m$  with modulo  $m$  addition and multiplication operations is a commutative ring.

**Definition 7.** A field  $F$  is a commutative ring in which every non-zero element (i.e., all elements except for the  $0$  element) have multiplicative inverses.

A subset  $S$  of a field  $F$  which itself is a field with respect to the operations in  $F$  is called a subfield of  $F$ . In this case  $F$  is said to be an extension field of  $S$ .

Definition 7 implies that a field  $F$  is a set on which two binary operations are defined, called addition and multiplication, and which contains two elements,  $0$  and  $1$ , which satisfy  $0 \neq 1$ . In particular,  $(F, +, 0)$  is an abelian group with additive identity  $0$  and  $(F^*, *, 1)$  is an abelian group under the multiplication operation with  $1$  as the multiplicative identity ( $F^*$  is the set  $F$  without the element  $0$ ). The operations of addition and multiplication are related to each other via the distributivity law, i.e.,  $\alpha * (\beta + \gamma) = (\alpha * \beta) + (\alpha * \gamma)$  and  $(\beta + \gamma) * \alpha = (\beta * \alpha) + (\gamma * \alpha)$ , where the second property follows automatically from the fact that  $(F^*, *, 1)$  is an abelian group under multiplication.

**Example 5.** (i) The set of integers  $\mathbf{Z}$  with the usual addition and multiplication operations is *not* a field since not all its elements have multiplicative inverses. In fact only  $1$  and  $-1$  have multiplicative inverses. (ii) The set of rational numbers  $\mathbf{Q}$ , the set of reals  $\mathbf{R}$ , and the complex numbers  $\mathbf{C}$  are all examples of fields. (iii) The set  $\mathbf{Z}_m$  of integers modulo  $m$  with the modulo  $m$  addition and multiplication operations is a field if and only if  $m$  is prime. For example,  $\mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_5$ , etc., are all fields.

**Definition 8.** The characteristic of a field is said to be  $0$  if  $\overbrace{1 + 1 + 1 + \dots + 1}^{m \text{ times}}$  is never equal to  $0$  for any value of  $m \geq 1$ . Otherwise, the characteristic of a field is the least positive integer  $m$  such that  $\sum_{k=1}^m 1 = 0$ . It can be shown that if the characteristic  $m$  of a field is not  $0$  then  $m$  is a prime.

Definition 8 implies that  $\mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_5, \dots, \mathbf{Z}_p$  where  $p$  is prime are fields of characteristic  $p$ . We notice in particular that they are fields with a finite number of elements and thus they have received the name of *finite fields* or *Galois fields* after its discoverer Evariste Galois, French mathematician of the 18<sup>th</sup> century. The number of elements in the field is called the *order* of the field. Finally, it is worth mentioning that  $\mathbf{Z}_p$ , for  $p$  prime, are just but a few of the existing finite fields. To provide constructions for other finite fields we introduce the concept of polynomial rings.

**Example 6.** (i) If  $p$  is prime then we can find the inverse of any number  $a$  modulo  $p$  via Fermat's Little theorem which states that if  $\gcd(a, p) = 1$ , (this is always true if  $p$  is prime and  $a < p$ ) then  $a^{p-1} = 1 \pmod p$  and therefore it follows that  $a^{p-2}$  is the inverse of  $a$  modulo  $p$ . (ii) The inverse of  $3$  modulo  $7$  ( $3^{-1} \pmod 7$ )

can be found as  $3^5 = 243 \equiv 5 \pmod{7}$ . A quick check verifies our assertion:  $3 \cdot 5 = 15 \equiv 1 \pmod{7}$ . (iii) A second way to find the inverse of an integer modulo  $p$  is to use the extended Euclidean algorithm which guarantees that we can find integers  $u$  and  $v$  such that  $a \cdot v + p \cdot u = d = \gcd(a, p)$ . It follows that if  $\gcd(a, p) = 1$ , then we can find the inverse of  $a$  modulo  $p$  as  $a \cdot v + p \cdot u = 1 \Rightarrow a \cdot v \equiv 1 \pmod{p} \Rightarrow a^{-1} \equiv v \pmod{p}$ .

## Polynomial Rings

**Definition 9.** If  $R$  is a commutative ring, then a polynomial in the indeterminate  $x$  over  $R$  is an expression of the form:

$$A(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

where each  $a_i \in R$  and  $n \geq 0$ . As in classical algebra, the element  $a_i$  is called the coefficient of  $x^i$  in  $A(x)$  and the largest  $n$  for which  $a_n \neq 0$  is called the degree of  $A(x)$ , denoted by  $\deg(A(x))$ . The coefficient  $a_n$  is called the leading coefficient of  $A(x)$ . If  $a_n = 1$  then  $A(x)$  is said to be a *monic* polynomial. If  $A(x) = a_0$  then the polynomial is a constant polynomial and has degree 0 whereas if  $A(x) = 0$  (i.e., all coefficients of  $A(x)$  are equal to 0), then  $A(x)$  is called the zero polynomial and for mathematical convenience is said to have degree  $-\infty$ .

**Example 7.** Two polynomials  $A(x) = \sum_{i=0}^n a_i x^i$  and  $B(x) = \sum_{i=0}^n b_i x^i$  over  $R$  are said to be equal if and only if  $a_i = b_i$  for  $0 \leq i \leq n$ . The sum of two polynomials is realized in the familiar way as:

$$A(x) + B(x) = \sum_{i=0}^n (a_i + b_i) x^i$$

**Example 8.** The product of two polynomials  $A(x) = \sum_{i=0}^n a_i x^i$  and  $B(x) = \sum_{j=0}^m b_j x^j$  over  $R$  is defined as follows  $C(x) = \sum_{k=0}^{n+m} c_k x^k = A(x) * B(x)$  where

$$c_k = \sum_{\substack{i+j=k \\ 0 \leq i \leq n \\ 0 \leq j \leq m}} a_i b_j$$

and addition and multiplication of coefficients is performed in  $R$ .

Together with the operations of addition and multiplication defined as above it is easily seen that the set of polynomials over  $R$  forms a ring.

**Definition 10.** Let  $R$  be commutative ring. Then the set of polynomials over  $R$  with addition and multiplication of polynomials defined as in Example 6 is called a *polynomial ring* and we denoted by  $R[x]$ . Notice the difference in notation between the set of all polynomials over  $R$ , together with the operations of addition and multiplication of polynomials, denoted by  $R[x]$  (square brackets) and one element of  $R[x]$ , say  $A(x)$ , which we denote also with capital letters but round parenthesis. In the remainder of this work we will only consider polynomial rings  $F[x]$  defined over  $F$ , where  $F$  is a field.

Elements of  $F[x]$  share many properties with the integers. Thus, it is possible to talk about divisibility of a polynomial by other polynomial. In particular, a polynomial  $B(x) \in F[x]$  is said to divide another polynomial  $A(x) \in F[x]$  if there exists a polynomial  $C(x) \in F[x]$  such that  $A(x) = B(x) * C(x)$ . Thus, we say that  $B(x)$  is a divisor of  $A(x)$  or that  $A(x)$  is a multiple of  $B(x)$ , or that  $A(x)$  is divisible by  $B(x)$ . The idea of divisibility leads to a division algorithm for polynomials. In fact, we can prove that for any  $B(x) \neq 0$  in  $F[x]$ , and for any  $A(x) \in F[x]$ , we can find polynomials  $Q(x)$  and  $R(x)$  such that  $A(x) = Q(x) * B(x) + R(x)$  where  $\deg(R(x)) < \deg(B(x))$  and  $Q(x)$  and  $R(x)$  are unique.

**Definition 11.** A polynomial  $P(x) \in F[x]$  is said to be irreducible over  $F$  if  $P(x)$  has positive degree and writing  $P(x) = B(x)*C(x)$  implies that either  $B(x)$  or  $C(x)$  is a constant polynomial. Otherwise  $P(x)$  is said to be reducible.

Much in the same way as with the integers, we say that if  $A(x), B(x) \in F[x]$ , then  $A(x)$  is said to be congruent to  $B(x)$  modulo  $T(x)$  if  $T(x)$  divides  $A(x) - B(x)$ , written  $T(x)|(A(x)-B(x))$ . The congruency relation is denoted as  $A(x) \equiv B(x) \pmod{T(x)}$ . For a fixed polynomial  $T(x)$ , the equivalence class of a polynomial  $A(x) \in F[x]$  is the set of all polynomials in  $F[x]$  congruent to  $A(x)$  modulo  $T(x)$ . It can be shown that the relation of congruency modulo  $T(x)$  partitions  $F[x]$  into equivalence classes. In particular, we can find a unique representative for each equivalence class as follows. From the division algorithm for polynomials we know that given any two polynomials  $A(x)$  and  $T(x)$  we can find *unique* polynomials  $Q(x)$  and  $R(x)$  where  $\deg(R(x)) < \deg(T(x))$ . Hence, every polynomial  $A(x)$  is congruent modulo  $T(x)$  to a unique polynomial  $R(x)$  of degree less than  $T(x)$ . Thus, we choose the unique polynomial  $R(x)$  to be the unique representative for equivalence class of polynomials containing  $A(x)$ . We denote by  $F[x]/(T(x))$  the set of equivalence classes of polynomials in  $F[x]$  of degree less than  $m = \deg(T(x))$ . It turns out that  $F[x]/(T(x))$  is a commutative ring and if  $T(x)$  is irreducible over  $F$ , then  $F[x]/(T(x))$  is a field.

**Definition 12.** An element  $\alpha \in F$ , is said to be a root (or zero) of the polynomial  $P(x) \in F[x]$  if  $P(\alpha) = 0$ .

## Construction of Finite Fields $GF(p^m)$

In previous sections, we saw that  $Z_p$ , for  $p$  prime, was an example of a finite field (also called Galois field  $GF(p)$ ) with  $p$  elements where addition and multiplication were the standard addition and multiplication modulo  $p$  operations and inversion could be achieved via Fermat's Little theorem or using the extended Euclidean algorithm for integers. In this section, we construct the remaining finite fields.

**Definition 13.** Let  $m$  be a positive integer and  $P(x)$  be an irreducible polynomial of degree  $m$  over  $GF(p)$ . Moreover, let  $\alpha$  be a root of  $P(x)$ , i.e.,  $P(\alpha) = 0$ . Then, the Galois field of order  $p^m$  and characteristic  $p$ , denoted  $GF(p^m)$ , is the set of polynomials  $a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \dots + a_2\alpha^2 + a_1\alpha + a_0$ , with  $a_i \in GF(p)$  together with addition and multiplication defined as follows. Let  $A(\alpha), B(\alpha) \in GF(p^m)$ , with  $A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i$  and  $B(\alpha) = \sum_{i=0}^{m-1} b_i \alpha^i$  and  $a_i, b_i \in GF(p)$  then:

$$(i) \quad C(\alpha) = A(\alpha) + B(\alpha) = \sum_{i=0}^{m-1} (a_i + b_i) \alpha^i \in GF(p^m)$$

(ii)  $C(\alpha) = A(\alpha) * B(\alpha) = \sum_{i=0}^{m-1} c_i \alpha^i \in GF(p^m)$  as follows: Define  $\overline{C(\alpha)}$  to be the result of multiplying  $A(\alpha)$  by  $B(\alpha)$  via standard polynomial multiplication as described in Example 8. Thus,  $\overline{C(\alpha)}$  is a polynomial of degree  $2m-1$ . Then, we define  $C(\alpha)$  to be  $\overline{C(\alpha)}$  modulo  $P(x)$ , i.e.,  $C(\alpha) \equiv \overline{C(\alpha)} \pmod{P(x)}$ . Notice that  $C(\alpha)$  can be found since the division algorithm guarantees that we can write  $\overline{C(\alpha)}$  as  $\overline{C(\alpha)} = P(\alpha)Q(\alpha) + C(\alpha)$  where  $\deg(C(\alpha)) < m$ . Since  $\alpha$  satisfies  $P(\alpha) = 0$ , we have that  $\overline{C(\alpha)} \equiv C(\alpha) \in GF(p^m)$ .

**Example 9.** Let  $p = 2$  and  $P(x) = x^4 + x + 1$ . Then,  $P(x)$  is irreducible over  $GF(2)$ . Let  $\alpha$  be a root of  $P(x)$ , i.e.,  $P(\alpha) = 0$ , then the Galois field  $GF(2^4)$  is defined by:

$$GF(2^4) = \{a_3 \alpha^3 + a_2 \alpha^2 + a_1 \alpha + a_0 \mid a_i \in GF(2)\}$$

together with addition and multiplication as defined in Definition 13. The Galois field  $GF(2^4)$  is of characteristic 2 and has order  $2^4 = 16$ , in other words, it has 16 elements.

To add  $\alpha^3 + 1$  and  $\alpha^3 + \alpha^2 + 1$  we simply perform polynomial addition and reduce the coefficients of the resulting polynomial modulo 2. Thus,  $(\alpha^3 + 1) +$

Table 2: Representation of elements of  $GF(2^4)$ 

As a 4-tuple	As a polynomial	As a power of $\alpha$
0000	0	0
0001	1	$\alpha^{15} \equiv 1$
0010	$\alpha$	$\alpha$
0011	$\alpha+1$	$\alpha^4$
0100	$\alpha^2$	$\alpha^2$
0101	$\alpha^2+1$	$\alpha^8$
0110	$\alpha^2+\alpha$	$\alpha^5$
0111	$\alpha^2+\alpha+1$	$\alpha^{10}$
1000	$\alpha^3$	$\alpha^3$
1001	$\alpha^3+1$	$\alpha^{14}$
1010	$\alpha^3+\alpha$	$\alpha^9$
1011	$\alpha^3+\alpha+1$	$\alpha^7$
1100	$\alpha^3+\alpha^2$	$\alpha^6$
1101	$\alpha^3+\alpha^2+1$	$\alpha^{13}$
1110	$\alpha^3+\alpha^2+\alpha$	$\alpha^{11}$
1111	$\alpha^3+\alpha^2+\alpha+1$	$\alpha^{12}$

$(\alpha^3+\alpha^2+1) = \alpha^2$ . Similarly,  $(\alpha^3+1)$  multiplied by  $(\alpha^3+\alpha^2+1)$  is obtained as  $(\alpha^3+1)*(\alpha^3+\alpha^2+1) = \alpha^6+\alpha^5+\alpha^3+\alpha^2+1 \equiv \alpha^3+\alpha^2+\alpha+1 \pmod{P(\alpha)}$ .

Notice that  $GF(2^4)^*$  in other words  $GF(2^4)$  minus the zero element, is a cyclic group of order 15 generated by  $\alpha$ , thus we can write  $GF(2^4)^* = \langle \alpha \rangle$ .

We end this section with some basic facts about finite fields.

- (i) (Existence and uniqueness of finite fields) If  $F$  is a finite field then  $F$  contains  $p^m$  elements for some prime  $p$  and positive integer  $m \geq 1$ . For every

prime power  $p^m$ , there is a unique, up to isomorphism, finite field of order  $p^m$ . The finite field is denoted as  $GF(p^m)$ . Informally speaking, two finite fields are isomorphic if they are structurally the same, although the representation of their field elements may be different.

- (ii) If  $GF(q)$  is a finite field of order  $q=p^m$ ,  $p$  a prime, then the characteristic of  $GF(q)$  is  $p$ . In addition,  $GF(q)$  contains a copy of  $GF(p)$  as a subfield. Hence  $GF(q)$  can be viewed as an extension of  $GF(p)$  of degree  $m$ .
- (iii) Let  $GF(q)$  a finite field of order  $q = p^m$ , then every subfield of  $GF(q)$  has order  $p^n$  for some positive divisor  $n$  of  $m$ . Conversely, if  $n$  is a positive divisor of  $m$ , then there is exactly one subfield of  $GF(q)$  of order  $p^n$ . An element  $A \in GF(q)$  is in the subfield  $GF(p^n)$  if and only if  $A^{p^n} = A$ . The non-zero elements of  $GF(q)$  form a group under multiplication called the multiplicative group of  $GF(q)$ , denoted  $GF(q)^*$ . In fact  $GF(q)^*$  is a cyclic group of order  $q-1$ . Thus,  $A^q = A$  for all  $A \in GF(q)$ . A generator of  $GF(q)^*$  is called a primitive element of  $GF(q)$ .
- (iv) Let  $A \in GF(q)$ , with  $q=p^m$ , then the multiplicative inverse of  $A$  can be computed as  $A^{-1} = A^{q-2}$ . Alternatively, one can use the extended Euclidean algorithm for polynomials to find polynomials  $S(\alpha)$  and  $T(\alpha)$  such that  $S(\alpha)A(\alpha) + T(\alpha)P(\alpha) = 1$ , where  $P(x)$  is an irreducible polynomial of degree  $m$  over  $GF(p)$ . Then,  $A^{-1} = S(\alpha)$ .
- (v) If  $A, B \in GF(q)$  is a finite field of characteristic  $p$ , then  $(A + B)^{p^t} = A^{p^t} + B^{p^t}$  for all  $t \geq 0$ .

## CRYPTOGRAPHY AND ITS IMPLEMENTATION

Cryptography involves the study of mathematical techniques that allow the practitioner to achieve or provide the following objectives or services (Menezes et al., 1997):

- Confidentiality is a service used to keep the content of information accessible to only those authorized to have it. This service includes both protection of all user data transmitted between two points over a period of time as well as protection of traffic flow from analysis.
- Integrity is a service that requires that computer system assets and transmitted information be capable of modification only by authorized users. Modification includes writing, changing, changing the status, deleting, creating, and the delaying or replaying of transmitted messages. It is important to point out that integrity relates to active attacks and, therefore, it is concerned with detection rather than prevention. Moreover, integrity

can be provided with or without recovery, the first option being the more attractive alternative.

- Authentication is a service that is concerned with assuring that the origin of a message is correctly identified. That is, information delivered over a channel should be authenticated as to the origin, date of origin, data content, time sent, etc. For these reasons this service is subdivided into two major classes: entity authentication and data origin authentication. Notice that the second class of authentication implicitly provides data integrity.
- Non-repudiation is a service that prevents both the sender and the receiver of a transmission from denying previous commitments or actions.

These security services are provided by using cryptographic algorithms. There are two major classes of algorithms in cryptography: Private-key or symmetric-key algorithms and Public-key algorithms. The next two sections will describe them in detail.

## Symmetric-Key Algorithms

Private-key or symmetric-key algorithms are algorithms in which the encryption and decryption key is the same, or where the decryption key can easily be calculated from the encryption key and vice versa. The main function of these algorithms, which are also called secret-key algorithms, is encryption of data, often at high speeds. Private-key algorithms require the sender and the receiver to agree on the key prior to the communication taking place. The security of private-key algorithms rests on the key; divulging the key means that anyone can encrypt and decrypt messages. Therefore, as long as the communication needs to remain secret, the key must remain secret.

There are two types of symmetric-key algorithms that are commonly distinguished: block ciphers and stream ciphers (Schneier, 1996). Block ciphers are encryption schemes in which the message is broken into strings (called blocks) of fixed length and encrypted one block at a time. Examples include the Data Encryption Standard (DES) (NIST FIPS PUB 46-3, 1999), the International Encryption Standard (IDEA) (Lai et al., 1991; Massey & Lai, 1992), and the Advanced Encryption Standard (AES) (NIST FIPS PUB 197, 2001). Note that, due to its short block size and key length, DES expired as a U.S. standard in 1998, and that the National Institute of Standards (NIST) selected Rijndael algorithm as the AES in October 2000. AES has a block size of 128 bits and the ability to support 128-, 192- and 256-bit long keys. Stream ciphers operate on a single bit of plaintext at a time. In some sense, they are block ciphers having block length equal to one. They are useful because the encryption transformation can change for each symbol of the message being encrypted. In particular, they are useful in situations where transmission errors are highly probable because they do not have error propagation. In addition, they can be used when the data must

be processed one symbol at a time because of lack of equipment memory or limited buffering. In this chapter, we are only concerned with block ciphers, thus, stream ciphers are treated no longer.

It is important to point out that the trend in modern symmetric-key cipher design has been to optimize the algorithms for both efficient software and hardware implementation, in contrast to DES which was designed with hardware implementations in mind. These design criteria are evident if one looks at the performance of the AES on different platforms. The internal AES operations can be broken down into 8-bit operations, which is important because many cryptographic applications run on smart cards, which are traditionally based on 8-bit CPUs. Furthermore, one can combine certain steps to get a suitable performance in the case of 32-bit platforms. At the same time, AES implementations can easily achieve speeds in the Gbits/sec range when implemented on hardware platforms such as ASICs or FPGAs. Finally, notice that one of the major issues with symmetric-key systems is the need to find an efficient method to agree on and exchange the secret keys securely (Menezes et al., 1997). This is known as the key distribution problem. Diffie & Hellman (1976) proposed a new concept that would revolutionize cryptography as it was known at the time. This new concept was called public-key cryptography.

## Public-Key Algorithms

Public-key (PK) cryptography is based on the idea of separating the key used to encrypt a message from the one used to decrypt it. Anyone that wants to send a message to party  $A$  can encrypt that message using  $A$ 's *public key* but only  $A$  can decrypt the message using his/her *private key*. In implementing a public-key cryptosystem, it is understood that  $A$ 's private key should be kept secret at all times. Furthermore, even though  $A$ 's public key is publicly available to everyone, including  $A$ 's adversaries, it is impossible for anyone, except  $A$ , to derive the private key (or at least to do so in any reasonable amount of time).

In general, one can divide practical public-key algorithms into three families:

- Algorithms based on the *integer factorization problem*: given a positive integer  $n$ , find its prime factorization. RSA (Rivest et al., 1978), the most widely used public-key encryption algorithm, is based on the difficulty of solving this problem.
- Algorithms based on the *discrete logarithm problem in finite fields*: given  $\alpha, \beta \in GF(q)$ , find  $x$  such that  $\beta = \alpha^x$ . The Diffie-Hellman key exchange protocol is based on this problem as well as many other protocols, including the Digital Signature Algorithm (DSA).
- Algorithms based on the *discrete logarithm in the group of points of an elliptic curve*: given two points  $P$  and  $Q$  on an elliptic curve  $E$ , find an integer  $k$  such that  $Q = kP$ . Elliptic curve cryptosystems (ECC) are the most

recent family of practical public-key algorithms, but are rapidly gaining acceptance. Notice that they have become part of standards such as the ANSI X9.62-1998, the FIPS 186-2 Digital Signature Standard, which includes ECDSA, and the IEEE P1363 Standard for Public-Key Cryptography. Due to their reduced processing needs, elliptic curves are especially attractive for embedded applications.

Despite the differences between these mathematical problems, all three algorithm families have something in common: they all perform complex operations on very large numbers, typically 1024-2048 bits in length for systems based on the integer factorization problem, i.e., RSA, and discrete logarithm problem in finite fields or 160-256 bits in length for elliptic curve based systems. Notice that the most common operation performed in public-key schemes is modular exponentiation, i.e., the operation  $a^x \bmod n$ , or point multiplication in the case of elliptic curves, i.e., computing  $kP$  by adding  $P$  to itself  $k$  times. Performing such an exponentiation (or point multiplication) with 1024-bit long operands (or 160-bit operands) is extremely computationally intensive and thus, it requires a careful selection of methods that take advantage of the characteristics of the underlying cryptographic scheme.

Public-key cryptosystems solve in a very elegant way the key distribution problem of symmetric-key schemes. However, PK systems have a major disadvantage when compared to private-key schemes. As stated above, public-key algorithms are very arithmetic intensive and, if not properly implemented or if the underlying processor has a poor integer arithmetic performance, this can lead to poor system performance. Even when properly implemented, all PK schemes proposed to date are several orders of magnitude slower than the best-known private-key schemes. Hence, in practice, cryptographic systems are a mixture of symmetric-key and public-key cryptosystems. Usually, a public-key algorithm is chosen for key establishment and authentication through digital signatures, and then a symmetric-key algorithm is chosen to encrypt communications and data transfers, achieving in this way high throughput rates.

## BLOCK CIPHER IMPLEMENTATION

In this section we illustrate the basic structure of symmetric block cipher algorithms. A symmetric block cipher can be viewed as a function with two types of inputs, the data blocks to be encrypted and the secret key, and one output, the encrypted data blocks. All block ciphers have as an input plaintext and as an output ciphertext (encrypted text) of the same size in bits. The adjective symmetric indicates that the same (secret) key is used both for encryption and decryption. Standardized block ciphers, such as DES and AES, are widely used since they guarantee an appropriate security level for particular applications. At

the same time, in most cases there is no known *practical* attack that can be performed with complexity better than that of an exhaustive search. In addition, they can be easily implemented in both software and hardware, and they admit different implementation choices; for instance, different algorithmic versions allowing a cost/performance trade-off (code size versus time latency in software, silicon area versus time latency/throughput in hardware).

Symmetric ciphers can be modeled according to a layered structure. At the bottom of the model are the operations directly performed on the data elements (bits, bytes or words): these include Galois Field operations, such as addition, multiplication, and inversion of field elements as well as bit permutations. The set of operations used by the encryption algorithm is organized in a macro-function called the round transformation, or simply the round, which depends on the specific cryptographic algorithm and, hence, it will be explained later for the cases of DES and AES in detail. The round transformation is the building block used to implement the two basic services necessary in the cryptosystem, encryption and decryption of data, and one additional internal service, usually called key schedule, which is dedicated to processing the secret key. A block cipher encrypts data in blocks of fixed size. Thus, in order to process messages that exceed the block length, so-called *modes of operation* are introduced. Generally the size of the input/output data block is 64 or 128 bits. Some algorithms use or admit larger data blocks, but most modern block ciphers (for example all AES candidates) have a block length of 128 bits. The block size is mainly driven by security requirements but also complexity of implementation and performance. For example, in a software implementation, data blocks to be processed larger than 128 bits would require the allocation of too many processor registers.

The number of rounds in a block cipher varies greatly and depends on the cipher's design criteria and on what the designers considered an appropriate security level. For example, DES has 16 rounds, the AES accepts 10, 12, and 14 rounds depending on the key size, and Serpent had 32 encryption rounds. From the number of rounds, it is possible to obtain a range of values for the throughput of the system. For example, in hardware implementations, if a dedicated functional unit able to execute a round in one clock cycle is available and the designer has enough area resources, it is possible to instantiate all rounds of the cipher by means of as many functional units as necessary and pipeline the system, thus achieving throughputs in the order of gigabits per second. In the case of software implementations on general purpose CPUs, most encryption algorithms process the input data block in hundreds or thousands of clock cycles, thus reaching only throughputs in the order of megabits bits per second.

All symmetric block cipher algorithms share a common structure as shown in *Figure 2*. They are not implemented as a single function mapping the input data block to the output; rather, they consist of a macro-function that is applied

Figure 1: Generic symmetric cipher algorithm as a layered model

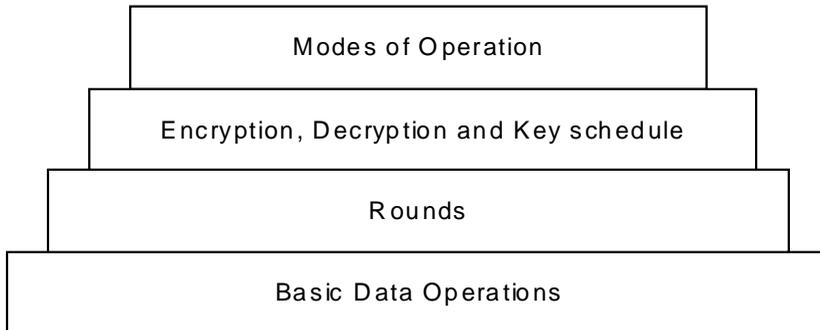
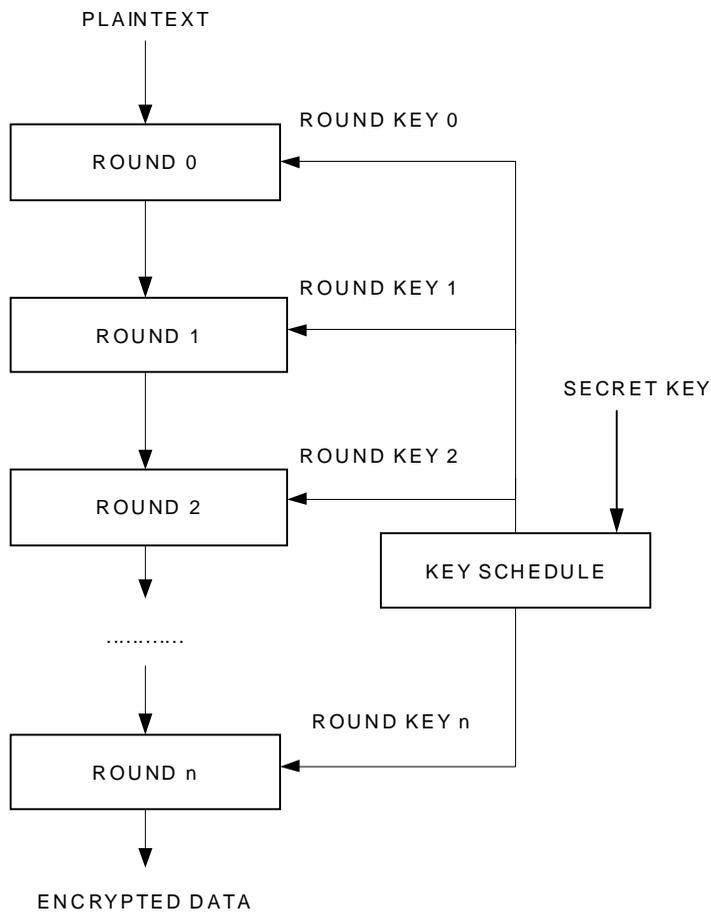


Figure 2: General structure of a block cipher algorithm

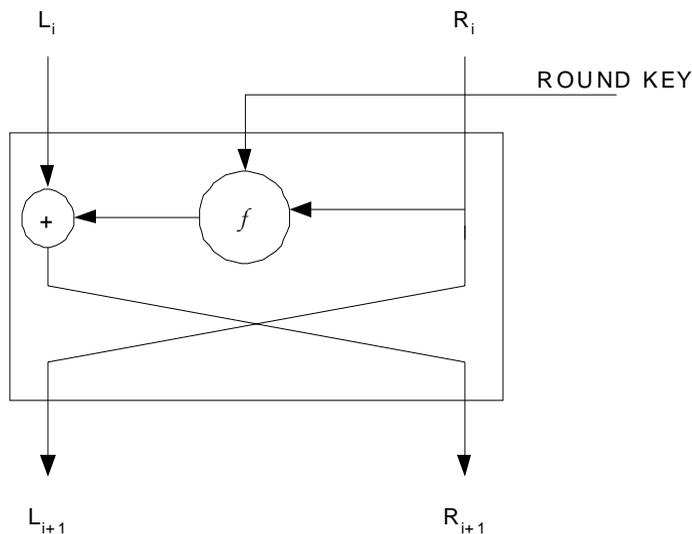


iteratively to the input data. As mentioned previously, this macro-function is called the round transformation. Thus, the structure of a block cipher algorithm can be viewed as a *for loop*: at the beginning the input of the algorithm is the plaintext, which is subsequently processed by the *loop body*, i.e., the *round*. The number of rounds applied to transform (encrypt or decrypt) the input data block is fixed and it is a constant of the algorithm (actually most modern algorithms admit two or more choices, which are chosen according to the desired security level). It is important to point out that the internal operations that make up the round should exhibit certain characteristics. In particular, at least one of the internal transformations should be dependent on the secret key and at least one of the internal transformations should be highly non-linear. We refer to Schneier (1996) for more details regarding the design of block cipher algorithms.

In order to increase the security level of the algorithm, the secret key is not used as it is in every round; rather, it is transformed. The processing of the secret key is called key schedule or key expansion. The data blocks extracted or derived from the secret key and then used by the rounds are called round keys or sub-keys.

One of the most common structures used in the design of block cipher algorithms is the Feistel network. The architecture of a Feistel network is depicted in *Figure 3*. The input block is divided into two halves, called left and right half or part. A round processes the right part of the input block through a non-linear function involving the round key, and the output of the non-linear function is then added to the left part. The result of the addition becomes the right

*Figure 3: The structure of a round organized as a Feistel network*



part of the next round, while the left part of the next round is the right part of the previous round, which was left unprocessed.

A relevant property of this cipher design is that the  $f$ -function should be non-invertible, even though the round itself is invertible due to its inner structure. In fact, it is possible to calculate  $L_i$  and  $R_i$  starting from  $L_{i+1}$  and  $R_{i+1}$  provided that the round key is available.

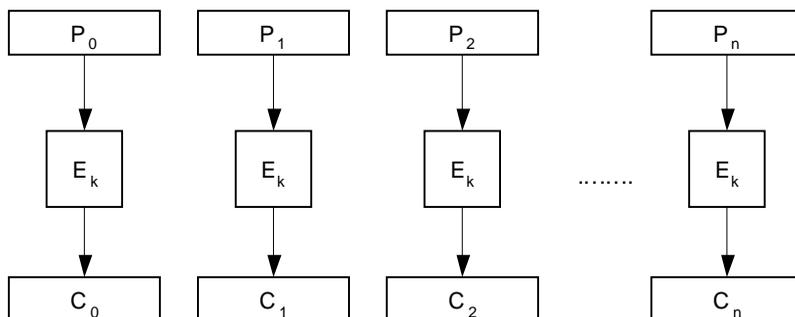
Some important block ciphers designed as Feistel networks are: DES, MISTY/Kasumi, (proposed in 3GPP), FEAL, GOST (the USSR DES), and Blowfish. The Feistel network is not the only possible architecture for designing a symmetric block cipher algorithm. For example, the Substitution Permutation Network (SPN) is another structure commonly used in modern block cipher design. The Advanced Encryption Standard (AES), which will be presented later, is precisely an SPN cipher algorithm. Usually SPN are based on the iterative application of a round primitive as well. Similarly to Feistel network-based ciphers, the secret key is not directly used, rather a key schedule algorithm is performed to derive the round keys, which are used directly in the encryption and decryption processes.

The general structure of a block cipher algorithm is therefore organized as a repetition of a single macro-function, the round, thus allowing for different implementation options. For instance, one can implement the round structure once and reuse it. This implementation strategy yields a benefit in terms of cost reduction. To reach a high performance in hardware, it is possible to instantiate all the rounds requested by the algorithm and pipeline them, thus reaching an extremely high throughput, as it is required in modern high-end network servers, capable of supporting thousands of simultaneous encrypted network channels. This strategy is performed at the additional cost of extra circuit area. So far we have seen basic design principles used to build block ciphers. In the next paragraph, we will look at how to process data whose size is larger than the input block of the algorithm.

The simplest way to use a block cipher algorithm consists in dividing the plaintext bit sequence into blocks, each block of the same size as the size of the algorithm's input data block and then encrypt the obtained blocks in succession. This mode of operation is the simplest one and it is referred to as the Electronic Code Book (ECB). ECB is quite simple but it may expose the encryption process to some weaknesses. If for instance there are two identical plaintext blocks, then they are mapped to identical cipher text blocks if the same key is used. This observation can be used to mount an attack which can possibly recover the encryption key.

In order to avoid such attacks, other modes of operation have been introduced. Among the most commonly used ones, we find the Cipher Block Chaining (CBC) mode. In this case the plaintext is not encrypted as it is, rather the output of the previous encryption operation is bit-wise XORed with the

Figure 4: ECB mode of operation



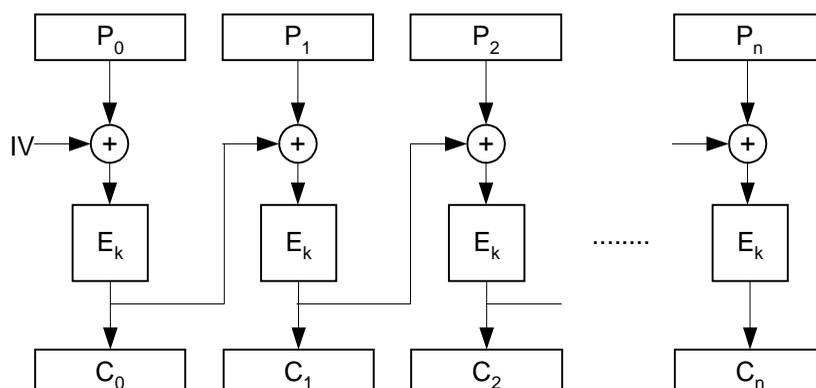
current plaintext block. Such a procedure yields a randomization of the blocks being encrypted. For the first block an initial vector (IV) is used, which is a random data block; clearly, it is necessary to exchange the IV between the two parties before starting the whole encryption process.

Notice that if the size of the plaintext is not a multiple of the block size, the plaintext is padded. The simplest way is to append as many bits as required to reach the closest bit size multiple of the block size. Other padding techniques exist as well, see Chapter 9 in Menezes et al. (1997).

Another mode of operation of interest in applications requiring fast encryption rates and high security is the “counter mode.” As in CBC, an Initial Vector (IV) is necessary. The IV is encrypted directly and added to the first block of plaintext. The IV is then incremented by one (the initialization vector is simply a counter, thus the name counter mode), encrypted again and added to the second block of plaintext, and so on for all plaintext blocks (from time to time the IV can be reinitialized). The counter mode method has been recently included in the IPsec specification. It is particularly interesting because it makes possible to prepare a stream of “encrypted noise” to be added in advance to the plaintext for improving security. A second interesting feature of the counter mode is that the decryption process does not require the presence of a dedicated decryption unit (or of a decryption procedure in the case of a software implementation): to decrypt the encrypted text it suffices to know the IV, encrypt it and subtract it from the encrypted text (which again is a bit-wise XOR operation), thus recovering the original plaintext. This property is particularly interesting for low cost CPUs, where decryption may happen to be slower in comparison with encryption.

The new encryption standard AES includes several other modes of operation. For more details we refer to NIST (2001c). Some of these modes of operation are particularly interesting since they combine the possibility to encrypt the plaintext and to compute a Message Authentication Code (MAC). The MAC

Figure 5: CBC mode of operation



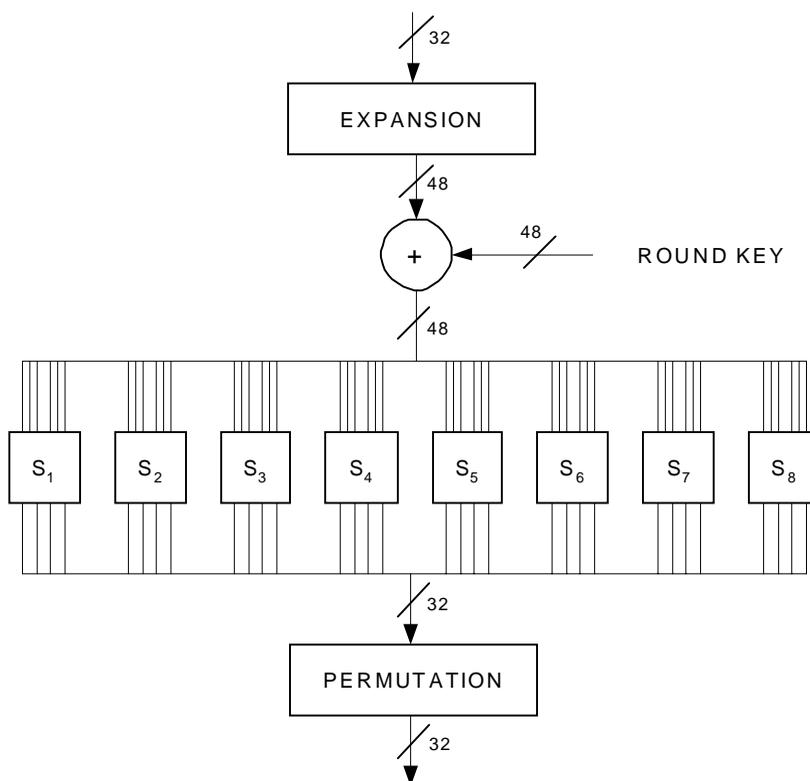
is a small additional group of bits, typically of about the same size as one data block, which allows checking the integrity of the plaintext.

## Data Encryption Standard (DES)

The Data Encryption Standard algorithm was developed at IBM. However, the documents that describe the design of DES are still classified. There have been a lot of rumors reported regarding the history and motivations of the design of DES, especially with respect to the structure of the S-BOXes. It has been said that the National Security Agency (NSA) altered the original IBM S-BOX, hiding in it a trapdoor. Notice that it is well known that NSA altered the S-BOXes originally submitted by IBM, what is not known is the reason for changing them. It was also said that the S-BOX was altered by NSA as a preventive countermeasure in case IBM had hidden their own trapdoors in it. Nevertheless, all such claims remain unsubstantiated to this day. In addition, the 56-bit long key size from the beginning caused a lot of controversy as it was considered as a compromise, large enough to stop ordinary attacks, but small enough to allow NSA to successfully carry out a brute force attack, even by means of the still limited computing power that was available in the 1970s.

The DES standard has been, and via the triple-DES extension will remain for some time, the most widely used symmetric block cipher algorithm in the last twenty years. It is still nowadays the algorithm supporting most of the cryptographic computing load all over the world. The most famous protocol using DES is the Secure Socket Layer (SSL). The SSL protocol is used to secure connections over the Internet, and it is supported by every web browser. Moreover, DES is one of the mandatory algorithms in the IPsec protocol, the secure version of the popular IP protocol, which is one of the two basic components of the TCP/IP protocol suite (the other component is TCP).

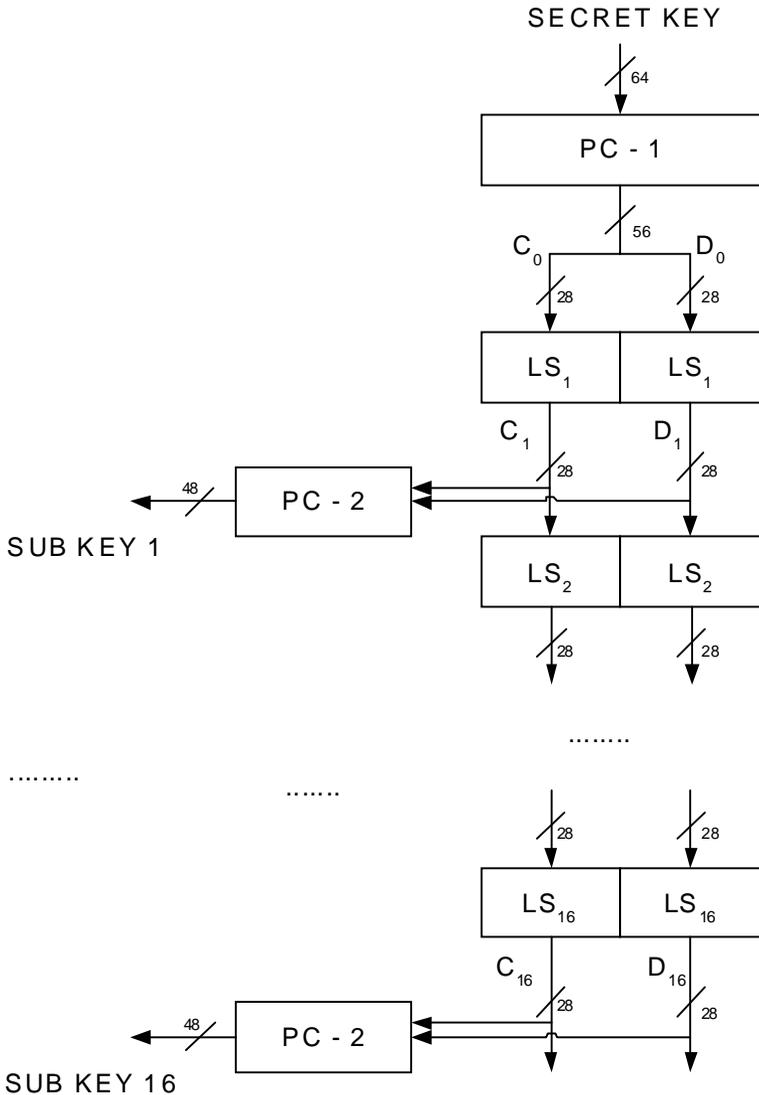
Figure 6: DES core function



DES was designed as a typical Feistel network. The algorithm maps 64 bits of plaintext to 64 bits of encrypted text. The input data block is first processed by an initial permutation, which simply changes the position of the input bits without any computation. Then the basic round is applied sixteen times to the data block. At the end of the last round the data block is processed by a final permutation that is exactly the inverse of the initial permutation. The secret key of DES is a block of 64 bits, but 8 of these bits are parity bits, introduced to guarantee error resilience. This effectively reduces the key size to 56 bits.

Figure 6 depicts the  $f$ -function of DES. It is composed of an initial expansion function, mapping 32 bits to 48 bits. There is no actual computation in this transformation: all the bits are permuted and some are duplicated. The second transformation is the key addition: the current round key (depending on the round number) is XORed to the expanded data. The third transformation is a substitution: 8 S-boxes are used. Each S-box transforms 6 bits into 4 bits, thus reducing the expanded data block from 48 bits back to 32 bits. The eight S-boxes are specified in the DES standard presenting them as Look-up Tables. The fourth and last transformation is a 32-bit permutation.

Figure 7: DES key schedule



As it was said in every DES round, a different round key is required. The round keys are derived from the 64-bit secret key. The process goes on iteratively, deriving the next round key from the previous one. Initially, the eight parity bits of the secret key are discarded; the remaining 56 bits are first permuted and then divided into two 28-bit long words. These two words are usually referred to as C (most significant bits) and D (least significant bits). In every round the C and D words are rotated to the left by two bit positions, except in rounds 1, 2, 9 and 16, where the two words are rotated (to the left) by a single

bit position. Such shifting scheme ensures that, after the sixteenth round of key schedule, the round key is exactly the same as it was at the beginning.

From the two words C and D, 48 bits are extracted at each round; these are the bits corresponding to the current round key. The choice of the bits to be extracted, which is performed by the block labeled PC-2 in *Figure 7*, is the same for the sixteen round keys. Since DES has the structure of a Feistel network, to perform decryption one applies the same round as for encryption, except that the sequence of round keys, which are the same as for encryption, are used in reverse order.

## Triple DES

It is possible to reuse the basic DES structure to obtain a stronger encryption scheme. A simple scheme is the so-called triple DES. The plaintext bit sequence is first encrypted using DES with secret key  $K_1$ , then it is decrypted with a second secret key  $K_2$ , and finally, it is encrypted again with a third secret key  $K_3$ . A naive analysis would indicate that the complete secret key of triple DES is a composition of three 56-bit long secret keys, for a total of 168 bits. However, triple DES is susceptible to the well known “man-in-the-middle” attack which can break any triple encryption algorithm with “only”  $2^{2n}$  trials and  $2^n$  blocks of storage as if only two simple DES keys were used (here  $n$  refers to the key size of the simple algorithm, in this case single DES), thus the actual key size is 112-bit long. Note that if  $K_1=K_2=K_3$ , the output is the same as if the plaintext had been encrypted with single DES. This feature is relevant for ensuring backward compatibility with single DES. To save memory when storing the complete secret key, it is possible to use  $K_1=K_3$ . This method, however, is less secure than triple-DES using three different keys. We refer to Schneier (1996) for details.

## DES Implementation

Hardware implementations of the Data Encryption Standard are quite straightforward. If we analyze the DES round in *Figure 6*, we see four basic transformations. First, an expansion of 32 bits is performed. Expansion in practice is a permutation: there is no actual computation and only wires and the duplication of certain bits are necessary. After the expansion, the round key is added. The required circuit is very simple since the data block and the round key are added modulo 2, which only requires XOR gates. The next transformation is the substitution via S-BOXes. The design criteria used for the eight S-boxes is not known, hence the S-boxes are usually implemented as look-up tables, via combinatorial circuits or using memory blocks. The round ends with a bit permutation; this transformation is implemented via wire-crossing as well.

One of the most commonly used design approaches to implement DES consists in instantiating a single round unit and feeding the output of the round unit

back to the input by means of two 32-bit registers: a sequential architecture. One way to reduce the critical path of the round function and, thus, speed up the implementation, is pipelining. However, pipelining is not always a viable and successful technique for speeding up an encryption algorithm. In particular, for certain modes of operation with data dependencies between consecutive plaintext blocks it is necessary to wait for the end of the encryption process of the first block before starting to encrypt the next one. This means that a pipeline implementation can only work with data blocks coming from different streams, possibly with different secret keys.

As it was illustrated in the description of the DES algorithm, the same logic used for encryption is reused for decryption; the only difference is in the key schedule. Since the round transformation is the same for encryption and decryption, only the key schedule is implemented both for encryption and decryption, and the secret key is properly processed depending on the type of operation. For instance, Sandia National Laboratories (see Wilcox et al., 1999) developed an ASIC DES chip, using a 0.6  $\mu\text{m}$  technology, running at a clock frequency of 105 MHz and reaching a throughput of 6.7 Gbits/s. In this implementation all sixteen rounds of DES were instantiated and pipelined for a total of 16 pipeline stages. Trimberger et al. (2000) presented a similar but more parallel approach. In their architecture each individual round was pipelined. The implementation was mapped onto an FPGA and the single round was divided into three stages, in such a way that 48 clock cycles are necessary to encrypt a 64-bit block. This implementation has a throughput of 12 Gbits/s. A new implementation optimized for Virtex FPGAs was recently proposed by Rouvroy et al. (2003). Due to the particular architecture of the logic block (CLB) in this FPGA, the DES round has been restructured to optimize its mapping and to allow pipelining of the round in three stages. With a clock frequency of 333MHz, this implementation has a throughput of 21 Gbits/s.

## **Advanced Encryption Standard (AES)**

Thanks to the steady increase in available computational power, it has become possible to perform exhaustive key search attacks in a reasonable amount of time by means of conventional computing equipment such as PCs or of low-cost and quickly programmable logic components such as FPGAs. A simple way to increase the security level of a cryptosystem is by increasing the length of the secret key, as it was seen with triple DES.

But this approach has an obvious drawback and bound in terms of the computational power and memory space required for processing and storing the secret key. Moreover, it is not always possible to deliberately enlarge the key size. For instance, the 168-bit triple-DES has a security lower bound of 112 bits for the length of the secret key, as discussed previously. Thus in 1997, the National Institute of Standard and Technology (NIST) decided to begin the

process of selecting a new block cipher algorithm for unclassified government documents with the explicit aim of replacing DES.

The selection process was very innovative compared to that adopted for the selection of DES many years before. The complete selection process was public and well documented. The procedure and terms for submitting an algorithm were quite simple. Each candidate algorithm was required (minimally) to be able to: encrypt an input block of 128 bits to an output block of 128 bits; and admit three different key sizes, i.e., 128 bits, 192 bits and 256 bits. Moreover, the security level of the algorithm was required to be comparable with that of the other submitted candidate algorithms. Another requirement was that the algorithm should be efficiently implementable in both software and hardware platforms. If selected, the algorithm was also required to be royalty free. A first call for algorithms was opened in January 1997; 15 candidate algorithms satisfied the initial submission requirements. In August 1999, five out of the fifteen candidate algorithms were selected as finalists. They were: MARS from IBM; RC6 from RSA; Rijndael, developed by Rijmen and Daemen from Belgium; Serpent by Biham et al.; and Twofish by Schneier et al. Three scientific conferences took place during the selection process receiving the names of AES1, AES2, and AES3 and held in 1998, 1999, and 2000, respectively. These conferences were similar to typical scientific conferences, but with the additional option (and actual invitation) to submit papers regarding the security level of a particular candidate algorithm(s) or its performance and implementation options when targeting different software and hardware platforms.

The choice of the winner algorithm was particularly hard, as all five finalist algorithms are today still considered good cryptographic algorithms and no practical attacks have been found against any of them. One particular characteristic that made Rijndael the winner algorithm is likely to have been its suitability for efficient implementation in constrained environments, such as 8-bit smart-cards and, more generally, in embedded systems. It is worth noticing that the scientific community also appreciated the selection of a European algorithm: many people in fact were convinced that a U.S. federal agency would not select a foreign algorithm for encryption purposes.

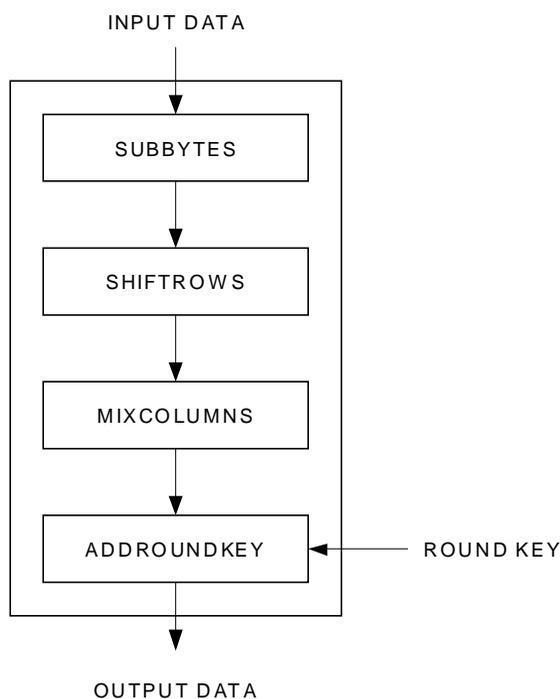
As a consequence of the public selection process, it is now possible to access all the design details of the winner algorithm. For readers interested in the details of the Rijndael design, the inventors Daemen & Rijmen have reported their work in Daemen & Rijmen (2001). The original proposed Rijndael algorithm allows the use of any combination of block and key sizes out of the following three cases: 128, 192, and 256 bits. The standard choice, which we refer to simply as AES, restricts to 128 bits the choice for input data block size, while the secret key can be chosen from 128, 192 or 256 bits. Nevertheless, the combination of 128 bits both for the data block and the secret key is the most frequently configuration used and most research work focuses on this parameter

combination only. In the remainder of this section, we shall make reference mainly to the 128-128 option.

Like all other block cipher algorithms, AES is also a composition of a basic round, which processes the input data block to encrypt, and of a key schedule, which processes the secret key to calculate the round keys. The encryption algorithm starts with an initial round, which simply adds the first round key to the input data block. Then, the round transformation is repeated 10, 12, or 14 times depending on whether the key is 128-bit, 192-bit, or 256-bit long, respectively. The basic round of the AES, shown in *Figure 8*, is composed of four internal transformations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The last round is different from the previous rounds, since it lacks the MixColumns transformation.

To easily describe the algorithm it is useful to model the data block as if it was arranged as a 2D array, which has received the name of “state matrix” or simply “state.” Each entry in the state matrix corresponds to a byte, the state matrix being squared with four rows and four columns. As it will be clear soon, the byte is the atomic information element for the AES cipher algorithm; this feature is intentional and it was adopted by the Rijndael designers to allow efficient implementations on 8-bit CPUs as well as 16-, 32- and 64-bit CPUs.

*Figure 8: The internal structure of an AES round*



The first internal transformation of the round is called SubBytes. As the name itself suggests, in this transformation the value of each byte constituting an entry of the state matrix is substituted. The substitution operation, as it usually happens in most block cipher algorithms, is non-linear. The substitution law is a mathematical function in a finite field (or Galois Field). The SubBytes transformation consists of two stages and it considers the byte as an element of the Galois field  $GF(2^8)$ . First the multiplicative inverse of the byte element is computed, with the additional constraint that if the initial byte is 0 then it is mapped to the 0 element as well. After performing the inversion, an affine transformation is applied to each bit of the byte. The following equation represents the affine transformation:

$$b'_i = b_i + b_{(i+4)\bmod 8} + b_{(i+5)\bmod 8} + b_{(i+6)\bmod 8} + b_{(i+7)\bmod 8} + c_i$$

where  $b_i$  indicates the  $i$ -th bit of the byte before the affine transformation, while  $b'_i$  indicates the same bit after the affine transformation,  $c$  is a constant with a hexadecimal value of  $0x63$ . The two stages of the transformation can be combined together and implemented as a look-up table of 256 bytes. When implemented as a look-up table, the SubBytes transformation is often called S-BOX. *Figure 9* depicts the classical table look-up for the AES.

The second internal transformation to be applied is called ShiftRows. It executes a fixed rotation of the four rows of the state matrix. The first (top) row is not touched; the second row is rotated by one byte position to the left; the third row is rotated by two byte positions to the left; and the fourth (bottom) row is rotated by three byte positions to the left. *Figure 10* summarizes in a graphical form the ShiftRows transformation, putting into evidence how the leftmost column is diffused through the state matrix.

The so-called MixColumns internal transformation is the third one, and it executes a vector-matrix multiplication of the columns of the state matrix, conceived as four vectors of four elements each, times a 4-by-4 square matrix having fixed coefficient entries interpreted as elements of  $GF(2^8)$ . Since the bytes are still considered elements of the finite field  $GF(2^8)$ , the multiplications

*Figure 9: SubBytes transformation, implemented as a S-BOX*

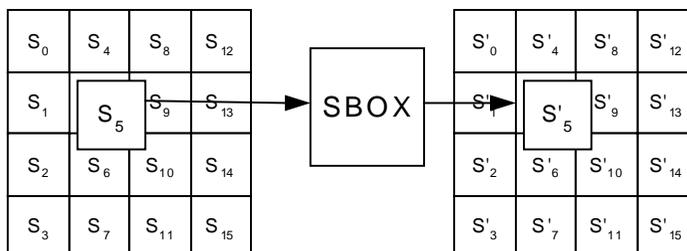
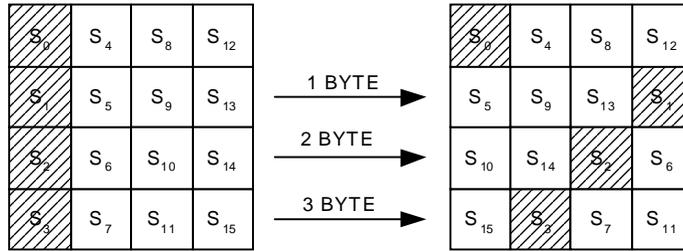


Figure 10: ShiftRows transformation and its diffusion effect



are performed in this finite field. The transformation is called MixColumns because each entry of the output state matrix in column  $i$  depends on the four entries of column  $i$  in the original input state matrix, weighted by the coefficients of the constant matrix. In other words, MixColumns spreads information across columns while ShiftRows does the same, but across rows. Of course, such a form of orthogonality is intentional, and aims at mixing and diffusing information.

Figure 11 depicts the calculation of the leftmost column as the multiplication of the column times the constant square matrix. As it can be seen, the coefficient entries of the constant square matrix have been properly chosen: their values (1, 2 and 3) make the multiplication computationally quite simple, since the multiplication times 2 is just a left shift of the byte (in the case the most significant bit of the byte is 1, reduction must be computed), while multiplication times 3 can be obtained by a left shift followed by one addition. The constant square matrix is invertible to allow decryption. Another characteristic of the constant matrix is that the four entries of the top row are cyclically repeated in the three rows down (this simplifies storing the coefficients).

The fourth and last internal transformation is called AddRoundKey. In this transformation the round key is added modulus 2 to the state matrix. This task just requires an array of XOR gates. It should be noted that the first and second transformation of each round, SubBytes and ShiftRows, can be permuted

Figure 11: MixColumns transformation, processing a single column

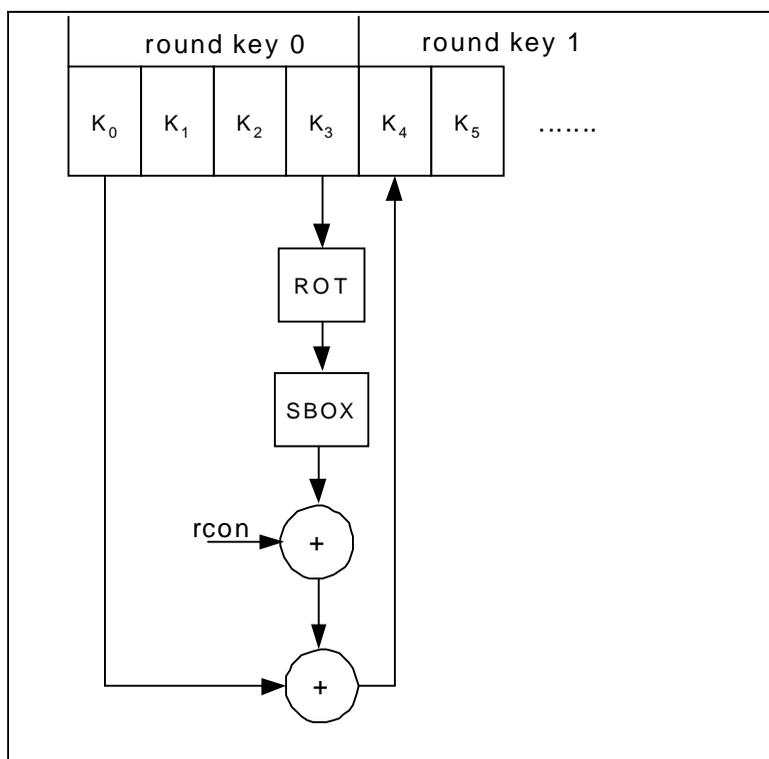
$$\begin{array}{|c|} \hline S'_0 \\ \hline S'_1 \\ \hline S'_2 \\ \hline S'_3 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} * \begin{array}{|c|} \hline S_0 \\ \hline S_1 \\ \hline S_2 \\ \hline S_3 \\ \hline \end{array}$$

without affecting the round. In fact, both work in parallel and independently on the 16 entries of the state matrix.

Similarly to any other block cipher algorithm, the secret key is iteratively processed to obtain the various round keys. In the Rijndael algorithm the key schedule is parametrized by the size of the secret key. The simplest formulation of key schedule applies to a 128-bit secret key. In the first round the secret key is added as it is, while in the subsequent rounds the current round key is derived from the previous round key in a sequential manner. Since the round key is just added to the state matrix, we can imagine the round key as if it was arranged as a matrix (the same arrangement adopted for the data block); each column of the round key is conceived as a word of 32 bits (four bytes). In total, the round key is composed of four such words.

When the current round key must be calculated, its first (leftmost) word (column) is derived from the first (leftmost) word (column) and from the fourth (rightmost) word (column) of the previous round key. In particular, the fourth (rightmost) word is first rotated, then it is processed through the S-BOX (each

*Figure 12: Key schedule*



of its four bytes is processed independently) and eventually it is added to a constant. After this transformation, the fourth (rightmost) word is bit-wise XORed to the first (leftmost) word.

The other three words are calculated in a simpler way: they are obtained as a bit-wise XOR between the word immediately on the left and the word in the same position of the previous round key.

The decryption algorithm is implemented using the four inverse internal transformations of the encryption algorithm, applied in reverse order. As seen before, the decryption round is composed of four transformations: *invSubBytes*, *invShiftRows*, *invMixColumns*, and *invAddRoundKey*. *InvSubBytes* consists of two steps: again the byte is considered as an element of the field  $GF(2^8)$ ; it is processed through the inverse of the affine transformation used in *SubBytes* and then it is inverted. *InvShiftRows* is a shift of rows, but in the inverse direction with respect to the direction of the *ShiftRows* transformation. *InvMixColumns* is a vector-matrix multiplication, but the coefficient matrix is the inverse of that used in the *MixColumn* transformation. *InvAddRoundKey* is still a bit-wise XOR operation, since the XOR gate is its self-inverse.

The round keys are used in reverse order. If the complete sequence of round keys is available at a time, it can be read starting from the end rather than from the beginning. The encryption process starts with an initial round consisting only of *AddRoundKey*, then the rounds 1 through 9 (11 or 13 if using 192-bit or 256-bit long keys) follow, each of which is the sequence of *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*. The algorithm ends with the last round, which lacks *MixColumns*, hence it is composed of *SubBytes*, *ShiftRows*, and *AddRoundKey*. The decryption algorithm starts with an initial round consisting only of *invAddRoundKey*, *invShiftRows*, and *invSubBytes*. The nominal round is a sequence of *invAddRoundKey*, *invMixColumns*, *invShiftRows*, and *invSubBytes*. The last round consists only of *invAddRoundKey*. It is also possible to exchange the order of execution of the *invShiftRows* and *invSubBytes* internal transformations (similarly to *SubBytes* and *ShiftRows* during the encryption process).

Notice that the sequence of internal transformations applied during the inverse round is different from that of the encryption round. The difference is in the order of *invMixColumns* and *AddRoundKey*. It is possible to switch the order by applying *invMixColumns* directly to the round key, because *invMixColumns* is linear. Such a feature is useful for some implementations of AES, and it will be explained in the next section. Notice also that due to the different values of the coefficients of the constant matrix in the direct and inverse *MixColumns* transformations, if the matrix multiplication is truly calculated (and not stored as a look-up table), the computations of decryption and encryption behave differently. This will be made clearer in the implementation section.

## Implementation of the AES

AES can be efficiently implemented in software on different types of CPUs, including low cost 8-bit microcontrollers, like those used in smart-cards and in embedded systems. However, for applications requiring high throughputs it is necessary to implement the encryption algorithm in hardware.

The general trend described for the implementation of DES still holds. It is possible to instantiate one round and use it iteratively, or to instantiate all the rounds, add independent functional units, and pipeline them. As it was mentioned in the case of DES, the individual round function unit can be operated in one clock cycle but if the critical path is too long, then it is possible to pipeline two or more functional units. Similarly to all block cipher algorithms, pipelining can be made difficult by the mode of operation, since many of them require finishing encrypting one block before starting with the following block.

Some of these implementations can be easily found in technical literature. Kuo & Verbauwhede (2001) had presented an ASIC AES chip that can reach 1.8 Gbps. Rijndael implemented on a Xilinx FPGA can be found on McLoone & McCanny (2001) with a throughput of 7 Gbits/s. Fischer & Drutarovský (2001) implemented the AES on an Altera FPGA with a throughput of 750 Mbits/s. Sklavos & Koufopavlou (2002) presented two ASIC implementations allowing throughputs from 259 Mbits/s to 3.6 Gbits/s. What changes in AES implementations, compared to DES ones, is that the clear and simple mathematical formulation of the AES S-BOX allows for some freedom at the time of implementation. As it has been explained previously, there exist two S-BOXes: the direct one, used for encryption; and the inverse one, for decryption.

One obvious possibility is to conceive the S-BOXes as two different tables without considering their mathematical formulation. The two S-BOXes can then be stored in memory or they can be synthesized as combinatorial networks. These two immediate solutions are usually the faster ones and the choice of one of the two depends usually on the particular technology. Alternatively, the S-BOX can be decomposed into a number of stages. For instance: the calculation of the multiplicative inverse (which is the same for both the direct and the inverse S-BOX), followed by the affine transformation. In this way, it is possible to save some silicon area or some memory cells, partially identifying the S-BOXes for encryption and decryption. But the calculation of the multiplicative inverse in a finite field is not an easy task. Algorithms based on the well-known extended Euclidean algorithm for the computation of the multiplicative inverse in a finite field exist. These algorithms are generally not considered because, although they are on average faster than exponentiation-based methods, they are hard to implement and expensive in terms of area when implemented in hardware. A first immediate solution consists in storing the inversion function alone in a look-up table. Another solution is based on the mathematical observation that the finite field  $GF(2^8)$  can be seen as a composite field  $GF((2^4)^2)$ , and thus inversion

in  $GF(2^8)$  can be reduced to inversion in  $GF(2^4)$ . The advantage of this formulation is that the look-up table storing the inverse in  $GF(2^4)$  is smaller, and the implementation by means of a memory or a combinatorial network is certainly simpler and less expensive. The adoption of composite finite fields is generally considered the smallest possible implementation of the S-BOX, in terms of silicon area. However, it is also the one requiring the longest critical path, as shown in Morioka (2002).

Two common platforms for the hardware implementation of the AES are: programmable devices, such as FPGAs, or custom (or semi-custom) logic, i.e., ASICs. In the case of FPGAs, there is usually a certain amount of memory available on the device. In those devices it is then better to store the S-BOXes in the already available memory, instead of computing it. However, some FPGAs contain enough memory to allow a different and more radical solution for the implementation of AES. As we mentioned in the previous section, it is possible to change the order of execution of the SubBytes and ShiftRows internal transformations. There is a precise reason for changing the order: organizing the SubBytes and MixColumns transformation in a single look-up table. This new look-up table is usually called T-table or `enc_table`. Since the T-table is used to substitute the two transformations, it is a mapping from a byte to four bytes. Each byte of the state should be processed via the T-table. To perform one round in one clock cycle it is necessary to use 16 T-tables. This approach can also be used for decryption. In this case, it is necessary to perform the AddRoundKey transformation after `invMixColumns`. This is possible since the two transformations are linear. However, the round keys must be processed through the `invMixColumns` transformation. In this situation we have two different expanded keys, one for encryption and one for decryption.

Combining SubBytes and MixColumns into a single table is particularly useful for decryption, since decryption requires a coefficient matrix having larger entries than those used in encryption. In fact, the larger entries make more difficult to resort to shift and addition for computing the multiplications, thus preventing the adoption the simple add-and-shift technique, which worked successfully in the implementation of encryption.

## **PUBLIC-KEY CRYPTOSYSTEM IMPLEMENTATION**

This section deals with public-key primitives and their implementation. In particular, we will describe methods used to implement DL-based, RSA-based, and ECC-based primitives. Since we cover three different primitives, it is desirable to look at these techniques in terms of a general framework or model. Such a model or framework will allow the practitioner to select techniques based on a particular application or end function. We have chosen to follow a similar

model to the one presented in the IEEE P1363 Standard Specifications for Public Key Cryptography (P1363, 2000) and augmented it with an additional layer which in our opinion is a distinct layer in the model. *Figure 13* shows the model.

*Figure 13* shows four different layers as part of the reference model. These are:

- **Layer 1 – Arithmetic Primitives:** includes the implementation of low level arithmetic operations such as addition, multiplication, and inversion in a finite field and exponentiation in finite field and RSA-type rings. This layer is from the implementation point of view one of the most important ones as its performance will influence the overall performance of the other layers.
- **Layer 2 – Cryptographic Primitives:** includes raw encryption operations, sometimes referred to as textbook encryption (i.e., textbook RSA, textbook ElGamal, etc.). The functions implemented in this layer are *not* meant to achieve security by themselves. In fact, they are not secure as they can be broken using simple attacks related to the encoding of the message for example.
- **Layer 3 – Cryptographic Schemes:** include a collection of related operations which when combined with the cryptographic primitives from Layer 2 provide complexity-theoretical security. This security is enhanced when these schemes are appropriately combined in the protocols of the next layer.
- **Layer 4 – Cryptographic Protocols:** implements a sequence of operations performed by multiple parties to achieve some security goal in a given application.

From an implementation point of view, Layers 1 and 2 correspond to low-level implementations, usually implemented with a cryptographic accelerator or software module, Layer 3 corresponds to medium level implementations usually implemented within cryptographic service libraries, and Layer 4 corresponds to high level implementations often instantiated as part of a whole application (P1363, 2000). We would like to point out that in this chapter we will only be concerned with Layers 1 and 2, in other words, with the part of the system that will determine the performance of the overall application and which, in many cases, is implemented in hardware.

## **RSA Cryptosystem**

The RSA cryptosystem, originally introduced by Rivest, Shamir & Adleman in 1978 (Rivest et al., 1978), is today the most widely used and deployed public-key cryptosystem and it is based on the difficulty of factoring integers. In this section we review the basic encryption (signature verification) and decryption (signature generation) procedures. We would like to emphasize that the RSA

Figure 13: Reference model for common public-key cryptographic techniques

Layer 4	Cryptographic Protocols
Layer 3	Cryptographic Schemes
Layer 2	Cryptographic Primitives
Layer 1	Arithmetic Primitives

primitive presented in this section is just a *primitive*. In particular, it is completely insecure if implemented as explained in this section. For an implementation of RSA to produce a secure scheme, it needs to be implemented according to existing standards such as the IEEE P1363 Standard or the RSA PKCS Standard. We remark that there is ongoing research in the cryptographic community regarding the *right* way to implement RSA and obtain a secure scheme. In fact, certain problems have been recently found with the security proofs of the OAEP padding scheme widely used for encoding the message prior to performing the encryption with the RSA primitive (Shoup, 2001).

As in any public-key primitive, each communicating party should generate a private-key and a public-key pair. Each entity generates a key pair according to Algorithm 1.

The integer  $e$  is called the encryption exponent, the integer  $d$  is referred to as the decryption exponent, and  $n$  is usually known as the *RSA modulus* or simply the *modulus*. Algorithm 2 describes the encryption and decryption process. The parameter  $\lambda = \text{lcm}(p-1, q-1)$  (where  $\text{lcm}(\ast)$  stands for least common multiplier of the numbers in parenthesis) can also be used instead of  $\phi$  in Steps 3 and 4 of Algorithm 1. Observe that  $\lambda/\phi(n)$  and thus using  $\lambda$  may result in a smaller decryption exponent which in turn might result in faster decryption. However, for random  $p$  and  $q$ , it is expected that  $\text{gcd}(p-1, q-1)$  be small and, thus,  $\lambda$  and  $\phi(n)$  should be of roughly the same size, resulting in little gain in the speed of the decryption operation (notice that in the above discussion we have made use of the fact that for any two positive integers  $a$  and  $b$  the following relation is satisfied:  $a*b = \text{gcd}(a,b) * \text{lcm}(a,b)$ ).

Step 2a of Algorithm 2 can be computed using the Chinese Remainder Theorem (CRT) which states that if integers  $n_1, n_2, \dots, n_k$  are pairwise relatively prime (i.e.,  $\text{gcd}(n_i, n_j) = 1$  with  $n = n_1 * n_2 * \dots * n_k$ ), then the system of simultaneous congruences

*Algorithm 1: Key generation for RSA public-key encryption*


---

**Output:** Public-key  $(n,e)$ , Private-key  $(d,p,q)$

1. Generate two large random and distinct primes, called them  $p$  and  $q$ , of roughly the same size.
  2. Compute  $n=p*q$  and  $\phi(n)=(p-1)(q-1)$  where  $\phi$  is the Euler phi function.
  3. Select a random integer  $e$  such that  $1 < e < \phi(n)$  and  $\gcd(e, \phi(n)) = 1$ .
  4. Compute  $d$  such that  $d*e = 1 \pmod{\phi(n)}$  via the extended Euclidean algorithm.
  5. Return party's public-key  $(n,e)$  and private-key  $(d,p,q)$ .
- 

$$X \equiv a_1 \pmod{n_1}$$

$$X \equiv a_2 \pmod{n_2}$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$X \equiv a_k \pmod{n_k}$$

has a unique solution given by  $X = \sum_{i=1}^k a_i N_i M_i \pmod{n}$  where  $N_i = n/n_i$  and  $M_i = N_i^{-1} \pmod{n_i}$ . In practice the CRT is implemented according to Algorithm 3, known as Gartner's algorithm.

Using the CRT speeds up RSA decryption (signature generation) by a factor of 3 or 4 (depending on the implementation). As a final remark, notice that it is common practice to use a small encryption exponent,  $e = 3, 17$ , or  $2^{16}+1$ , to speed up the encryption operation.

## Arithmetic Primitives

### *Techniques for Exponentiation*

Most public-key schemes are based on modular exponentiation (RSA, see Rivest et al., 1978) and Discrete Logarithm (DL) based systems (Diffie & Hellman, 1976; NIST, 2000) or point multiplication (Elliptic Curve Cryptosystems, see Miller, 1985; Koblitz, 1987; NIST, 2000). Both operations are, in their most

*Algorithm 2: RSA public-key encryption*


---

**Encryption Input:** Party A's public-key  $(n,e)$  and a message  $m$ .

**Encryption Output:** An encrypted message  $y$ .

**Decryption Input:** Party A's Private-key  $(d,p,q)$  and a message  $y$  encrypted with Party A's Public-key  $(n,e)$ .

**Decryption Output:** Message  $m$ .

1. Encryption procedure
    - a. Obtain A's authentic public key  $(n,e)$ .
    - b. Represent the message  $m$  as an integer in the interval  $[0,n-1]$ .
    - c. Compute  $y=m^e \bmod n$ .
    - d. Send encrypted message  $y$  to A.
  2. Decryption procedure
    - a. Recover the message  $m$  as  $m=y^d \bmod n$ .
- 

basic forms, performed via the binary method for exponentiation or one of its variants (Gordon, 1998). Algorithm 4 shows the binary method for exponentiation also known as the square-and-multiply algorithm.

Algorithm 4 takes  $t$  squarings (we do not count the operation  $1 \times 1$  as a squaring) and on average  $t/2$  multiplications by  $g$ . The most common generalization of Algorithm 4 is called the  $k$ -ary method for exponentiation and it is depicted in Algorithm 5. In this context  $k$  is known as the window size. The basic idea in Algorithm 5 is to process more than one bit at a time. This is done at the additional cost of pre-computation.

Algorithm 5 requires  $l$  squaring and  $2^k-3$  multiplications for the pre-computation steps (Steps 2 through 4). In addition, for a  $(t+1)$ -bit long exponent  $e$ , the number of words of size  $k$  bits in  $e$  is  $s+1 = \lceil (t+1)/k \rceil$ , thus it follows that the number of squarings (notice that we do not count  $1^b$  as performing squarings) is just equal to  $sk$  whereas, on average, the number of multiplications is equal to  $s(2^k-1)/2^k$ , (see Menezes et al., 1997). A further improvement over Algorithm 5 is the so-called sliding-window exponentiation algorithm which reduces the number of precomputations by half but requires more complicated loop logic to perform the exponentiation. The sliding-window algorithm for exponentiation is depicted in Algorithm 6.

*Algorithm 3: Gartner's algorithm for the CRT*


---

**Input:** Positive integer  $n = \prod_{i=1}^k n_i > 1$ , with  $\gcd(n_i, n_j) = 1$  for all  $i \neq j$  and a modular

representation  $v(X) = (x_1, x_2, \dots, x_k)$  where  $x \equiv x_i \pmod{n_i}$

**Output:** the integer  $X$

1. **for**  $i = 2$  to  $k$  **do**
  2.      $C_i \leftarrow 1$
  3.     **for**  $j = 1$  to  $i-1$  **do**
  4.          $u \leftarrow n_j^{-1} \pmod{n_i}$
  5.          $C_i \leftarrow u C_i \pmod{n_i}$
  6.     **end for**
  7. **end for**
  8.  $u \leftarrow v_1$
  9.  $X \leftarrow u$
  10. **for**  $i = 2$  to  $k$  **do**
  11.      $u \leftarrow (v_i - X) C_i \pmod{n_i}$
  12.      $X \leftarrow X + u \prod_{j=1}^{i-1} n_j$
  13. **end for**
  14. Return( $X$ )
- 

Using Algorithm 6 has an effect similar to using the  $k$ -ary method for exponentiation (Algorithm 5) with a window of size  $k+1$  but at no extra cost in precomputation. Thus, the total number of windows processed and, consequently, the number of general multiplications behaves as  $(t+1)/(k+1)$  as opposed to  $(t+1)/k$  as in Algorithm 5 (Blake et al., 1999).

We notice that the exponentiation algorithms presented so far have been written in a multiplicative group notation. It is trivial to change the algorithms to additive notation by changing multiplications to additions and exponentiations to multiplications if one is dealing with a group where the operation is addition, as

*Algorithm 4: Left-to-right binary exponentiation algorithm*


---

**Input:**  $g \in G$  and a positive integer  $e = (e_t e_{t-1} e_{t-2} \dots e_2 e_1 e_0)_2$

**Output:**  $g^e$

1.  $A \leftarrow I$
  2. **for**  $i = t$  to  $0$  **do**
  3.      $A \leftarrow A \times A$
  4.     **if**  $e_i = 1$  **then**
  5.          $A \leftarrow A \times g$
  6.     **end if**
  7. **end for**
  8. Return ( $A$ )
- 

it is in the elliptic curve case. We also point out that the methods presented in this section are but a few of the many available. For example, one could use exponent recoding techniques in groups where inversion is not expensive, such as in elliptic curves, and combine such recoding techniques with the algorithms previously presented. We refer the reader to Gordon (1998) and Menezes et al. (1997) for good treatments of exponentiation techniques for cryptographic purposes.

Finally, notice that the atomic operation in any exponentiation method is either modular multiplication, in the case of RSA and DL-based systems, or point addition, in the case of ECC, which is, in turn, performed through a combination of multiplications and additions on the field of definition of the elliptic curve. Thus, the first part of the following section is mainly concerned with how to perform modular multiplication efficiently in hardware devices. The second part of the next section treats the case of elliptic curves and state of the art processor architectures, which have been proposed to perform EC operations.

*Modular Multiplication*

The problem of modular multiplication and, more specifically, the problem of modular reduction has been extensively studied since it is a fundamental building block of any cryptosystem. Among the algorithms that have been proposed we find:

*Algorithm 5: Left-to-right k-ary exponentiation*


---

**Input:**  $g \in G$  and a positive integer  $e = (e_s e_{s-1} e_{s-2} \dots e_2 e_1 e_0)_b$

where  $b = 2^k$  for some  $k \geq 1$ .

**Output:**  $g^e$

*Precomputation*

1.  $T[0] \leftarrow 1$
2. **for**  $i = 1$  to  $2^k - 1$  **do**
3.  $T[i] \leftarrow T[i-1] \times g$  (Note:  $T[i] = g^i$ )
4. **end for**

*Main computation*

5.  $A \leftarrow 1$
  6. **for**  $i = s$  to  $0$  **do**
  7.  $A \leftarrow A^b$
  8.  $A \leftarrow A \times T[e_i]$
  9. **end for**
  10. Return ( $A$ )
- 

- *Sedlak's Modular Reduction*  
Originally introduced in Sedlak (1987), this algorithm is used by Siemens in the SLE44C200 and SLE44CR80S microprocessors to perform modular reduction (Naccache & Raihi, 1996). Sedlak notices that the algorithm improves the reduction complexity by an average factor of  $1/3$  when compared to the basic bit-by-bit reduction.
- *Barret's Modular Reduction*  
It was originally introduced in Barret (1986) in the context of implementing RSA on a DSP processor. Suppose that you want to compute  $X \equiv R \pmod{M}$  for some modulus  $M$ . Then, we can re-write  $X$  as  $X = Q * M + R$  with  $0 \leq R < M$ , which is a well known identity from the division algorithm. See Definition 2.82 in Menezes et al. (1997). Thus

$$R = X \pmod{M} = X - Q * M \tag{1}$$

*Algorithm 6: Sliding-window exponentiation*


---

**Input:**  $g \in G$  and a positive integer  $e = (e_i e_{i-1} e_{i-2} \dots e_2 e_1 e_0)_2$

**Output:**  $g^e$

*Precomputation*

1.  $T[1] \leftarrow 1$
2.  $T[2] \leftarrow g^2$
3. **for**  $i = 1$  to  $2^{k-1} - 1$  **do**
4.      $T[2i+1] \leftarrow T[2i-1] \times T[2]$
5. **end for**

*Main computation*

6.  $A \leftarrow 1; i \leftarrow t$
  7. **while**  $i \geq 0$  **do**
  1.     **if**  $e_i = 0$  **then**
  2.          $A \leftarrow A^2$
  3.          $i \leftarrow i - 1$
  4.     **else**
  5.         Find the longest bitstring  $e_i e_{i-1} \dots e_s$  such that  $i - s + 1 \leq k$  and  $e_s = 1$
  6.         do the following:  $A \leftarrow A^{2^{i-s+1}} \times T[(e_i e_{i-1} \dots e_s)_2]$
  7.          $i \leftarrow s - 1$
  8.     **end if**
  9. **end while**
  10. Return ( $A$ )
- 

Barret's basic idea is that one can write  $Q$  in (1) as:

$$Q = \left\lfloor \frac{X}{M} \right\rfloor = \left\lfloor \left( \frac{X}{b^{n-1}} \right) \left( \frac{b^{2n}}{M} \right) \left( \frac{1}{b^{n+1}} \right) \right\rfloor \quad (2)$$

In particular,  $Q$  can be approximated by

$$\hat{Q} = Q_3 = \left\lfloor \left\lfloor \left( \frac{X}{b^{n-l}} \right) \left( \frac{b^{2n}}{M} \right) \left( \frac{1}{b^{n+l}} \right) \right\rfloor \right\rfloor$$

Notice that the quantity  $\mu = b^{2n}/M$  can be precomputed when performing many modular reductions with the same modulus, as is the case in cryptographic algorithms. Having precomputed  $\mu$ , the expensive computations in the algorithm are only divisions by powers of  $b$ , which are simply performed by right-shifts, and modular reduction modulo  $b^i$ , which is equivalent to truncation. We refer to Section 14.3.3 in Menezes et al. (1997) for further discussion of implementation issues regarding Barret reduction, and to Dhem (1994; 1998) for improvements over the original algorithm.

- *Brickell's Modular Reduction*  
Originally introduced in Brickell (1982), is dependent on the utilization of carry-delayed adders (Norris & Simmons, 1981) and combines a sign estimation technique [See for example Koç & Hung (1991) and Omura's modular reduction (Omura, 1990)].
- *Quisquater's Modular Reduction*  
Quisquater's algorithm, originally presented in Quisquater (1990;1992) can be thought of as an improved version of Barret's reduction algorithm. Benaloh & Dai (1995) and Walter (1991) have proposed similar methods. In addition, the method is used in the Phillips smart-card chips P83C852 and P83C855, which use the CORSAIR crypto-coprocessor (De Waleffe & Quisquater, 1990; Naccache & Raihi, 1996) and the P83C858 chip, which uses the FAME crypto-coprocessor (Ferreira et al., 1996). Quisquater's algorithm, as presented in (De Waleffe & Quisquater, 1990), is a combination of the interleaved multiplication reduction method (basically, combine a normal multiprecision algorithm with modular reduction, making use of the distributivity property of the modular operation) and a method that makes easier and more accurate the estimation of the quotient  $Q$  in (1).
- *Montgomery Modular Multiplication*  
The Montgomery algorithm, originally introduced in Montgomery (1985), is a technique that allows efficient implementation of the modular multiplication without explicitly carrying out the modular reduction step. We discuss it in detail, as it is the most widely used algorithm for modular multiplication in the literature.

### *Montgomery Modular Multiplication*

The idea behind Montgomery's algorithm is to transform the integers in  $M$ -residues and compute the multiplication with these  $M$ -residues. At the end, one

transforms back to the normal representation. As with Quisquater & Barret's method, this approach is only beneficial if we compute a series of multiplications in the transform domain (e.g., modular exponentiation). The Montgomery reduction algorithm can be stated as follows: Given integers  $M$  and  $R$  with  $R > M$  and  $\gcd(M, R) = 1$  with  $M' \equiv -M^{-1} \pmod R$  and  $T$  an integer such that  $0 \leq T < M^*R$ , if  $Q \equiv TM' \pmod R$ , then  $Z = (T + QM)/R$  is an integer and, furthermore,  $Z \equiv T^*R^{-1} \pmod M$ . Notice that our description is just the reduction step involved in a modular multiplication. The multiplication step can be carried out via multi-precision multiplication, see for example Chapter 14 in Menezes et al. (1997) and Koc et al. (1996). As with other algorithms, one can interleave multiplication and reduction steps. The result is shown in Algorithm 7.

In Algorithm 7, it is assumed that  $X$  and  $Y$  are already in the Montgomery domain (i.e., they are  $M$ -residues) and, in particular,  $X = X'R \pmod M$  and  $Y = Y'R \pmod M$  for integers  $0 \leq X', Y' < M$ . In practice  $R$  is a multiple of the word size of the processor and a power of two. This means that  $M$ , the modulus, has to be odd (because of the restriction  $\gcd(M, R) = 1$ ) but this does not represent a problem as  $M$  is a prime or the product of two primes (RSA) in most practical cryptographic applications. In addition, choosing  $R$  a power of 2, simplifies the computation of  $Q$  and  $Z$  as they become simply truncation (modular reduction by  $R$ ) and right shifting (division by  $R$ ). Notice that  $M' \equiv -M^{-1} \pmod R$ . In Dussé & Kaliski (1990) it is shown that if

$$M = \sum_{i=0}^{n-1} m_i b^i$$

for some radix  $b$ , typically a power of two, and  $R = b^n$ , then  $M'$  can be substituted by  $m_0' = -M^{-1} \pmod b$ . In Eldridge & Walter (1993), the authors simplify the combinatorial logic needed to implement Montgomery reduction.

The idea in Eldridge & Walter (1993) is to shift  $Y$  by two digits (i.e., multiply  $Y$  by  $b^2$ ) and thus, make  $q_i$  in Step 4 of Algorithm 7 independent of  $Y$ . Notice that one could have multiplied  $Y$  by  $b$  instead of  $b^2$  and have also obtained a  $q_i$  independent of  $Y$ . However, by multiplying  $Y$  by  $b^2$ , one gets  $q_i$  to be dependent only on the partial product  $Z$  and on the lowest two digits of the multiple of  $M$  (i.e.  $q_i * M$ ). The price of such a modification is two extra iterations of the for-loop for which the digits of  $X$  are zero. The architecture proposed by Eldridge & Walter (1993) is only considered for the case  $b=2$  and estimated to be twice as fast as previous modular multiplication architectures at the time of publication.

### *Higher Radix Montgomery Modular Multiplication*

In Vuillemin et al. (1996) and Shand & Vuillemin (1993) modular exponentiation architectures are implemented on an array of 16 Xilinx 3090 FPGAs. Their design uses several speed-up methods (Shand & Vuillemin, 1993) including the

*Algorithm 7: Montgomery multiplication*


---

**Input:**  $X = \sum_{i=0}^{n-1} x_i b^i, Y = \sum_{i=0}^{n-1} y_i b^i, M = \sum_{i=0}^{n-1} m_i b^i$ , with  $0 \leq X, Y < M, b > 1, m' = -m_0^{-1} \bmod b$ ,

$R = b^n, \gcd(b, M) = 1$

**Output:**  $Z = X * Y * R^{-1} \bmod M$

1.  $Z \leftarrow 0$       { where  $Z = \sum_{i=0}^n z_i b^i$  }
  2. **for**  $i=0$  to  $n-1$  **do**
  3.       $q_i \leftarrow (z_0 + x_i * y_0) m' \bmod b$
  4.       $Z \leftarrow (Z + x_i * Y + q_i * M) / b$
  5. **end for**
  6. **if**  $Z \geq M$  **then**
  7.       $Z \leftarrow Z - M$
  8. **end if**
  9. Return ( $Z$ )
- 

CRT, asynchronous carry completion adder, and a windowing exponentiation method. Some of the improvements are:

- Avoid having to perform a subtraction after every modular product of the exponentiation algorithm by letting all intermediate results have *two extra bits* of precision. Shand & Vuillemin (1993) also show that even allowing for the two extra bits of precision, one can always manage to work with intermediate results no larger than  $n$ -digits if  $M < b^n/4$  and  $X, Y \leq 2M$ .
- A second improvement is the use of a radix  $b=2^2$ , which permits for a trivial computation of the quotient  $q_i$  in Step 4 of Algorithm 4 and for the use of Booth recoded multiplications (this doubles the multipliers performance compared to  $b=2$  at an approximate 1.5 increase in hardware complexity). Higher radices, which would offer better performance, were dismissed since they involve too great of a hardware cost and the computation of the quotient digits is no longer trivial.
- They rewrite Montgomery's Algorithm in a similar way to Eldridge & Walter (1993) to allow for pipeline execution, basically getting rid off of the

$q$ , dependency on the least significant digit of the partial product  $Z$ . The cost for  $d$  levels of pipelining is  $d$  extra bits of precision and  $d$  more cycles in the computation of the final product.

The result of all these speedup methods is an RSA secret decryption rate of over 600 Kbits/sec for a 512-bit modulus and of 165 Kbits/sec for a 1024-bit modulus, using the CRT. While the previous results make full use of the reconfigurability of the FPGAs (reconfigurability is required to recombine the result of the CRT computations), they derive a single gate-array specification whose size is estimated under 100K gates and speed over 1 Mbit/sec for RSA 512-bit keys.

The main obstacle to the use of higher radices in the Montgomery algorithm is that of the quotient determination. In Orup (1995), the author presents a method that avoids quotient determination altogether and thus makes higher-radix Montgomery practical. The price to pay for avoiding quotient determination is more precision and, at most, one more iteration in the main loop of the algorithm. The final improvement in Orup (1995) is the use of quotient pipelining. Unlike Shand & Vuillemin (1993), Orup (1995) is able to achieve quotient pipelining only at the cost of extra loop iterations and no extra precision.

As an example, Orup (1995) considers an architecture with 3 pipeline stages and a radix  $b=2^8$ . The author estimates the critical path of the architecture to be no more than 5ns assuming 1995 CMOS technology. It is also assumed the use of a redundant representation for the intermediate values of the Montgomery multiplier. However, the outputs have to be converted back to non-redundant representation using a carry-ripple adder with an asynchronous carry completion detection circuit as proposed in Shand & Vuillemin (1993). With these techniques, the author estimates the time of one 512-bit modular multiplication at 415 nsec. Using the left-to-right binary method for exponentiation, one 512-bit exponentiation would take 319  $\mu$ sec, which corresponds to a 1.6 Mbit/sec throughput. If instead, one uses the right-to-left binary exponentiation algorithm, one can perform multiplications and squarings in parallel as shown in Orup & Kornerup (1991), thus achieving a factor of two speedup, i.e., more than 2.4 Mbit/sec throughput. This is four times faster than the implementation of Shand & Vuillemin (1993), which at the time was the fastest. Furthermore, if the modulus is composite, as in the RSA case, and its prime factorization is known, it is possible to obtain a factor of four speedup through the use of the CRT as in Shand & Vuillemin (1993).

In Blum & Paar (1999), the authors implemented a version of Montgomery's algorithm optimized for a radix two hardware implementation. Blum & Paar (2001) extend Orup (1995) to reconfigurable hardware, a systolic array architecture as presented in Kornerup (1994), and following Orup (1995) high radix hardware implementations of modular exponentiation. There had been a number

of proposals for systolic array architectures for modular arithmetic but, to our knowledge, Blum & Paar (1999; 2001) were the first implementations that have been reported. For the exact design and technical details we refer the reader to Blum (1999) and Blum & Paar (1999; 2001). Here, however, we summarize their results. As target devices, Blum & Paar (1999;2001) used the Xilinx XC40250XV, speed grade -09, 8464 CLBs, for the larger designs (>5000 CLBs), and the XC40150XV, speed grade -08, 5184 CLBs, for the smaller designs. *Table 3* shows results for a full-length modular exponentiation, i.e., an exponentiation where base, exponent, and modulus have all the same bit length. We notice that Blum & Paar (1999; 2001) both use the right-to-left method for exponentiation.

*Table 4* shows Blum & Paar (1999;2001) RSA encryption results. The encryption time is calculated for the Fermat prime  $F_n = 2^{16} + 1$  exponent (Knuth, 1981), requiring  $2 \cdot 19 \cdot (n+2)$  clock cycles for the radix 2 design (Blum & Paar, 1999), and  $2 \cdot 19 \cdot (n+8)$  clock cycles if the radix 16 design is used, where the modulus has  $n-2$  bits.

For decryption, Blum & Paar (2001) apply the CRT. They either decrypt  $m$  bits with an  $m/2$  bit architecture serially, or with two  $m/2$  bit architectures in parallel. The first approach uses only half as many resources, the latter is almost twice as fast. A little time is lost here because of the slower delay specifications of the larger devices.

## Elliptic Curve Cryptosystems over Finite Fields

In this section, we first provide a brief introduction to elliptic curve point addition and doubling. Additional information can be found in Miller (1985), Koblitz (1987) and Blake et al. (1999).

### *Elliptic Curves over GF(p)*

An elliptic curve  $E$  over  $GF(p)$  for  $p > 3$  is the set of solutions  $P=(x,y)$  which satisfy the following Weierstrass equation:

$$E: \quad y^2 = x^3 + Ax + B \text{ mod } p$$

where  $A, B \in GF(p)$  and  $4A^3 + 27B^2 \neq 0 \text{ mod } p$ , together with the point at infinity  $O$ . It is well known that the points on an elliptic curve form a group under an addition operation which is defined as follows. Let  $P=(x_1, y_1) \in E$ ; then  $-P=(x_1, -y_1)$ .  $P + O = O + P = P$  for all  $P \in E$ . If  $Q=(x_2, y_2) \in E$  and  $Q \neq -P$ , then  $P + Q = (x_3, y_3)$ , where

$$x_3 = \lambda^2 - x_1 - x_2 \tag{3}$$

$$y_3 = \lambda (x_1 - x_3) - y_1 \tag{4}$$

Table 3: CLB usage and execution time for a full modular exponentiation

Radix	512 bit		768 bit		1024 bit	
	C	T	C	T	C	T
	(CLBs)	(msec)	(CLBs)	(msec)	(CLBs)	(msec)
2 (Blum & Paar, 1999)	2555	9.38	3745	22.71	4865	40.05
16 (Blum & Paar, 2001)	3413	2.93	5071	6.25	6633	11.95

and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + A}{2y_1} & \text{if } P = Q \end{cases} \quad (5)$$

This coordinate representation is known as affine representation. In affine representation a point addition takes 1 inversion and 3 multiplications whereas a

Table 4: Application to RSA: Encryption

Radix	512 bit		768 bit	
	C	T	C	T
	(CLBs)	(msec)	(CLBs)	(msec)
2 (Blum & Paar, 1999)	2555	0.35	4865	0.75
16 (Blum & Paar, 2001)	3413	0.11	6633	0.22

Table 5: Application to RSA: Decryption

Radix	512 bit		512 bit		1024 bit		1024 bit	
	2 x 256 serial		2 x 256 parallel		2 x 512 serial		2 x 512 parallel	
	C	T	C	T	C	T	C	T
	(CLBs)	(msec)	(CLBs)	(msec)	(CLBs)	(msec)	(CLBs)	(msec)
2 (Blum & Paar, 1999)	1307	4.69	2416	2.37	2555	18.78	5110	10.18
16 (Blum & Paar, 2001)	1818	1.62	3636	0.79	3413	5.87	6826	3.10

point doubling takes 1 inversion and 4 multiplications, where we have counted squarings as multiplications and additions and subtractions have been ignored. However, in many applications it is more convenient to represent the points  $P$  and  $Q$  in projective coordinates. This is advantageous when inversion is more computationally expensive than multiplication in the finite field  $GF(p)$ . Thus, algorithms for projective coordinates trade inversions in the point addition and the point doubling operations for a larger number of multiplications and a single inversion at the end of the algorithm. This inversion can be computed via exponentiation using the fact that  $A^{-1} \equiv A^{p-2} \pmod{p}$ , for prime modulus  $p$  (Fermat's Little Theorem). In projective coordinates, a point  $P=(x,y)$  is represented as  $P=(X,Y,Z)$  where  $X=x$ ,  $Y=y$ , and  $Z=1$ . To convert from projective coordinates back to the affine ones, we use the following relations:

$$x = \frac{X}{Z^2}, \quad y = \frac{Y}{Z^3}$$

Finally, one can obtain expressions equivalent to (3), (4), and (5) for doubling and addition operations in projective coordinates. Algorithms 8 and 9 summarize point addition and doubling in projective coordinates. One can achieve a point doubling in the general case with 10 finite field multiplications which can be reduced to 8 multiplications when  $A=-3$ . This follows from the fact that in this case Step 1 of Algorithm 9 can be computed as  $3(X_1-Z_1^2)*(X_1-Z_1^2)$  reducing the number of multiplications for this step from 4 to 2 (Chudnovsky & Chudnovsky, 1987; Blake et al., 1999). Similarly, addition requires 16 field multiplications in the general case and only 11 when one of the points being added is constant, i.e.,

the point is given in affine coordinates and thus  $Z_i = 1$  throughout the computation. This case occurs often in cryptographic applications (P1363, 2000). Note that in this context, the complexity of adding or doubling a point on an elliptic curve is usually given by the number of field multiplications and inversions (if affine coordinates are being used), field additions are relatively cheap operations compared to multiplications or inversions as well as multiplications by a small constant such as 2, 3, or 8.

## ELLIPTIC CURVES OVER $GF(2^n)$

An elliptic curve  $E$  over  $GF(2^n)$  is the set of solutions  $P=(x,y)$  which satisfy the following Weierstrass equation:

$$E: \quad y^2 + xy = x^3 + Ax^2 + B$$

where  $A, B \in GF(2^n)$  and  $B \neq 0$ , together with the point at infinity  $\mathbf{O}$ . As before let  $P=(x_1, y_1) \in E$ ; then  $-P=(x_1, y_1 + x_1)$ .  $P + \mathbf{O} = \mathbf{O} + P = P$  for all  $P \in E$ . If  $Q=(x_2, y_2) \in E$  and  $Q \neq -P$ , then  $P + Q = (x_3, y_3)$  can be computed in affine representation as follows. If  $P \neq Q$

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + A \\ y_3 &= \lambda(x_1 + x_3) + y_1 + x_3 \end{aligned}$$

and if  $P = Q$  then

$$x_3 = \lambda^2 + \lambda + Ay_3 = \lambda(x_1 + x_3) + y_1 + x_3$$

where

$$\lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1} & \text{if } P \neq Q \\ \frac{y_1}{x_1} + x_1 & \text{if } P = Q \end{cases}$$

These formulae implies that one needs 1 inversion, 2 multiplications, and 1 squaring over  $GF(2^n)$  to perform a point addition or a point doubling. Notice that over  $GF(2^n)$ , squaring is a much cheaper operation than general multiplication.

As in the  $GF(p)$  case, a point  $P=(x,y)$  in projective coordinates is represented as  $P=(X,Y,Z)$  where  $X=x$ ,  $Y=y$ , and  $Z=1$ . To convert from projective coordinates back to the affine ones, we use the same relations that we used for

*Algorithm 8: Point addition in projective coordinates over  $GF(p)$ ,  $p > 3$*

---

**Input:**  $P = (X_1, Y_1, Z_1)$ ,  $Q = (X_2, Y_2, Z_2)$  with  $P, Q \neq \mathbf{O}$  and  $P \neq \pm Q$

**Output:**  $P + Q = (X_3, Y_3, Z_3)$

1.  $S_1 \leftarrow X_1 * Z_2^2$  2 multiplications
  2.  $S_2 \leftarrow X_2 * Z_1^2$  2 multiplications
  3.  $S_3 \leftarrow S_1 - S_2$
  4.  $S_4 \leftarrow Y_1 * Z_2^3$  2 multiplications
  5.  $S_5 \leftarrow Y_2 * Z_1^3$  2 multiplications
  6.  $S_6 \leftarrow S_4 - S_5$
  7.  $S_7 \leftarrow S_1 + S_2$
  8.  $S_8 \leftarrow S_4 + S_5$
  9.  $Z_3 \leftarrow Z_1 * Z_2 * S_3$  2 multiplications
  10.  $X_3 \leftarrow S_6^2 - S_7 * S_3^2$  3 multiplications
  11.  $S_9 \leftarrow S_7 * S_3^2 - 2X_3$
  12.  $Y_3 \leftarrow (S_9 * S_6 - S_8 * S_3^3) / 2$  3 multiplications
  13. Return( $X_3, Y_3, Z_3$ )
- 

the  $GF(p)$  case. Algorithms 10 and 11 summarize point addition and doubling in projective coordinates for curves defined over fields of characteristic 2.

One can achieve a point doubling in the general case with 5 finite field multiplications and 5 finite field squarings. Similarly, addition requires 15 field multiplications and 5 squarings in the general case and only 14 multiplications and 4 squarings if  $A=0$ . If we consider a fixed point, i.e.,  $Z = 1$  for one of the points, then general addition requires 11 multiplications and 4 squarings which is further reduced to 10 multiplications and 3 squarings when  $A=0$ .

### *FPGA Processor Architecture for ECC over $GF(p)$*

Orlando & Paar (2001) propose a new elliptic curve processor (ECP) architecture for the computation of point multiplication for curves defined over fields  $GF(p)$ . Point multiplication is defined as the product  $kP$ , where  $k$  is an integer,  $P$  is a point on the elliptic curve, and by multiplication we mean that  $P$  is

*Algorithm 9: Point doubling in projective coordinates over  $\text{GF}(p)$ ,  $p > 3$*

---

**Input:**  $P = (X_1, Y_1, Z_1)$  with  $P \neq \mathbf{O}$

**Output:**  $2P = (X_3, Y_3, Z_3)$

- |  |                   |
|--|-------------------|
| 1. $S_1 \leftarrow 3X_1^2 + A * Z_1^4$   | 4 multiplications |
| 2. $Z_3 \leftarrow 2Y_1 * Z_1$           | 1 multiplication  |
| 3. $S_2 \leftarrow 4X_1 * Y_1^2$         | 2 multiplications |
| 4. $X_3 \leftarrow S_1^2 - 2S_2$         | 1 multiplication  |
| 5. $S_3 \leftarrow Y_1^4$                | 1 multiplication  |
| 6. $Y_3 \leftarrow S_1(S_2 - X_3) - S_3$ | 1 multiplication  |
| 7. Return( $X_3, Y_3, Z_3$ )             |                   |
- 

added to itself  $k$  times. We emphasize that there is no multiplication operation on the elliptic curve, only additions and doublings of points  $P \in E$ . The ECP is best suited for the computation of point multiplications using projective coordinates. Inversions are computed via Fermat's Little Theorem and multiplication via Montgomery reduction.

The ECP has a scalable architecture in terms of area and speed specially suited for memory-rich hardware platforms such as field programmable gate arrays (FPGAs). This processor uses a new type of high-radix Montgomery multiplier that relies on the precomputation of frequently used values and on the use of multiple processing engines. The ECP consists of three main components. These components are the main controller (MC), the arithmetic unit controller (AUC), and the arithmetic unit (AU). The MC is the ECP's main controller. It orchestrates the computation of  $kP$  and interacts with the host system. The AUC controls the AU. It orchestrates the computation of point additions/subtractions, point doubles, and coordinate conversions. It also guides the AU in the computation of field inversions. Both the MC and the AUC execute their respective operations concurrently and they have the capability of executing one instruction per clock cycle. The AU incorporates a multiplier, an adder (or adders), and a register file, all of which can operate in parallel on different data. The AU's large register set supports algorithms that rely on precomputations.

As with systems based on RSA or the DL problem in finite fields, in ECC-based systems, multiplication is also the most critical operation in the computation of elliptic curves point multiplications. The elliptic curve processor (ECP) introduced in Orlando & Paar (2001) develops a new multiplier architecture that

draws from Orup (1995), and Frecking & Parhi (1999) an approach for high radix multiplication, from Shand & Vuillemin (1993) and Orup (1995) the ability to delay quotient resolution, and from Blum (1999) the use of precomputation. In particular, this work extends the concept of precomputation. The resulting multiplier architecture is a high-radix precomputation-based modular multiplier, which supports positive and negative operands, Booth recoding and pre-computation.

Orlando & Paar (2001) developed a prototype that implemented the double-and-add algorithm using the projective coordinates algorithms for point addition and point double operations on a Xilinx's XCV1000E-8-BG680 (Virtex E) FPGA. This prototype was programmed to support the field  $GF(2^{192}-2^{64}-1)$ , which is one of the fields specified in FIPS 186-2 (2000). To verify the ECP's architectural scalability to larger fields, a modular multiplier for fields as large as

*Algorithm 10: Point addition using projective coordinates for curves over fields  $GF(2^n)$*

---

**Input:**  $P = (X_1, Y_1, Z_1)$ ,  $Q = (X_2, Y_2, Z_2)$  with  $P, Q \neq \mathbf{O}$  and  $P \neq \pm Q$

**Output:**  $P + Q = (X_3, Y_3, Z_3)$

- |  |                       |
|--|-----------------------|
| 1. $S_1 \leftarrow X_1 * Z_2^2$                    | 1 mult. + 1 squaring  |
| 2. $S_2 \leftarrow X_2 * Z_1^2$                    | 1 mult. + 1 squaring  |
| 3. $S_3 \leftarrow S_1 + S_2$                      |                       |
| 4. $S_4 \leftarrow Y_1 * Z_2^3$                    | 2 mult.               |
| 5. $S_5 \leftarrow Y_2 * Z_1^3$                    | 2 mult.               |
| 6. $S_6 \leftarrow S_4 + S_5$                      |                       |
| 7. $S_7 \leftarrow S_3 * Z_1$                      | 1 mult.               |
| 8. $S_8 \leftarrow S_6 * X_2 + S_7 * Y_2$          | 2 mult.               |
| 9. $Z_3 \leftarrow Z_2 * S_7$                      | 1 mult.               |
| 10. $S_9 \leftarrow S_6 + Z_3$                     |                       |
| 11. $X_3 \leftarrow A * Z_3^2 + S_6 * S_9 + S_3^3$ | 3 mult. + 2 squarings |
| 12. $Y_3 \leftarrow S_9 * X_3 + S_8 * S_7^2$       | 2 mult. + 1 squaring  |
| 13. Return( $X_3, Y_3, Z_3$ )                      |                       |
-

*Algorithm 11: Point doubling using projective coordinates for curves over fields  $GF(2^n)$*

---

**Input:**  $P = (X_1, Y_1, Z_1)$  with  $P \neq \mathbf{O}$  and  $S_2 = B^{2^{n-2}}$

**Output:**  $2P = (X_3, Y_3, Z_3)$

- |   |                       |
|---|-----------------------|
| 1. $Z_3 \leftarrow X_1 * Z_1^2$             | 1 mult. + 1 squaring  |
| 2. $X_3 \leftarrow >(X_1 + S_2 * Z_1^2)^4$  | 1 mult. + 2 squarings |
| 3. $S_1 \leftarrow Z_3 + X_1^2 + Y_1 Z_1$   | 1 mult. + 1 squaring  |
| 4. $Y_3 \leftarrow X_1^4 * Z_3 + S_1 * X_3$ | 2 mult. + 1 squaring  |
| 5. Return( $X_3, Y_3, Z_3$ )                |                       |
- 

$GF(2^{521}-1)$  was also prototyped. The ECP prototype for  $GF(2^{192}-2^{64}-1)$  used 11,416 LUTs, 5,735 Flip-Flops, and 35 BlockRAMS. The frequency of operation of the prototype was 40 MHz for 192 bit operands and 37.3 MHz for the 521-bit multiplier. The authors in Orlando & Paar (2001) point out that, assuming that the ECP is coded in a form that extracts 100% throughput from its multiplier, it will compute a point multiplication for an arbitrary point on a curve defined over  $GF(2^{192}-2^{64}-1)$  in approximately 3 msec.

## ATTACKS AGAINST CRYPTOSYSTEMS

In this section, a review of the possible attacks that can be performed against a cryptosystem is carried out. The main purpose of an attack is to get access to the secret key or to be able to read a message that was not intended for the attacker. Once the secret key is obtained, it becomes possible to decrypt a message, or to sign a message as being originated from the secret key's legal owner. When the attack is successful, i.e., the attacker has been able to obtain the secret key, it is said that the cryptosystem has been "cracked" or "broken".

All cryptosystems are, in principle, prone to two different families of attacks: theoretical or algorithmic attacks and implementation attacks. The former ones try to exploit the mathematical properties of the algorithm, while the latter ones try to exploit the defects, flaws, and faults of the device implementing the algorithm, if not to actively cause some fault in order to exploit them. Implementation attacks can also take advantage of an error in the implementation of the algorithm (not necessarily of the device where the algorithm runs). The science that studies the methods used to break various cryptographic

algorithms, and hence assess their robustness against such attacks, is named cryptanalysis.

A theoretical (algorithmic) attack models the cryptosystem as a black box generating some output data. However, frequently some internal details are known, such as the algorithm itself (after Kerchoff's principle), and possibly some implementation parameters, such as the key size. In all cases, the aim of a theoretical attack is to extract the secret key by means of a careful analysis of the output data. The simplest theoretical attack is brute-force key search: all possible secret keys are used to attempt decryption, one of them necessarily succeeds. For the key lengths used in modern ciphers this type of attack is not feasible at present.

It must be noted that there is a principle, known as Kerckhoff's principle, which states that the attacker knows *all* the details of the encryption function except for the secret key. Thus, the security level of the algorithm should depend only on the secrecy of the key. This principle is widely accepted for commercial systems, where the standard network protocols require that all the parties involved in the communication must be able to participate, and it is also an unavoidable consequence of the mass volume production of cryptographic devices. In addition, the idea of the algorithm being widely known favors the interoperability of devices from different manufacturers. However, in the military community, it is often the case that the algorithm is kept secret, simply because it makes cryptanalysis harder to perform. If everybody knows the cryptographic algorithm then, at least in principle, anybody could perform a brute force attack. This is a first, obvious, but fundamental indicator for choosing the size of the secret key during the design of a cryptographic algorithm long enough to withstand brute force attacks.

In the case of theoretical attacks, recovering the secret key from a single block of output data (when encrypted with a well designed cipher), or from few blocks, is a hard task; most attacks require a sufficiently large set of output data blocks. If it is possible to obtain some plaintext-ciphertext pairs, the attack is referred to as a known plaintext attack. Both theoretical and implementation attacks can take advantage of these additional pieces of information. For instance, it is common practice to exchange encrypted files attached to e-mail messages. In most cases the format of such files is easily predictable (documents, executables, picture files, sound files, etc.), since they are standardized (*de iure* rather than *de facto*) and most standard file formats start with some common and possibly invariant header. Therefore, it is reasonable to assume that the attacker is able to obtain some plaintext – ciphertext pairs. In some cases it is possible to force the cryptosystem to encrypt some properly chosen sequence of plaintext blocks. In this case the attacker has the possibility to exploit some particular property of the cryptosystem. This type of attack has received the name of chosen plaintext attack.

Attacks that try to take advantage of defects or flaws in the implementation of the cryptosystem are more interesting from an engineering point of view. Such attacks are generally divided into two sub-families: invasive and non-invasive attacks. Invasive attacks try to read out some important piece of data from the system and, generally, they alter the system physically. In some cases, the functionality of the system is definitely compromised by the attack. For instance, microchips implementing a cryptographic module are de-packaged and probes are positioned to read out the data transferred between the memory and the CPU. Many of these attacks are derived from well-known techniques available in reverse engineering studies; the interested readers should see Anderson (2001).

To design and implement hardware cryptographic modules resistant against these types of attacks some common design rules have been developed. Two frameworks have been standardized and nowadays are well known. One is known under the name of Common Criteria (Common Criteria), and it is sponsored by the governments of France, U.K., Canada, Germany, the Netherlands, and the U.S., among others. The other has been developed by NIST for cryptographic modules used by the U.S. government. In the Common Criteria (Common Criteria) website can be found the specification of the framework, while the NIST specification can be found in NIST (2001a).

The two frameworks do not specify only the requirements of the product but also methodologies necessary during its design and fabrication. Common Criteria requirements are more comprehensive than FIPS140, and NIST defers to Common Criteria for some aspects, such as the security level of the operating system. Manufacturers of hardware cryptographic modules must submit their products to independent laboratories, which certify the security level reached by the module. The two standards specify a list of accredited laboratories. Both standards define Critical Security Parameters as that data that should be inaccessible from outside the system without appropriate authorization, such as secret keys, seeds for the random number generators, PINs, passwords, etc.

A minimal security level should guarantee that these parameters are set to zero (or zeroized) in case the perimeter of the system is trespassed and/or the system is tampered with in some way. This means that the system should always be aware of its security perimeter and of its integrity state. More interesting and recent are those types of attacks that do not require entering the perimeter of the system, i.e., the so-called non-invasive attacks.

These attacks work on what is called side-channel information. During the execution of a generic encryption algorithm, the system frequently releases (and in some cases, unavoidably, because the effect depends on some very fundamental physical law) information about the computation that is being performed. This information could be highly correlated to the value of critical parameters such as the secret key. If such information can be recorded, then it becomes possible to set up an attack. One of the systems most affected by side-channel attacks

are smart cards. Smart cards are very small, inexpensive, and the attacker could easily get into possession (illegally but also legally) of the complete system and operate it at will.

One of the best-known attacks related to side-channel analysis is the so-called power analysis. Every electronic system requires a power supply to work. The basis of the attack is to notice that the power consumption is not constant in time, rather it is dependent on the operation currently performed by the system. The reader can easily imagine a system performing an elliptic curve scalar multiplication via the simple double-and-add algorithm. It is known that if the  $i$ -th bit of the secret key is equal to zero, then a point doubling operation is performed, while a point doubling operation followed by a point addition operation is executed when the bit is equal to 1. If, additionally, it is practically feasible to measure the power consumption profile of the device by means of an ordinary oscilloscope or some other kind of measurement equipment, it becomes possible to understand, from the power trace, whether a single doubling operation or a doubling operation followed by an addition operation has been performed, thus recovering the secret key. This kind of attack has revolutionized the design of encryption algorithms in smart-card environments in recent years. In its simplest form, it has received the name of Simple Power Analysis (SPA).

More complex, but also more powerful, is Differential Power Analysis (DPA). In this case statistical analysis is applied to the recorded power traces, with the aim of inferring some bits of the secret key, (see Kocher et al., 1999). These attacks are particularly oriented to smart cards, since a smart card usually has a simple interface to the external world, just a few pins such as power ( $V_{cc}$ ), Ground, Clock, reset and I/O, which makes tracing power consumption very easy. Thermal analysis could be used as well, since most of the power is quickly dissipated by the Joule effect.

After the publication of this attack, several countermeasures were proposed. The simplest proposed countermeasure introduces some dummy operations. The attack to the double-and-add scalar multiplication is based on the asymmetry of the implementation. We can degrade the performance of the algorithm and perform a dummy point addition operation when the key bit is equal to zero; in this way the algorithm is (more) balanced. This solution is known as double-and-add-always, see Coron (1999). At the price of a higher power consumption, the device becomes more resistant against power attacks.

Most of the research effort has been dedicated to finding different formulations for the encryption algorithms, in order to make the computation independent of the secret-key bits. One way to achieve this independence is through randomization: each time the algorithm is executed a different random input is used. In this way side-channel information is dependent on an unknown variable, that is changing continuously and thus, uncorrelated to the secret data. In the case of the elliptic curve cryptosystem, there have been many proposals

suggesting the adoption of different coordinate systems, allowing balanced computation requirements for the point addition and point doubling operations, in order to make them indistinguishable in their power consumption profiles.

Moore et al. (2001) proposed to use asynchronous design and dual-rail logic: one bit of information is coded in two complementary wires, offering a balanced power consumption. In some smart cards, equipped with an 8-bit microprocessor, it has been noted that the power trace is independent of the computation, but it is instead related to the data transferred from the memory to the CPU. From the power consumption profile it is then possible to understand the relative Hamming distance of two words read in succession from the memory. If these two words are part of a secret (round) key, the attacker is able to reduce the key search space.

A new type of side-channel attack has been developed in recent years: instead of recording the power consumption, the electromagnetic emission (Electromagnetic Analysis) is profiled. In theory this attack is very powerful, since profiling could be performed even at a long distance. As with power analysis it is possible to perform a simple or differential attack based on EM radiation.

Another possibility is to change the value of some data during the computation of a cryptographic operation, thus producing an error during the execution of the algorithm, which eventually leads to discovering the secret key. This new technique is known as fault injection (Boneh et al., 2001). It is based on the observation that glitches injected into the power supply can force the output of a logic gate to a particular value. To understand what this attack might do, imagine being able to set to zero the value of the data block currently processed by AES, just before the key addition during the last round. What we obtain as output is actually the last round key, which is enough to recover the original secret key, and hence to break the implementation.

This short review of the implementation attacks should convince the reader that even the best cryptographic algorithms can be weak if the implementation is poor or the system performing the encryption is itself insecure.

## CONCLUSION

We notice that the implementation of cryptographic systems presents several requirements and challenges. First, the performance of the algorithms is often crucial. One needs encryption algorithms to run at the communication link transmission rates or at fast enough rates that customers do not become dissatisfied. Second, in order to achieve such satisfactory performance, it is imperative to have a good understanding and knowledge of: (i) the encryption algorithms, (ii) the algorithms underlying their implementation (not necessarily the encryption algorithm but algorithms which are used to implement them, such

as algorithms for finite field arithmetic), and (iii) the hardware platform. Finally, the security engineer also has to be aware of the latest trends in the design of encryption schemes, as well as the latest attacks. This chapter makes emphasis on the implementation aspects. We provide several implementation approaches and compare them, thus allowing the reader to have a wide range of options for different applications. In other words, some applications might require the fastest possible implementation of the AES, without regard to power consumption and/or area, whereas others might want to be optimized for the last two parameters as long as an acceptable performance level is still achievable. Finally, we also hint at possible attacks on implementations and some solutions presented in the literature.

## REFERENCES

- Altera Corporation. (2000). *Nios Soft Core Embedded Processor*. Altera Corporation. Available from the World Wide Web: <http://www.altera.com/products/devices/nios/nio-index.html>.
- Altera Corporation. (2002a). *Excalibur Device Overview*. Altera Corporation. Retrieved from the World Wide Web: <http://www.altera.com/products/devices/arm/arm-index.html>.
- Altera Corporation. (2002b). *Stratix FPGA Family*. Altera Corporation. Retrieved from the World Wide Web: <http://www.altera.com/products/devices/dev-index.jsp>.
- Barrett, P. (1986). Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In A. M. Odlyzko (ed.), *Advances in Cryptology - CRYPTO '86* (Vol. LNCS 263) (pp. 311-323). Berlin: Springer-Verlag.
- Benaloh, J., & Dai, W. (1995). Fast modular reduction. Rump session of CRYPTO '95.
- Blake, I., Seroussi, G., & Smart, N. (1999). *Elliptic curves in cryptography*. Cambridge University Press, London Mathematical Society Lecture Notes Series 265.
- Blum, T. (1999). Modular exponentiation on reconfigurable hardware (M.S. Thesis). ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA.
- Blum, T., & Paar, C. (1999). Montgomery modular multiplication on reconfigurable hardware. In *Proceedings of the 14th IEEE Symposium on Computer Arithmetic (ARITH-14)* (pp. 70-77).
- Blum, T., & Paar, C. (2001, July). High radix Montgomery modular exponentiation on reconfigurable hardware. *IEEE Transactions on Computers*, 50 (7), 759-764.

- Bondalapati, K., & Prasanna, V. (2002). Reconfigurable computing systems. In *Proceedings of the IEEE*.
- Boneh, D., DeMillo, R., & Lipton, R. (2001). On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 14 (2), 101-119.
- Brickell, E. F. (1982). A fast modular multiplication algorithm with applications to two key cryptography. In D. Chaum, R. L. Rivest & A. T. Sherman (eds.), *Advances in Cryptology - CRYPTO '82* (pp. 51-60). New York, USA: Plenum Publishing.
- Chameleon Systems Inc. (n.d.). Retrieved from the World Wide Web: <http://www.chameleonsystems.com/>.
- Chudnovsky, D., & Chudnovsky, G. (1986). Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, 7 (4), 385-434.
- Common Criteria. (n.d.). Retrieved from the World Wide Web: <http://www.commoncriteria.org>.
- Coron, J. (1999). Resistance against differential power analysis for elliptic curve cryptosystems. In C.K. Koc & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES '99* (Vol. LNCS 1717) (pp. 292-302). Berlin: Springer-Verlag.
- Daemen, J., & Rijmen, V. (2001). *The design of Rijndael, AES - The Advanced Encryption Standard*. Berlin: Springer-Verlag.
- De Waleffe, D., & Quisquater, J.J. (1990). CORSAIR: A smart card for public key cryptosystems. In A. J. Menezes & S. A. Vanstone (eds.), *Advances in Cryptology - CRYPTO '90*, (Vol. LNCS 537) (pp. 502-514). Berlin: Springer-Verlag.
- Dhem, J.F. (1994, July 18). Modified version of the Barret modular multiplication algorithm. *UCL Technical Report* (CG-1994/1). Universite catholique de Louvain.
- Dhem, J.F. (1998). *Design of an efficient public-key cryptographic library for RISC-based smartcards* (Ph.D. Thesis). UCL Universite catholique de Louvain, Louvain-la-Neuve, Belgium.
- Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. In *IEEE Transactions on Information Theory IT-22* (pp. 644-654).
- Dusse, S. R., & Kaliski, B. S. (1990). A cryptographic library for the Motorola DSP56000. In I. B. Damgard (ed.), *Advances in Cryptology - EUROCRYPT '90* (Vol. LNCS 473) (pp. 230-244). Berlin: Springer-Verlag.
- Eldridge, S. E., & Walter, C. D. (1993, July). Hardware implementation of Montgomery's modular multiplication algorithm. *IEEE Transactions on Computers*, 42 (6), 693-699.
- Ferreira, R., & Malzahn, R., & Marissen, P., & Quisquater, J.J., & Wille, T. (1996). FAME: A 3rd generation coprocessor for optimising public key

- cryptosystems in smart card applications. In P. H. Hartel, P. Paradinas & J. J. Quisquater (eds.), *Proceedings of CARDIS 1996, Smart Card Research and Advanced Applications* (pp. 59-72). Amsterdam: Stichting Mathematisch Centrum, CWI.
- Fischer, V., & Drutarovský, M. (2001). Two Methods of Rijndael Implementation in Reconfigurable Hardware. In C.K. Koc, D. Naccache & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001* (Vol. LNCS 2162) (pp. 77-92). Berlin: Springer-Verlag.
- Frecking, W., & Parhi, K. K. (1999). A unified method for iterative computation of modular multiplications and reduction operations. In *International Conference on Computer Design ICCD '99* (pp. 80-87).
- Gordon, D. M. (1998). A survey of fast exponentiation methods. *Journal of Algorithms*, 27, 129-146.
- Hauser, J., & Wawrzynek, J. (1997). Garp: A MIPS processor with reconfigurable coprocessor. In K. Pocek & J. Arnold (eds.), *IEEE Symposium on FPGAs for Custom Computing Machines* (pp.12-21).
- Knuth, D. E. (1981). *The art of computer programming*. Volume 2: Seminumerical Algorithms (2nd ed.). Reading, MA: Addison-Wesley.
- Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48, 203-209.
- Koc, C.K., & Hung, C. Y. (1991). Bit-level systolic arrays for modular multiplication. *Journal of VLSI Signal Processing*, 3 (3), 215-223.
- Koc, C.K., Acar, T., & Kaliski, Jr., B. (1996, June). Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, pp. 26-33.
- Kocher, P., Jaffe, J., & Jun, B. (1999). Differential power analysis. In M. Wiener (ed.), *Advances in Cryptology - CRYPTO '99* (Vol. LNCS 1666) (pp. 388-397). Berlin: Springer-Verlag.
- Kornerup, P. (1994, August). A systolic, linear-array multiplier for a class of right-shift algorithms. *IEEE Transactions on Computers*, 43 (8), 892-898.
- Kuo, H., & Verbauwhede, I. (2001). Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES Rijndael algorithm. In C.K. Koc, D. Naccache & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001* (Vol. LNCS 2162) (pp. 156-163). Berlin: Springer-Verlag.
- Lai, X., Massey, Y., & Murphy, S. (1991). Markov ciphers and differential cryptanalysis. In D. W. Davies (ed.), *Advances in Cryptology - EUROCRYPT '91* (Vol. LNCS 547). Berlin: Springer-Verlag.
- Lidl, R., & Niederreiter, H. (1997). *Finite Fields* (2nd ed). Encyclopedia of Mathematics (Vol. 20). London: Cambridge University Press.
- Massey, J. L., & Lai, X. (1992). *Device for converting a digital block and the use thereof*. European Patent, Patent Number 482154.

- McLoone, M., & McCanny, J. (2001). High performance single-chip FPGA Rijndael Algorithm. In C.K. Koc, D. Naccache & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001* (Vol. LNCS 2162) (pp. 65-76). Berlin: Springer-Verlag.
- Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1997). Handbook of applied cryptography. Boca Raton, FL: CRC Press.
- Miller, V. (1986). Uses of elliptic curves in cryptography. In H. C. Williams (ed.), *Advances in Cryptology - CRYPTO '85* (Vol. LNCS 218) (pp. 417-426). Berlin: Springer-Verlag.
- Montgomery, P. L. (1985, April). Modular multiplication without trial division. *Mathematics of Computation*, 44 (170), 519-521.
- Moore, S., Anderson, R., Cunningham, P., Mullins, R., & Taylor, G. (2002). Improving smart card security using self-timed circuits. In *IEEE Proceedings of the Eighth International Symposium on Advanced Research in Asynchronous Circuits and Systems*.
- Morioka, S., & Satoh, A. (2002). An optimized S-Box circuit architecture for low power AES design. In C.K. Koc, D. Naccache & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2002* (Vol. LNCS 2162) (pp. 172-186). Berlin: Springer-Verlag.
- Naccache, D., & M'Raihi, D. (1996). Cryptographic smart cards. *IEEE Micro*, 16 (3), 14-24.
- NIST. (1999). *NIST FIPS PUB 46-3, Data Encryption Standard (DES)*. U.S. Department of Commerce/National Institute of Standards and Technology. Retrieved from the World Wide Web: <http://csrc.nist.gov/encryption/tkencryption.html>.
- NIST. (2000). *NIST FIPS PUB 186-2, Digital Signature Standard (DSS)*. U.S. Department of Commerce/National Institute of Standard and Technology. Retrieved from the World Wide Web: <http://csrc.nist.gov/encryption>.
- NIST. (2001a.) *NIST FIPS PUB 140-2, Security requirements for cryptographic modules*. Retrieved from the World Wide Web: <http://csrc.nist.gov/encryption>.
- NIST. (2001b.) *NIST FIPS PUB 197, Specification for the Advanced Encryption Standard (AES)*. U.S. Department of Commerce/National Institute of Standard and Technology. Retrieved from the World Wide Web: <http://csrc.nist.gov/encryption/aes>.
- NIST. (2001c.) *Recommendation for block cipher modes of operation. Methods and techniques* (NIST Special Publication 800-38A). Retrieved from the World Wide Web: <http://csrc.nist.gov/encryption>.
- Norris, M. J., & Simmons, G. J. (1981). Algorithms for high-speed modular arithmetic. *Congressus Numeratum*, 31, 153-163.

- Omura, J. K. (1990). A public key cell design for smart card chips. In *International Symposium on Information Theory and its Applications*(pp. 983-985).
- Orlando, G., & Paar, C. (2001). A scalable GF(p) elliptic curve processor architecture for programmable hardware. In C.K. Koc, D. Naccache, and C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001* (Vol. LNCS 2162) (pp. 348-363). Berlin: Springer-Verlag.
- Orup, H. (1995). Simplifying quotient determination in high-radix modular multiplication. In *Proceedings of the 12th IEEE Symposium on Computer Arithmetic (ARITH 12)* (pp. 193-199).
- Orup, H., & Kornerup, P. (1991). A high-radix hardware algorithm for calculating the exponential  $M^E$  Modulo N. In P. Kornerup & D. W. Matula (eds.), *Proceedings of the 10th IEEE Symposium on Computer Arithmetic (ARITH 10)* (pp. 51-56).
- P1363 2000. (n.d.). *IEEE P1363-2000: IEEE Standard Specifications for Public Key Cryptography*. Retrieved from the World Wide Web: <http://standards.ieee.org/catalog/olis/busarch.html>.
- Quisquater, J.J. (1990). *Fast modular exponentiation without division*. Rump session of EUROCRYPT '90.
- Quisquater, J. J. (1992). *Encoding system according to the so-called RSA method, by means of a microcontroller and arrangement implementing this system*. United States Patent, Patent Number 5166978.
- Rivest, R. L., Shamir, A., & Adleman, L. (1978, February). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2), 120-126.
- Rouvroy, G., Standaert, F., Quisquater, J., & Legat, J. (2003). Efficient uses of FPGAs for implementations of DES and its experimental linear cryptanalysis. *IEEE Transaction on Computers*, 52 (4), 473- 482.
- Schneier, B. (1996). *Applied cryptography* (2nd ed). New York: John Wiley & Sons Inc.
- Sedlak, H. (1987). The rsa cryptography processor. In D. Chaum & W. L. Price (eds.), *Advances in Cryptology - EUROCRYPT '87* (Vol. LNCS 304) (pp. 95-105). Berlin: Springer-Verlag.
- Shand, M., & Vuillemin, J. (1993). Fast implementations of RSA cryptography. In E. Swartzlander, Jr., M. J. Irwin & G. Jullien (eds.), *Proceedings of the 11th IEEE Symposium on Computer Arithmetic (ARITH-11)* (pp. 252-259).
- Shoup, V. (2001). OAEP reconsidered. In J. Kilian (ed.), *Advances in Cryptology – CRYPTO 2001* (Vol. LNCS 2139) (pp. 239-259). Berlin: Springer-Verlag.

- Sklavos, N., & Koufopavlou, O. (2002). Architectures and VLSI implementations of the AES-Proposal Rijndael. *IEEE Transaction on Computers*, 51 (12), 1454- 1459.
- Trimberger, S., Pang, R. & Singh, A. (2000). A 12 Gbps DES Encryptor/ Decryptor Core in an FPGA. In C.K. Koc & C. Paar (Eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES'00* (Vol. LNCS 1965, pp. 156-163). Berlin: Springer-Verlag.
- Triscend Corporation. (n.d.). Retrieved from the World Wide Web: <http://www.triscend.com/>.
- Vuillemin, J. E., & Bertin, P., & Roncin, D., & Shand, M., & Touati, H. H., & Boucard, P. (1996, March). Programmable active memories: Reconfigurable systems come of age. *IEEE Transactions on VLSI Systems*, 4 (1), 56-69.
- Walter, C. D. (1991). Faster modular multiplication by operand scaling. In J. Feigenbaum (ed.), *Advances in Cryptology – CRYPTO '91* (Vol. LNCS 576) (pp. 313-323). Berlin: Springer-Verlag.
- Wilcox, D. C., & Pierson, L., Robertson, P., Witzke, E., & Gass, K. (1999). A DES ASIC suitable for network encryption at 10 Gbps and beyond. In C.K. Koc & C. Paar (eds.), *Workshop on Cryptographic Hardware and Embedded Systems - CHES '99* (Vol. LNCS 1717) (pp. 37-48). Berlin: Springer-Verlag.
- Wong, S., Vassiliadis, S., & Cotofana, S. (2002). Future directions of (programmable and reconfigurable) embedded processors. In *Embedded Processor Design Challenges, Workshop on Systems, Architectures, Modeling, and Simulation - SAMOS 2002*.
- Xilinx Inc. (2002). *VirtexTM-II Platform FPGA Data Sheet*. Xilinx Inc. Retrieved from the World Wide Web: <http://www.xilinx.com/partinfo/databook.htm>.
- Xilinx Inc. (2003). *Virtex-II ProTM Platform FPGAs: Introduction and overview*. Xilinx Inc. Version 2.4. Retrieved from the World Wide Web: <http://direct.xilinx.com/bvdocs/publications/ds083.pdf>.

## Chapter II

# Digital Certificates and Public-Key Infrastructures

Diana Berbecaru, Politecnico di Torino, Italy

Corrado Derenale, Politecnico di Torino, Italy

Antonio Lioy, Politecnico di Torino, Italy

### ABSTRACT

*The technical solutions and organizational procedures used to manage certificates are collectively named Public Key Infrastructure (PKI). The overall goal of a PKI is to provide support for usage of public-key certificates within – and also outside – its constituency. To this aim, several functions are needed, such as user registration, key generation, certificate revocation and many others. It is the aim of this paper to describe issues related to digital certificates and PKIs, both from the technical and management viewpoint.*

### INTRODUCTION

In 1976, Diffie & Hellman introduced the concept of public-key (or asymmetric) cryptography in their paper “New Directions in Cryptography”. This kind of cryptography uses a pair of mathematically related keys to perform the encryption and decryption operations. One key is named the “private key” and is known only to its owner, while the other key is named “public key” and must be publicly known. Public-key cryptography is a quantum leap in the field of security because it offers a better solution to several old problems: data and party authentication, privacy without a shared secret and key distribution.

The full range of benefits of public-key cryptography can be obtained only when there is assurance about the entity associated to the public key being used; that is the entity that controls the corresponding private key. To this purpose, members of small groups of communicating parties can meet face-to-face and directly exchange their public keys, for example on labelled floppy disks, and then ensure that these keys are securely stored on each user's local system. This is usually known as manual key distribution (Ford & Baum, 1997), but it is seldom used outside small closed groups because it is highly impractical. Another approach is to aggregate the keys into a so-called "public file" (i.e., the list of keys and associated entities), managed by a trusted entity that makes it publicly available. This solution has its own problems too: the file is insecure and can be manipulated, the trusted entity is a single point of failure (the whole system fails if it gets compromised or access to it is denied) and the whole approach doesn't scale well.

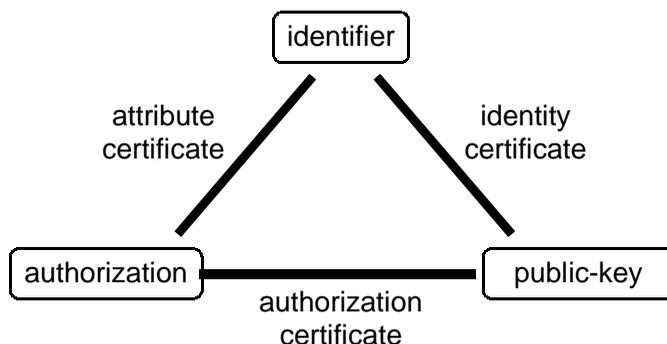
A better solution would be to bind the public key to the controlling entity on an individual basis and protect this binding with some cryptographic measure. To this aim, Loren Kohnfelder, in his MIT Bachelor thesis (1978), proposed to use a signed data structure named *public-key certificate* (PKC). Webster's Dictionary defines a certificate as a "document containing a certified statement, especially as to the truth of something" (Webster's New Collegiate Dictionary, 1980).

The Kohnfelder's approach leaves open the issue about the signer of the certificate. This could be a user Alice that would digitally sign Bob's public key along with Bob's name and other accessory information. The result would be Bob's certificate, which could convince anyone who trusts Alice that Bob's public key really belongs to him. This is the approach taken by some certification systems, such as PGP (Garfinkel, 1995) in which a certificate is signed by all the users that vouch for the data contained in the certificate. However this approach is unpractical, relies on personal judgement and doesn't scale well. Thus, usually the role of certificate signer is taken by a specialized entity named *Certification Authority* (CA) that handles the certificates on behalf of its constituency and takes some sort of liability for having performed the necessary trust and security checks. When no privacy issue exists, the certificates are published in appropriate repositories (such as a web server or a LDAP directory) to make them widely available. Since a certificate is digitally signed information, it is intrinsically secure and no other specific security measure is needed when it is stored or downloaded from the repository.

When a third party accepts a certificate as part of a security measure to protect a data exchange with a PKI user, he plays the role of a *relying party* (RP) because he relies on the issuer to have provided accurate data inside the certificate.

In the remainder of the chapter we deal with certificate formats, standards, and certificate management principles.

Figure 1: Certificates



## CERTIFICATE FORMATS

Digital certificates can be divided into three classes based on the data they bind together (*Figure 1*): identity, attribute and authorization certificates.

An *identity certificate* binds together a public-key and some information that uniquely identifies the certificate's *subject*, that is the person, device, or entity that controls the corresponding private key. A certificate of this type is issued by a CA. For its similarity to the identity documents used in the human world, this is the most used type of certificate for a variety of applications and it is usually simply referred to as public-key certificate.

An *attribute certificate* binds an identity to an authorization, title or role by a digital signature. That signature is produced by a trusted third party, named *Attribute Authority* (AA), which has the power and takes the liability of assessing the attribute. Attribute certificates are finding increasing use in access control and electronic signatures.

An *authorization certificate* binds an authorization, title or role directly to a public key rather than to an identity. The concept here is that the public key can speak for its key-holder; that is the entity that controls the corresponding private key. Thus, a public key is by itself an identifier. This kind of certificate has been proposed to shorten the authorization process: when the AA coincides with the consumer of the attribute certificate (i.e., the resource controller), then the controller can directly issue an authorization certificate. Authorization certificates are rarely used and are applied mostly in access control within closed groups.

### The X.509 Standard

The X.509 ISO/IEC/ITU recommendation (ITU-T Recommendation, 2000) is the most widely accepted standard for the format of digital certificates. This

is a general and very flexible standard. As such, to achieve application interoperability often a profiling operation is needed. For example, the IETF PKIX working group has defined a X.509 certificate profile for use with Internet applications (Housley et al., 1999; 2002) while Visa, MasterCard and other players adopted X.509 as the basis of the certificate format used by the SET standard for electronic commerce (SET, 2003).

Originally, X.509 was conceived as the authentication framework for the X.500 directory service, but later has been applied mostly out of its original scope. Four versions of this standard exist: version 1 (1988) is the foundation but quickly showed some severe limitations, not solved by version 2 (1993), that is a minor one. Widespread use of X.509 took place with version 3 (1996) that addresses the limitations of the previous versions:

- since each subject can hold several certificates (for the same or different public keys), a mechanism to unambiguously distinguish between them is needed;
- as X.500 is rarely used in practice, other ways to identify the subject and issuer are needed;
- for commercial applications, it is important to know the certification policy used to issue the certificate because it is related to the legal liability;
- mechanisms should allow the definition of mutual trust relationships between different certification authorities.

X.509v3 uses optional fields (named *extensions*) to carry these and other additional data. Finally, ITU-T Recommendation (2000) further extends the aim of this standard by introducing attribute certificates, in addition to the identity certificates supported since version 1.

The basic fields of an X.509v3 certificate are shown in *Table 1* and their meaning is explained.

The *serial number* field contains the unique identifier of the certificate within all the certificates created by the issuer. Since it is a unique identifier, the “issuer name and serial number” pair always uniquely identifies a certificate and hence a public key.

The *signature algorithm* field identifies the algorithm and optional parameters used by the issuer when signing the certificate. This is very important to avoid a class of cryptographic attacks based on detaching a signature created by one algorithm and claiming that it was done with a different algorithm.

The *issuer name* field specifies the X.500 distinguished name (DN) of the CA that issued the certificate. This field must always be non-empty.

The *validity period* field specifies the start and the expiration date of the validity of the certificate. The issuer backs the certificate only within this period.

The *subject name* field specifies the X.500 distinguished name (DN) of the entity holding the private key corresponding to the public key identified in the

certificate. If this field is left empty (because the subject doesn't have a directory entry) then the identity information must be carried in the *Subject Alternative Name* extension field that must be marked as critical. This permits one to identify the subject by the name or names in the extension because it binds a key to an application-level identifier (such as an e-mail address or an URI, when support for secure email or Web is desired).

The *subject public key information* field contains the value of the public key owned by the subject, and the identifier of the algorithm with which the public key is to be used.

The *CA Signature* field contains the actual value of the digital signature of the CA and the identifier of the signature algorithm by which it was generated. Quite obviously, this identifier must be the same as the one in the CA signature algorithm field.

Table 1: Sample X.509 public-key certificate

<i>Field name</i>		<i>Example</i>
Version		version 3
Serial number		12345
CA signature algorithm identifier		algorithm sha1WithRsaEncryption (OID 1.2.840.113549.1.1.5)
Issuer		CN=Politecnico di Torino Certification Authority, O=Politecnico di Torino, C=IT
Validity Period	Not Before	Wed Dec 19 18:00:00 2001 (011219170000Z)
	Not After	Wed Dec 31 10:00:00 2003 (031231090000Z)
Subject		CN=Alice Twokeys, OU=Dipartimento di Automatica e Informatica, O=Politecnico di Torino, C=IT
Subject Public Key Information		Algorithm RSA (OID 1.2.840.113549.1.1.1)
Certificate Extensions	AuthorityKeyIdentifier	F40C 6D6D 9E5F 4C62 5639 81E0 DEF9 8F2F 37A0 84C5
	Subject Key Identifier	9E85 84F9 4CAF A6D3 8A33 CB43 ED16 D2AE 8516 6BB8
	Key Usage	digitalSignature nonRepudiation keyEncipherment dataEncipherment
	Subject alternative names	RFC822: alice.twokeys@polito.it
...		...
CA Signature		BDD6A3EC D7C1C411 146FBD7E DF25FF 4AB7871E 023F18BB 4DA23262 90822E EFBE8370 58A2248D A6839F23 FA1AA1 E9CC4BC1 144CF894 E391D5B0 B79D86 48F2EE52 7550A7E1 41CDFEDA DCB394 340A3B66 5C835BE7 00B4B97A F1D3D3 4E12E0A0 8983B194 D0101B95 5A94B3 57FD72CA A0E96C01 66BB1CD4 F9C9F7

Since certificate extensions are more complex, their treatment is deferred to the homonymous section.

The X.509 standard distinguishes among end-entity (EE) certificates and CA certificates. An *end-entity certificate* is issued by a CA to a key-holder that can use it for several applications (e.g., to digitally sign documents) but not to sign certificates. On the contrary, a *CA certificate* is issued by a CA to another CA; in this case, the private key corresponding to the certified public key is mainly used to issue other certificates.

CA certificates can be self-signed or cross certificates. A *cross certificate* is issued by a CA to another CA to empower it to sign certificates for its constituency. On the contrary, a *self-signed certificate* is issued by a CA to itself (i.e., the subject and the issuer of the certificate are the same entity). This is usually done to distribute the certificate of a so-called *root CA* or *trusted CA*; that is a CA that is directly trusted by the end user. For example, self-signed certificates are installed inside the most common web browsers to let the user automatically trust some commercial root CAs. If this is positive or negative from a security viewpoint is still a topic for debate.

## Certificate Status Information (via CRL)

X.509 certificates have a validity period with an expiration date. However a certificate can become invalid before its natural expiration for various reasons. For instance, the secret key may have been lost or compromised, or the owner of the certificate may have changed his affiliation. In general, any time any data present in a certificate changes, then the certificate must be revoked. This is a task of the certificate issuer that must also let this event be publicly known by making available fresh certificate status information about the revoked certificate. The X.509 standard does not mandate any specific way in which an authority should maintain the revocation information, but suggests the Certificate Revocation List (CRL) method. The CRL is a signed object, which contains the serial numbers of the revoked certificates. A sample CRL is shown in *Table 2*. If a relying party application encounters a revoked certificate, then it should be configured to perform different actions depending on the revocation reason. For example, a certificate that was revoked because the private key was compromised has more serious implications compared with the case when the certificate was revoked due to a change in the affiliation of the subject. Thus, from version 2 of the CRL format, it can be stated also the revocation reason. Besides, the suspension state is introduced where it is specified that the certificate is temporarily invalid. After a period of time, the reference to the certificate can be removed from the CRL, thus making the certificate valid again, or it can be definitely revoked.

The decision to revoke a certificate is the responsibility of the CA, usually as a response to a request coming from an authorized entity. According to its

Table 2: Sample Certificate Revocation List (CRL)

<i>Field name</i>		<i>Example</i>
Version		version 1
CA signature algorithm identifier		md5WithRsaEncryption
Issuer DN		CN=Politecnico di Torino Certification Authority, O=Politecnico di Torino, C=IT
Validity Period	This Update	Sep 4 12:15:11 2003 GMT
	Next Update	Oct 5 12:15:11 2003 GMT
Revoked certificates	Serial Number:	0127
	Revocation Date	Jun 9 12:54:18 2003 GMT
	Serial Number	00E7
	Revocation Date	Oct 3 07:53:38 2002 GMT
CA Signature		0B7FABDC D7C1C411 146FBD7E DF25FF6A 4AB7871E 023F18BB 4DA23262 90822E57 E1238370 58A2248D A6839F23 FCD56A8 E9CC4BC1 01476894 E391D5B0 B79D86C0 48F2EE52 7550A7E1 41CDFEDA DCB394C6 340A3B66 5C835BE7 00D3457A F1D3D349 4E12E0A0 8983B194 D0101B95 5A94B3E2 57FD72CA A0E96C01 66BB1CD4 F9C9F7B5

internal rules, the CA authenticates the source of the revocation request and, after taking the decision to revoke the certificate, the CA has the obligation to inform the PKI community about the revocation event. Several methods have been proposed to allow RP to retrieve the certificate status. CRL-based mechanisms are the primary methods for revocation notification in PKI: the RP retrieves the CRL from well-known servers. Alternatively, mechanisms that provide immediate notification of revocation have been proposed, such as OCSP, the on-line certificate status protocol (Myers et al., 1999).

All the available revocation mechanisms share the design goals of correctness, scalability, and availability:

- all verifiers must be able to correctly determine the state of a certificate within well-known time bounds;
- the costs for the determination of current revocation status of certificates should not grow exponentially with the size of the PKI user community;
- replicated repository servers should be provided in order to ensure service availability.

CRLs may be distributed by the same means as certificates, namely via untrusted channels and servers. The disadvantage is the increasing size of the CRL that leads to high repository-to-user communication costs. Thus, CRLs can introduce significant bandwidth and latency costs in large-scale PKIs. Another disadvantage is that the time granularity of revocation is limited to the CRL validity period; hence, the timeliness of revocation information is not guaranteed.

It is worth adding a security warning for the clients, signalling that in retrieving the CRLs without verifying the server's identity the risk exists that an obsolete CRL is sent. Clearly, an attacker cannot create a false CRL (without compromising the CA), but an attacker does have a window of opportunity to use a compromised key by denying access to the latest CRL and providing access to a *still-valid-but-obsolete* CRL. The term *still-valid-but-obsolete* implies that CRLs have a validity period. Certificates have a validity period clearly specifying the start date and the expiry date of the certificate. CRLs instead have values for the date when a CRL is issued (the `thisUpdate` field) and the date when the next CRL will surely be issued (the `nextUpdate` field). However, nothing prevents a CA from generating and publishing a new CRL (let us call it *off-cycle* CRL) immediately when a new revocation takes place (Ford & Baum, 1997). If an intruder deletes or blocks access to an *off-cycle* CRL from an untrusted server and leaves the previous periodic CRL in its place, this cannot be detected with certainty by the RP.

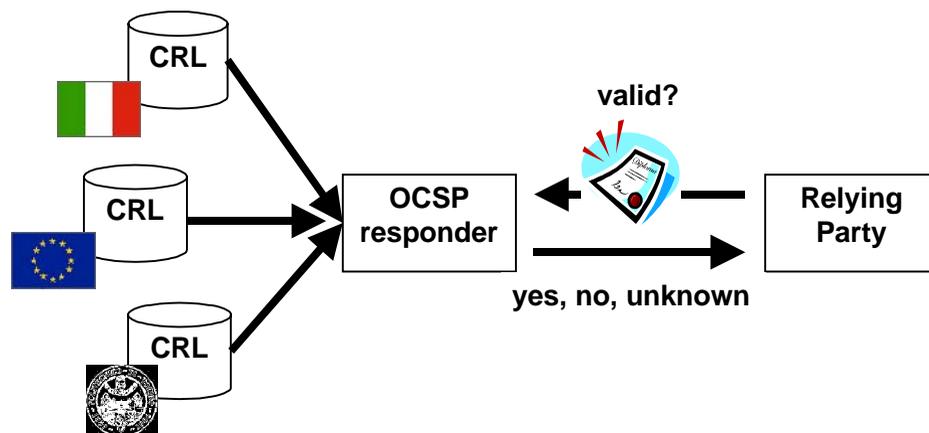
Another important observation is that a CRL does not become invalid when the next CRL is issued, although some applications behave as though it would be. *CRL Validity* interpretation eliminates the ability of the client to decide, based on the value of information, how fresh its revocation information needs to be. For example, if CRLs are issued every hour, a user might demand a CRL less than two hours old to authenticate a high value purchase transaction. If a CA issues CRLs every month, a user would rather prefer to be warned that, according to his application preferences, that CA's certificates shouldn't be used for high value purchases. This means that the user does not want the application to blindly treat them as fresh as certificates from CAs that issue CRLs daily or hourly. Unfortunately, there is no way to express the update frequency in a formal way inside the CRL; however, this information might be available in the certification practice of the CA.

Since the size of a CRL may increase, it is possible to split the CRL in non-overlapping segments. This can be done via the CRL distribution point (CDP) extension of X.509 certificates. A CDP is the location from which the RP can download the latest CRL for a specific certificate. By using different CDP for different sets of certificates (e.g., one CDP every 10,000 certificates), it is possible to set an upper limit to the size of a CRL. Usually, the CDP extension contains multiple access methods (such as LDAP, HTTP, FTP), to support as many applications as possible.

## Certificate Status Information (via OCSP)

The main alternative to CRL is the Online Certificate Status Protocol (OCSP) (Myers et al., 1999). This protocol allows applications to request the revocation status of a specific certificate by querying an online OCSP responder that provides fresh information about certificate status (*Figure 2*).

Figure 2: OCSP



The main advantage of OCSP is its speed, since it does not require downloading huge CRLs. The response is very simple and may convey three different values: valid, invalid, unknown.

Additionally, OCSP can provide more timely revocation information than CRLs (when it is fed directly with revocation data, before a CRL is generated) and seems also to scale well for large user communities. However, since certificate status validation implies a specific client/server request to OCSP responders, the mechanism can overload the network and generate an intense traffic toward the responder. Thus, even if the cost of user-to-repository communication is lower compared to the traffic involved in transmitting a CRL, there still is an intense communication toward the OCSP server. Since the revocation information is produced at the server, the communication channel between the relying party and the server must be secured, most likely by using signed responses. Signing operations could also limit the server scalability, since digital signature generation is computationally intensive.

On the other hand, the decision to trust an OCSP responder is an important decision to be made. Consequently, all the issues related to the distribution and maintenance of the trusted CA's public keys will apply to the OCSP responder's public key too. For revocation notification in enterprise environments, there should exist an additional mechanism to manage and enforce trusted responders in a centralized manner. Revocation of OCSP server's public key requires usage of an alternative revocation method for checking server's public key status. The OCSP responses are signed objects and so the OCSP client must verify the validity of the signature on them. In order to verify the validity of the signature on the OCSP response messages the OCSP client has to verify the status of the OCSP responder certificate. But she cannot ask to the OCSP responder if its

certificate is still valid, because in the hypothesis that the OCSP responder private-key has been compromised its responses can be manipulated and then give false status information.

In other words, how will the OCSP clients verify the status of the responder's certificate? In the standard three alternatives are mentioned to solve this issue.

In the first alternative, the client trusts the responder's certificate for the entire validity period specified within. For this purpose the CA issues the responder's certificate in question adding a special non-critical extension called *id-pkix-ocsp-nocheck*. However, the effects of such a choice are self-evident. In case the responder's private key is compromised in any way, the entire PKI is compromised. An attacker having the control of the responder's private key can provide any status information to the PKI community. Therefore, when this alternative is chosen to be deployed in a PKI, the CA should issue the responder's certificate with a very short validity period and renew it frequently. This would cause also the OCSP client to download too often the fresh OCSP responder certificate from the certificate repository.

In the second alternative, the OCSP clients are not suggested to trust the responder certificate and an alternative method of checking the responder certificate status must be provided. Typically, a CRLDP extension is inserted into the responder's certificate when CRLs are employed, or an *Authority Information Access* (AIA) extension is provided in the responder's certificate if other means of revocation notification should be employed.

The third case is when the CA does not specify any means for checking the revocation status of the responder certificate. In such case, the OCSP client should fall back to its local security policy in order to decide whether the certificate at hand should be checked or not.

Another interesting feature of OCSP is pre-production of response. For this, OCSP responders could produce, at a specified moment of time, a complete set of responses for all the certificates issued by the CA. This procedure has its advantages and disadvantages. The advantage is that the responder saves up its computational resources by having available the pre-produced responses. The disadvantage comes from the fact that, by means of pre-produced responses, the server exposes itself to replay attacks. This type of attack would typically consist of an attacker sending back to an OCSP client an old response with a good status inside just before the response expires and after the certificate was revoked. However, this type of attack is feasible also for responses generated on the fly. To prevent replay attacks, a special unique value (named "nonce") could be inserted in the request and copied back by the responder into the signed response. In this way, a replay attack would be immediately detected.

The OCSP responders are vulnerable to yet another type of attack, namely the denial of service (DoS) attack. This vulnerability is evident when imaging a

flood of queries toward the server. The flaw is sharpened by the requirement to sign the responses. This is a fact that slows down the capacity of the responder to answer queries and eventually can take it to a halt. Nevertheless, producing unsigned responses would not be an alternative, since the attacker could be able to send false responses to the clients on behalf of the responder. For this reason, the protocol implementers must carefully consider the alternative of restricting access to the responder by accepting only signed requests from known partners or by using other access control techniques.

In general, OCSP is better for fast and specific certificate status lookup at the present time, as needed for online transactions, while CRLs are superior in providing evidence for long periods of time, as needed for archival of electronic documents.

## X.509 Extensions

Since version 3, X.509 has defined a mechanism to extend the certificate format to include additional information in a standardized and yet general fashion. The term *standard extension* refers to those extensions that are defined in the standard itself while the term *private extension* refers to any extension defined by a single company or closed group. For example, before the definition of OCSP, Netscape defined the *NetscapeRevocationURL* extension supported by its products to provide certificate status lookup.

Each extension consists of three fields, namely the *extension type*, the *extension value* and the *criticality bit*. While the meaning of the first two fields is straightforward, special attention must be paid to the *criticality* field, which is a single-bit flag. When an extension is marked as critical, this indicates that the associated value contains information that the application cannot ignore and must process. If an application cannot process a critical extension, the application should reject the whole certificate. On the contrary, if an application encounters an unrecognised but non-critical extension, it can silently ignore it.

Note that the certificates containing critical extensions are defined by the CA to be used for a specific purpose. If an application encounters a critical extension and does not process the extension in accordance with its definition, then the CA is not liable for the misuse/processing of the certificate. Thus, certificate extensions are not only related to technical issues but also to legal aspects.

Standard certificate extensions are grouped into four main categories: certificate subject and certificate issuer attributes, key and policy information, certificate path constraints, and CRL distribution points.

The *certificate subject* and *certificate issuer attributes* extensions support alternative names of various forms, to identify the subject or the issuer in a way consistent with the application that requires the certificate. These extensions can also convey additional information about the certificate subject,

to assist a relying party's application in being confident that the certificate subject is a specific person or entity.

The *subject and issuer alternative name* extensions allow one or more unique names to be bound to the subject of the certificate. These extensions support several forms of names, such as email identifiers, domain names, IP addresses, X.400 originator/recipient addresses, EDI party names and much more. These additional names are very valuable to perform additional security checks at the application level. For example, if the issuer inserts the RFC-822 email address of the user in the *subject alternative name* of her certificate (e.g., `alice.twokeys@polito.it`), then the secure email applications can associate the sender of an email with her cryptographic identity. This is actually what happens in S/MIMEv3 (Ramsdell, 1999) where the *subject alternative name* is specified as the preferred mean to verify the correspondence between the "From" header of the RFC-822 message and the identity present in the certificate used to digitally sign the email. If the values do not match then the S/MIME mail user agent usually displays a warning message.

The *key information* extensions convey additional information about the keys involved, to identify a specific key or to restrict key use to specific operations.

The *authority key identifier* extension allows one to identify a particular public key used to sign a certificate. This is the case when a CA uses two key pairs (one for low and one for high assurance operations). The identification of the key can be performed in two ways: either with a key identifier, which typically is the digest of the public key, or with the pair issuer name and serial number. This extension is always non-critical but, nonetheless, in some software it is very important because it is used for constructing the certification paths.

The *key usage* extension identifies the range of applications for which a certain public key can be used. The extension can be critical or non-critical. If the extension is critical, then the certificate can be used only for the cryptographic operations for which the corresponding value is defined. For example, if a certificate contains the extension *key usage* with the values set up to Digital Signature and Non-Repudiation then the certificate can be used to generate and to verify a digital signature, but not for other purposes. As another example, if a certificate contains the *key usage* extension with the value set to Key Encipherment then the corresponding public key in the certificate can be used in a key distribution protocol to encrypt a symmetric key.

The *policy information* extensions convey additional information about the policy used in certificate creation and management. A *certificate policy* is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of certificate to the authentication of electronic transactions for trading goods up

to a given price. This extension is very important as it is being used to establish the validity of a certificate for a specific application. Relying party applications with specific certificate policy requirements are expected to have a list of acceptable policies and to compare the policies in the certificate to those in this list.

The *certification path constraints* extensions allow constraints to be included in a CA certificate to limit its certification power. The following types of constraints are defined:

- *basic constraints* tell whether an entity is a CA or not, i.e., whether it is authorised to issue certificates or if it is a leaf in the certification tree
- *name constraints* restrict the domain of trustworthy names that can be placed by the CA inside the certificates (e.g., only a certain subset of email identifiers or IP addresses)
- *policy constraints* restrict the set of acceptable policies that can be adopted by the CA in its operations.

These extensions are very important and they must be processed correctly when a relying party application must determine the validity of a certificate. For example, one paper (Hayes, 2001) describes a security attack, named certificate masquerading attack, which successfully occurred because the certificate-enabled application did not properly apply the external name constraints and policies.

The *CRL distribution point* extension identifies the point of distribution for the CRL to be used in the process of determining the validity of a certain certificate. The value of this extension can be either a directory entry, an email address or a URL.

## PKIX Certificate Profile

Extensions are a good way to make the certificates more flexible and accommodate different needs. However, this can make interoperability a nightmare. Therefore, several bodies have started to define *certificate profiles* that suggest which extensions to use for specific applications or environments.

Among all bodies that defined profiles, the work of the IETF-PKIX group is particularly important because it applies X.509 certificates to the security of common Internet applications, such as web protection via SSL/TLS, email security via S/MIME, and the protection of IP networks via IPsec. The PKIX profile was originally defined by RFC-2459 (Housley et al., 1999) and later updated by its successor RFC-3280 (Housley et al., 2002). The profile suggests the use of X.509v3 certificates, coupled with X.509v2 CRL, and addresses the following issues:

- the format and semantics of certificates and certificate revocation lists for the Internet PKI;
- procedures for processing the certification paths;
- encoding rules for popular cryptographic algorithms.

In addition to the standard extensions, PKIX defined several private extensions. However, since the group that defined these extensions is the Internet itself, extensions can hardly be regarded as private. These PKIX extensions are subject information access, authority information access and CA information access.

The *subject information access* extension specifies a method (e.g., http, whois) to retrieve information about the owner of a certificate and a name to indicate where to locate it (address). This extension is fundamental when X.500 is not used for certificate distribution. The *authority information access* specifies how to get access to information and services of the CA that issued a certificate, while *CA information access* specifies how to get access to information and services of the CA owning the certificate.

In addition to these new extensions, the PKIX profile introduces new application-specific values for the *extended key usage* extension, in addition or in place of the basic purposes indicated in the key usage field. For example, if a certificate has the value of this extension set to *server authentication* then it can be used to authenticate the server in the TLS protocol. Other values are defined for TLS client authentication, OCSP signing, timestamping generation, code authentication and email protection.

## Certificate Validation

Once a certificate has been issued, the task of the CA is completed. However, when a relying party accepts a digital signature (and hence the associated pub certificate), it is its own responsibility to check for the certificate's validity. This is quite a complex task that requires many actions to be taken and many checks to be performed, but it is at the heart of trust in PKIs.

Suppose that Alice receives a digitally signed message by Bob and she needs to validate the certificate that comes along with the message. First Alice needs to check the authenticity of the CA signature, for which it is necessary to obtain the public key of the CA. Next, if Alice directly trusts this CA then she can validate Bob's certificate immediately. Otherwise Alice needs to get a set of certificates that start from Bob's one, continue with all the intermediate CA's certificates up to a CA that Alice trusts (called also trust anchor, TA). This ordered set of certificates is called a *certification path* or *certificate chain*. More than one certification path can exist for Bob's certificate. The process of finding all of them is called *certificate path discovery (CPD)*. CPD tries to find a certificate sequence that leads to a trusted CA. This may require constructing

several certificate chains before finding an acceptable one. The process of constructing a certificate path rooted in a trusted CA is called *path construction*.

Once the certification path is built, Alice needs to execute a *path validation* algorithm that takes as input the certification path and returns as output whether the certificate is valid or not. Both the ITU (ITU-T Recommendation, 2000) and the IETF (Housley et al., 2002) provide sample algorithms for path validation. These algorithms are a reference to establish the correct results of path processing, but are not necessarily the best or most optimised way to validate certification paths. They were chosen because of the ability to describe them fully and accurately. RPs are free to implement whatever path validation algorithm they choose, as long as the results are guaranteed to be the same as these standard algorithms. The path validation algorithm contains the following steps:

- a) *Syntax check*. Parse and check the syntax of the digital certificate and its contents, including some semantic check like use of certificate compared to allowed use (*key usage* extension), presence of mandatory fields and critical extensions.
- b) *Signature validation*. Validate the CA's signature on the certificate. This requires a trusted copy of the CA's own public key. If the CA is not directly trusted then a certification path (or certificate chain) must be constructed up to a trusted CA. The definition of the certificate chain is given.
- c) *Temporal validation*. Check that the digital certificate is within its validity period, as expressed by the "not before" and "not after" fields in the certificate. For real-time checking, this must be compared against the current time, while for old signed messages, the signature time must be considered.
- d) *Revocation status*. Check that the certificate is not revoked; that is, declared invalid by the CA before the end of the validity period. This may require a CRL lookup or an OCSP transaction.
- e) *Semantic check*. Process the certificate content by extracting the information that shall be presented to the relying party either through a user interface or as parameters for further processing by the RP application. This should include an indication of the quality of the certificate and of the issuer, based on the identification of the certificate policy that the CA applied for certificate issuance.
- f) *Chain validation*. In case a certificate chain had to be constructed, the above steps must be repeated for each certificate in the chain.
- g) *Constraints validation*. Execute controls on each element of the path to check that a number of constraints have been respected (e.g., naming and policy constraints).

A valid certificate is a certificate signed by a CA that a relying party is willing to accept without further checks — that is, the certificate has a CA trust point, the certificate's signature can be verified, the certificate is not revoked and the certification path up to a trusted CA can be constructed and is valid (Housley, 2001).

In simple hierarchical closed environments, where all the entities trust a single root, certificate validation seems a trivial task. However, security attacks can be performed at the application level, as explained in Hayes (2001). When multiple CAs exist, if the entities that trust different CAs want to communicate among themselves, the CAs must establish efficient structuring conventions to create trust between each other. These conventions are often called PKI trust models and are discussed later. Generally speaking, a PKI trust model is employed to provide a chain of trust from Alice's trusted anchor to Bob's certificate.

In order to perform certificate validation, it is important to know where certificate status information is available and to locate the certificate of the CA that issued the certificate being validated. Unfortunately, there is not yet general agreement about where this information should be put inside the certificate. The following X.509 extensions could be used:

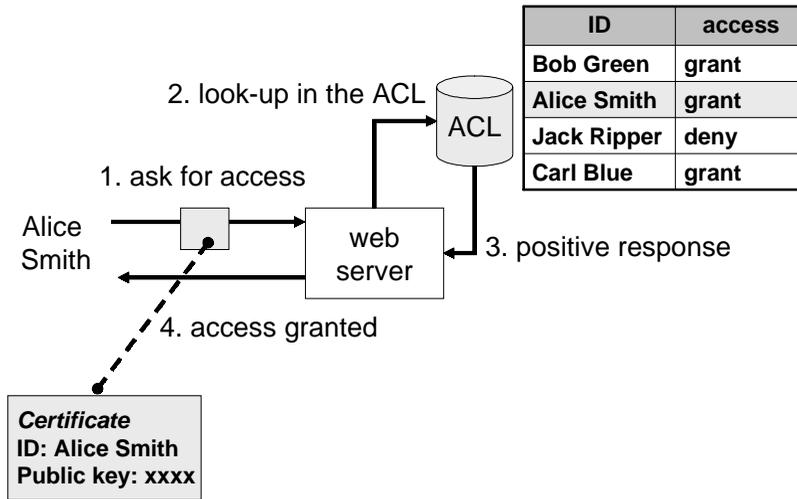
- the *Issuer Alternative Name* extension can contain an URI related to the issuer, but the interpretation of this URI is entirely application specific
- the *CRL Distribution Point* can contain the location of CRLs, but too often this field is omitted
- the *Authority Info Access* extension can contain an URI indicating the location of the CA certificate and/or the location of an OCSP responder

If a certificate does not contain any of the above extensions – as in the Verisign-Microsoft case (Microsoft, 2001) – then the certificate's revocation status cannot be automatically checked, unless it is configured in some other way into the relying party application.

## Attribute Certificates

Attribute certificates are an extension of the identity certificates and were introduced to offer a robust, distributed and scalable system for the management of authorizations. In fact, identity certificates can contain attribute information to be used for authorization purposes. For example, a widely used access control method employs Access Control Lists (ACLs), together with the identity information placed inside a public-key certificate. This technique is based on a list of records that state the authorization granted by the system to an identity. *Figure 3* shows a sample system that uses an ACL for controlling user access to a web server. The performed steps are as follows:

Figure 3: Example ACL



- the user sends her public-key certificate to the server
- the server checks the certificate validity and then engages the user into a challenge-response protocol to check possession of the corresponding private key
- if the check is positive then the identity extracted from the certificate is used as a search key in the ACL and the selected action is executed

Other systems base the ACL not on identities, but rather on roles or authorizations. By using X.509 extensions (such as the *Directory Attributes* one), roles and authorizations can be directly inserted into an identity certificate. However, this solution exposes the certificate to a higher risk of revocation if any of the roles or authorizations change. Moreover, the CA is rarely the correct entity with the right to state roles and permissions that are usually defined directly by the application servers.

To avoid these problems, the Attribute Certificate was defined with the idea to store privilege information in a structure similar to that of a public key certificate but with no cryptographic key. This type of certificate is used for the express purpose of storing privilege information and has been standardized in X.509v4 (ITU-T Recommendation, 2000). The main fields of an X.509 AC are illustrated in *Table 3* and discussed in the following:

- *Version*: This field indicates the version of the format in use. For attribute certificates conforming to the standard (ITU-T Recommendation, 2000) the version must be v2.

Table 3: Main fields of an attribute certificate

<i>Field name</i>		<i>Example</i>
Version		version 2
Serial number		3514534
Signature algorithm identifier for AA		RSA with MD5
Issuer		CN=Politecnico di Torino Attribute Authority, O=Politecnico di Torino, C=IT
attrCertValidityPeriod		start=01/01/2001, expiry=01/02/2002
Holder	issuer	CN=Politecnico di Torino Certification Authority, O=Politecnico di Torino, C=IT
	serialNumber	12345
Attributes	type	2.5.4.72 (role)
	value	Student
AA Signature		EF1dGhvcml0eTCCASiWl2Nybc5kZXIwTgYDV R0gBEcgEEAakHAQEBMDUwMwYIKwYBBQUHAgE WJ2h0dHeg6a5r61a4jUqp4upKxuzgu6unsw/ +RkU2Kz1Nm053JOcsZs/0IFiMW1GJB2P7225 WWDF01OtQcmLYspoi ffUPy2g+KvCG1b9zHmf JoaDn5y+kQQpHs/ZIZeUyNe9ULi fu3GgG

- *Holder*: This field identifies the principal with which the attributes are being associated. Identification can be either directly by name or by reference to an X.509 public key certificate (by a pair issuer name and certificate serial number).
- *Issuer*: This field identifies the AA that issued the AC.
- *Signature*: This field indicates the digital signature algorithm used to sign the AC.
- *Serial Number*: This field contains a unique serial number for the AC. The number is assigned by the issuing AA and used in a CRL to identify the attribute certificate.
- *attrCertValidityPeriod*: This field may contain a set of possibly overlapping time periods during which the AC is assumed to be valid.
- *Attributes*: This field contains a list of attributes that were associated to the AC owner (the owner is the principal that is referred to in the subject field). This field is a sequence of attributes, each one defined as a pair *type* and *value*. The standard allows one to specify in a single AC a set of attributes for the same owner. The information may be supplied by the subject, the AA, or a third party, depending on the particular attribute type in use.

With an attribute certificate of this type, the issuer declares that the subject has the set of attributes listed in the attributes field. Attributes certified by an AC can be anything, from group membership (e.g., the subject belongs to group of

administrators), to financial limitations (e.g., the subject has an upper limit of 500 Euro for online transactions). The *attrCertValidityPeriod* field indicates the period of time for which the attributes hold for the subject.

## SDSI/SPKI Certificates

Ronald L. Rivest & Butler Lampson, in their paper (1996), proposed a new distributed security infrastructure. The motivation of this proposal lies in the authors' perception that many PKIs (such as those based on X.509 certificates) were incomplete and difficult to deploy due to their dependence on a global name space, like the one proposed by X.500.

The main goal of the SDSI infrastructure is to eliminate the need of associating an identity to a key. The key itself is the identifier of the key-holder. According to its authors, this is a "key-centric" system. A public key represents and speaks for the entity that controls the associated private key. Rivest & Lampson base their arguments on the fact that names are never global, but have value only in a local space. Names are bound to a person by experience and have, therefore, always a limited scope. However this leaves room for two entities having the same name: this problem is called by Ellison the "John Wilson problem" (Ellison, 2002). To differentiate between such two entities with the same name, a large quantity of personal details is needed (e.g., birth date and place, town of residence, name of parents and many other issues that would form a dossier of personal information that would violate privacy). In SDSI, names are always related to a very limited and specific context and should therefore disambiguate very easily.

Each public key, along with the algorithm used to generate it, is contained in an object called a *principal*. Signatures are appended to the object or can also be detached from it. Objects can be co-signed by many signers.

A signed object is a particular data structure that must contain:

- the hash of the object being signed, along with the algorithm used to generate it
- the signature date
- the output produced by the signature algorithm

Below is an example of a SDSI signed object:

```
( Signed:
( Object-Hash: (SHA-1 =7Yhd0mNcGFE071QtzXsap=q/uhb= ) )
( Date: 1996-02-14T11:46:05.046-0500 )
( Signature: #3421197655f0021cdd8acb21866b)
( Re-confirm: PT8H ( Principal: ... ) ) )
```

The SDSI signed object can have an absolute expiration time or may need a reconfirmation. An absolute expiration time means that the signature is valid until the date present inside the object. If due to some accident, e.g., a private-key compromise, the signature is not valid anymore, then there is the need of a procedure to revoke the signed object. Reconfirmation is the opposite way of thinking: a signature is valid until the signer reconfirms it. In this paradigm the signature is valid if the time passed from the signature date is less than the reconfirmation period. The signed object can contain the *Expiration-date*: or the *Re-confirm*: attribute; in the latter case, a time interval is specified in ISO (8601:1998) format: for example, PT8H says that the reconfirmation is needed every eight hours.

The *Re-confirm*: field has an interesting optional value used to specify a different reconfirmation principal from the original signing one. This is the case, for example, of a server specialized in the reconfirmation duty. The SDSI architecture does not rely on Certification Revocation List model in order to deny validity to a signature but on a reconfirmation method. The paper (Rivest & Lampson, 1996) proposes a client-server protocol in which someone who needs to verify a signature could query the original signer itself or a delegated entity for reconfirmation of the signature.

SDSI identity certificates are used to bind a principal to a person, and, because humans should always examine the identity certificates, they should always contain some readable text to describe the person being certified. An identity certificate is the union of some data to a signed object. It contains:

- a local name in the field *Local-name*
- a principal in the fields *Value:(Principal:)*
- a description of the certified entity in the field *Description*:
- a signature (signed object) in the field *signed*

The local name is chosen arbitrarily; however, a system exists to link all the names. For example, to refer to Alice Smith, Bob's best friend and from him certified, it could simply be said Bob's Alice or, in formal notation,

( *ref: bob alice* )

while a reference to Alice's mother would be:

( *ref: bob alice mother* )

Here is an example of an SDSI identity certificate:

```
( Cert:
  ( Local-Name: Alice)
  ( Value: (Principal: ...))
  ( Description:
    [text/richtext]
    "Alice Smith is a researcher at the Politecnico di Torino.
    ( Phone: 39-011-594-7087 )
    ( signed: ...))
```

In SDSI, each principal (that is, each public key) can act as a certification authority; that is, can bind a key to an identity by a signature. In doing so the principal should use his personal judgment about the identity of the key owner being certified.

Based on the SDSI theory, the SPKI model (Simple PKI) was proposed to the IETF in February 1997 by a working group, which terminated its work in 2001. The main target of the SPKI certificates (Ellison, 1999) is *authorization* rather than *authentication*. SPKI identifies the problem of having a unique identifier for a person as being the problem that led the X.500 project to failure. Then SPKI uses local names to represent entities. But relationships on the net are established between people who have never met in person or who do not have common friends or references. So, SPKI elects as a global identifier the public-key itself or a hash free function of the public-key. This is based on the assumption that any public-key is different from another.

## PKI COMPONENTS

A Public Key Infrastructure (PKI), as defined by the IETF working group PKIX (Arsenault et al., 2002), is “the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public-key certificates based on public-key cryptography.”

The main components of a PKI are:

- certification authority (CA)
- registration authority (RA)
- local registration authority (LRA)
- repository
- relying party (RP)
- end entity (EE).

The *Certification Authority (CA)* is the authority trusted by users. The main duties and responsibilities of a CA are:

- issue certificates
- manage certificates (suspend, renew, publish, revoke)
- generate and publish certificates status information
- keep safe its own private key
- keep safe the CA hardware and software.

The CA's private key is the heart of the whole system. The safety of this key is one of the main responsibilities of a CA. To this aim a CA should adopt any security measure it can afford. At least the key should be stored in an encrypted form and possibly it should be used only by some hardware secure crypto-device, such as a smart-card or a cryptographic accelerator. To achieve a higher degree of security, the key to decrypt the private key or to activate the crypto device could be divided into two parts, each one held by a different person that need to join to operate the CA. Another good practice to protect the CA is to keep the system that issues certificates and CRLs off-line in a secure vault.

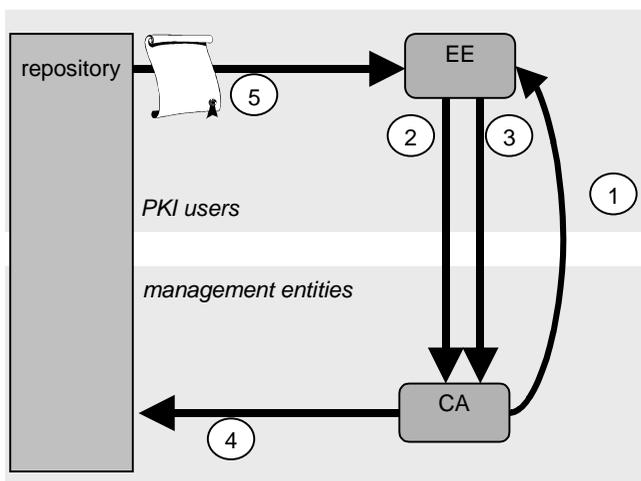
Despite any security measure implemented by a CA, there is always the chance of a human error or attack based on human misbehaviour (the so-called "social engineering" attacks). Therefore the CA should also pay special attention to its internal procedures and to the phase of user registration that is usually delegated to a different component (the RA), whose behaviour should be periodically audited.

The *Registration Authority (RA)* is the entity responsible for the registration of users requiring a certificate. It verifies the identity of the certificate applicants and the validity of their certificate requests. It is an optional part in the PKI architecture because in its absence its duties can be carried out directly by the CA, but it is extremely useful, not only because it relieves the CA of non-technical duties (like control of the applicants' identity), but also because it can be physically located closer to the users to be certified.

The *Local Registration Authority (LRA)* is a registration authority that is local to a specific and limited community of users. The users who belong to this community and want a certificate from their community CA should go to the LRA to have their request and identity verified. Since the LRA is closer than the CA to the user pool that it should serve, it is likely that it can identify them more easily. Quite often the terms RA and LRA are used interchangeably, while other times the term RA is used for the general concept (that is, to represent the set of all LRAs).

The *repository* is the logical storage for certificates and other information (such as CRLs) made publicly available by the CA. The repository can be implemented in different technologies and accessed by different access proto-

Figure 4: Sample CA



cols (LDAP, HTTP, FTP...). Since it usually contains only signed information (PKCs and CRLs), no special security measure is needed when accessing it via network. However, to avoid denial-of-service attacks and to offer a better service to its customers, the repository is usually replicated at different sites and sometimes it implements access control measures (such as SSL/TLS client-authentication or password-based authentication over a secure channel).

A *relying party* (RP) is an entity that would make some decision based on the certificate content. Basically, a relying party is someone or something whose action depends on the certificate content and on the result of the certificate validation process. For example, an e-banking server acts as a relying party when it accepts and checks a user certificate for authentication in accessing a bank account.

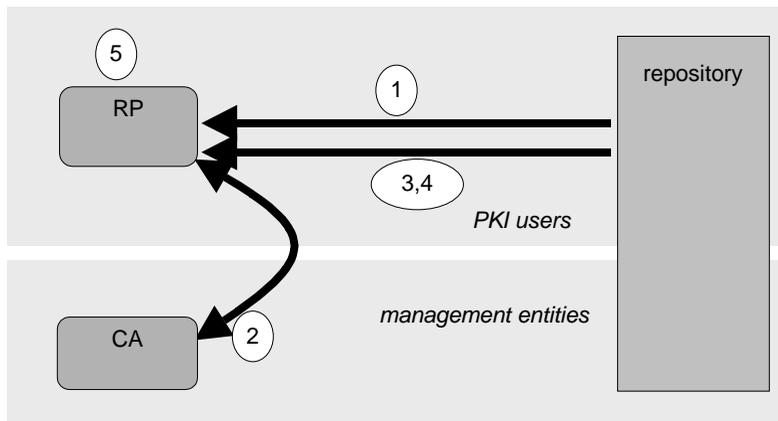
The *end-entity* (EE) is the holder of the private key corresponding to the public one certified and consequently the subject of the certificate. It can be a certificate user or a Certification Authority.

Using these basic components, several PKI architectures can be built. The simplest PKI architecture is composed merely of a CA, a repository and an end-entity.

In this architecture the steps performed to issue a certificate are shown in Figure 4:

1. The EE requires the CA certificate and verifies it with an out-of-band method; if the verification is successful, then the EE sets the public key found in the CA certificate as her trust point.

Figure 5: Certificate usage



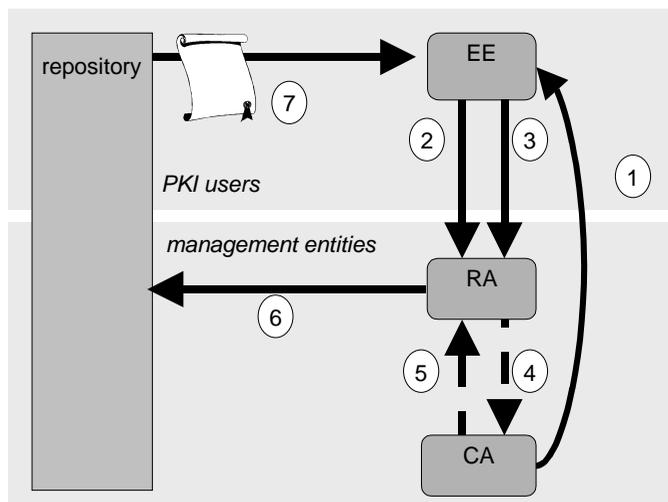
2. The EE generates an asymmetric key pair, stores locally the private key, inserts the public key in a certificate requests that is sent to the CA.
3. The EE physically goes by the CA to be authenticated according the CA's procedures; this step could also occur before the second one.
4. If authentication (step 3) is successful (i.e., the EE's identity is verified) the CA issues the certificate and publishes it into the repository
5. The EE downloads her certificate from the repository.

As shown in *Figure 5*, when an RP needs to use the EE's certificate, it has to:

1. Fetch the CA certificate from the repository.
2. Verify the CA certificate with an out-of-band method (steps 1 and 2 can be avoided if the RP already performed them in the past or if the RP has pre-configured trust CAs).
3. Fetch the EE's certificate from the repository.
4. Check the certificate status (e.g., by fetching the CRL from the repository).
5. Verify the EE's certificate validity (e.g., check that the current date is within the validity period).
6. Use the key found in the EE's certificate for the application needs (e.g., to encrypt data to be sent to the EE or to verify an EE's signature).

In both procedures, the first steps (those that establish the CA as a trust point for the EE or the RP) are very critical and difficult because they require out-of-band verification and human decision. If these steps are successful, the EE becomes part of the PKI built around the given CA and the RP trusts the CA

Figure 6: Simple CA with RA



for its application needs. The importance of both things should not be underestimated and should require a careful analysis of the CA technical configuration and administrative procedures to see if they match the needs of the EE or RP.

Usually, to help parties in performing these checks, the CA makes publicly available its Certificate Policy (CP) and Certificate Practice Statements (CPS); pointers to these documents can be inserted in the certificatePolicies extensions of X.509 certificates.

A simple architecture that includes a RA is shown in *Figure 6*. In this architecture, the EE sends the certificate request to the RA rather than to the CA, and goes to the RA to be properly identified.

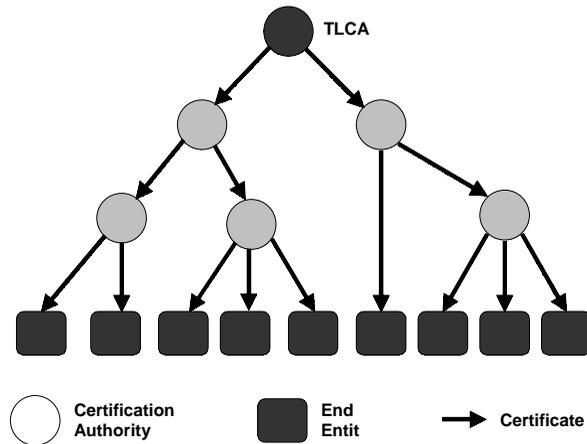
## PKI TRUST MODELS

This section describes several different trust models for CA interconnection and evaluates the relative advantages and disadvantages.

### Hierarchical PKI

A strict hierarchy of certification authorities is shown graphically as an inverted tree, with the root CA at the top and the branches extended downward (*Figure 7*). The root CA issues certificates to its immediate descendants, which in turn certify their descendants, and so on. The CA at the top of the hierarchy, also known as TLCA (Top Level CA), is the trust anchor for the entire PKI. Each CA between the root and the subscribers is referred to as *intermediate CA*.

Figure 7: Hierarchy



In a hierarchical PKI, trust starts at the TLCA and flows down the hierarchy through a chain of subordinate CAs to the end-entities. All intermediate CAs have a certificate issued by their parent CA, but the TLCA which has a self-signed certificate as it does not have a parent CA. Therefore all EE certificates can be verified electronically but that of the TLCA. As a consequence it is very important that the certificate of the TLCA is distributed to EEs in a secure way. If someone succeeds in substituting the certificate of the TLCA maintained by an EE with another one, he can get the EE to trust a completely different infrastructure.

The hierarchical PKI model is the first that was introduced and therefore it is well understood and supported by nearly all PKI products and commercial CA service providers. Also, two famous electronic financial transaction systems (SET and Identrus) use this model to organize their PKI.

A new CA can enter a hierarchical PKI through the process of subordination, in which an existing CA issues a certificate for the new CA. Thus, certificates are issued in only one direction – from parent to child – and a CA never certifies another CA superior to itself. The subordination process is transparent and does not impact the users of the hierarchy that can correctly process the certificates issued by the new CA without any configuration changes. Integrating an existing, foreign CA into a hierarchical PKI, however, is more problematic as it requires the users of the foreign CA to directly trust the root CA of the hierarchical PKI, which may be difficult to achieve in a peer-to-peer business relationship.

The hierarchical trust model offers a scalable, easy-to-administer PKI because each CA serves a specific user community in the hierarchy. Each

member CA processes enrolment requests, issues certificates, and maintains revocation information for its own community. The extension of this community can be restricted by the parent CA via appropriate usage of the X.509 NameConstraints and PolicyConstraints certificate extensions. They permit or restrict the set of names of EEs that can be certified (e.g., only the mail addresses of the form “\*@polito.it”) and the applicable policies: these are clear advantages over other trust models, as pointed out by Linn (2000).

Another advantage of the hierarchical model is the simplicity in constructing the certification paths between any two entities. This simply requires the RP that wants to validate an EE certificate to retrieve issuer certificates until it finds a certificate issued by the trust anchor. The direction of path construction is *bottom-up*. With bottom-up chain building, an application starts with an end-entity’s target certificate and uses the information in the certificate to locate the issuer’s certificate, iterating the process until the TLCA is reached. This process is further simplified by the convention adopted by many secure applications (such as SSL/TLS web servers and clients, and S/MIME mailers) to send the whole chain (up to the TLCA) along with the EE certificate.

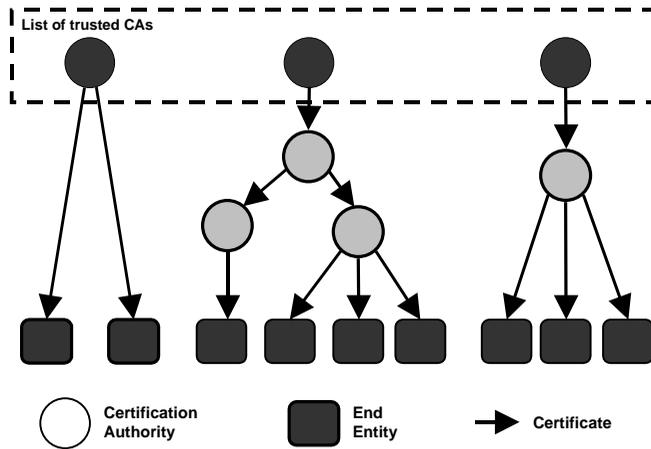
An important disadvantage of the hierarchical PKI model is the existence of a single point of failure: if the private key of the TLCA is compromised then so is the whole PKI. Instead, if the private key of an intermediate CA is compromised, then only its subordinate branch is compromised and the damage can be limited by quickly revoking the certificate of the compromised CA. This has the effect to invalidate all the certificates issued by any CA belonging to subtree rooted in the compromised CA. This subtree must then be completely reconstructed.

In conclusion, this model is very successful, but it is mainly applicable within isolated, hierarchical organizations, be they a multinational enterprise or a national government. It is more difficult to apply across organizational boundaries (Linn, 2000) where there is clearly a political – rather than technical – issue: it is hard to identify a single entity trusted by all communicating parties and to establish common policies acceptable to all participants. Participants are reluctant to rely on other organizations to preserve the integrity of their subordinated namespaces.

## Trust List

The trusted list model (*Figure 8*) requires RP applications to maintain a list of trusted TLCAs. In this way, interoperability between different hierarchical PKIs is achieved at the application level of the RP. This model is the most successful on a commercial ground: most current secure applications (SSL browsers, S/MIME mailers) come pre-configured with a list of commercial trusted root CAs.

Figure 8: Trusted CA list



However this model presents some security weaknesses. The main problem is that it moves trust management away from the CAs towards end users. The organization must rely on the users to take the necessary actions to handle (add, remove, check) the CA from the trust list, to configure policies and to maintain certificate status information up to date. However, the typical user has little or no idea of what a PKI is and which are the policies or operating practices of the various roots, and could be easily fooled to believe into a phoney CA. The alternative is to perform extensive management actions to configure all end users' applications and/or workstations, to achieve a certain uniform trust policy across an organization. Also, the revocation of a TLCA is nearly impossible, since it requires deleting the certificate of that TLCA from the trust list of every end user. Last, but not least, there is no way to limit the span of a specific hierarchy, because the TLCAs in the list have all the same value, while user-defined name constraints would be needed to restrict a specific hierarchy to a subset of the global namespace.

Trust lists are useful when they involve only a relatively small numbers of globally well-known CAs, for direct use within enterprises, and/or to support interactions across a predefined set of enterprise boundaries.

## Cross-Certification

In order to overcome the problem of hierarchical PKI when interconnecting different realms, various models based on the concept of cross-certification have been proposed.

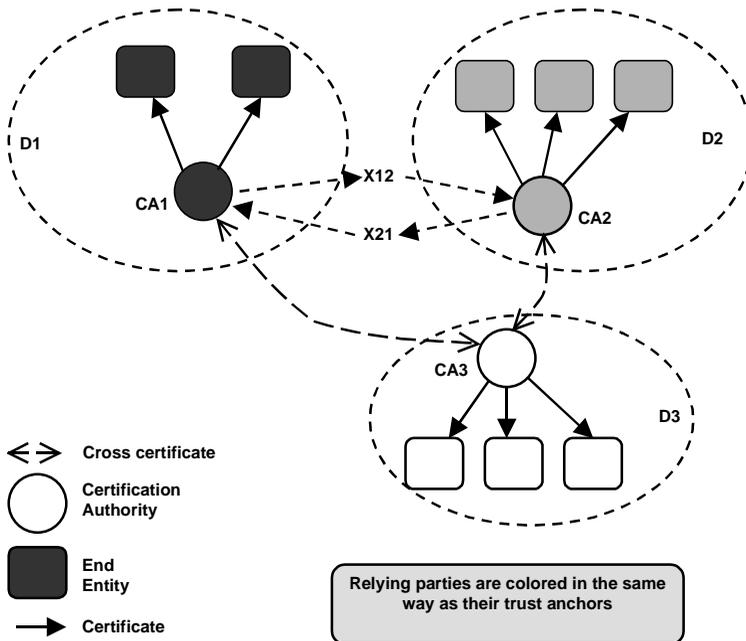
In the *mesh* (or *network*) *trust model*, all CAs are self-signed, and trust flows through the network via cross-certificates. An EE directly trusts only the

CA that issued its certificate, and trusts another CA only if its direct CA has cross-certified the foreign CA. Because cross-certification creates a parent-child relationship between two CAs, a network PKI can also be viewed as a hierarchical PKI, with the difference that many self-signed roots and many hierarchies exist at the same time.

Figure 9 shows a sample networked PKI with three PKI domains that reciprocally trust each other through six cross-certificates. The certificate X12 represents a *cross-certificate* between CA1 and CA2; CA1 is the *cross-certifying CA*, whereas CA2 is the *cross-certified CA*. The certificate X21 plays the reverse role of the X12 certificate and allows CA2 to cross-certify CA1. The combination of X12 and X21 cross-certificates creates a *bilateral cross-certification* between domains D1 and D2.

Figure 9 depicts a mesh PKI that is fully cross-certified; however, it is possible to deploy an architecture with a mixture of uni-directional and bi-directional cross-certifications. A new CA enters a networked PKI through the process of *cross-certification*, in which an existing CA issues a cross-certificate for the new CA. An existing CA leaves the network by revoking its cross-certificates. The cross-certification process is transparent and does not impact the users of the network, provided that they can retrieve the cross-certificates from a global directory. The cross-certification process can also integrate an

Figure 9: Mesh



existing, foreign CA into a networked PKI without changing the relative point of trust for either PKI.

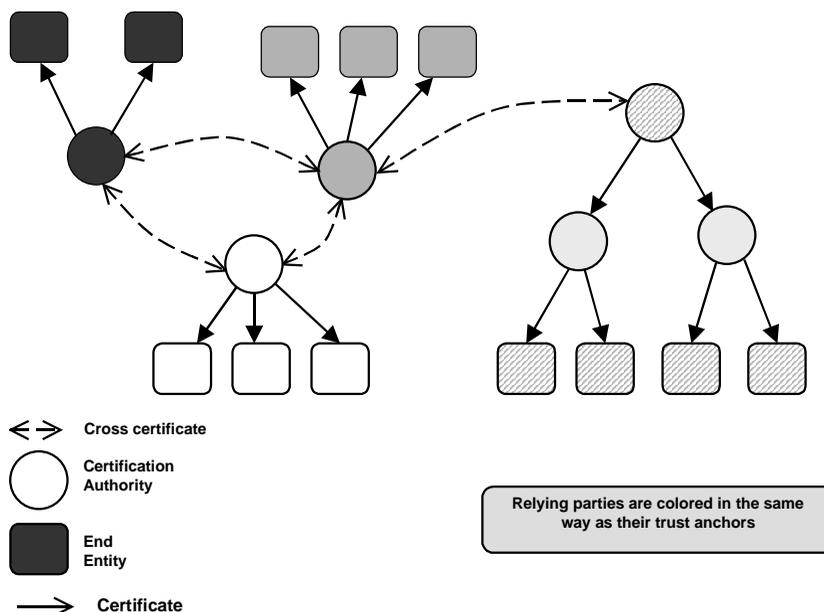
Compared to the hierarchical trust model, the network trust model might better represent peer-to-peer business relationships, where peers develop trust in each other through cross-certification instead of subordination. However, certification chain construction in a mesh PKI is more complex than in a hierarchical PKI because multiple paths can exist between a certificate and the relying party's trust anchor.

One of the major drawbacks of this model is that it requires a globally accessible directory to distribute cross-certificates to PKI clients. Without a global directory, a RP cannot generally find the cross-certificates necessary to chain the certificate of a communicating peer to the CA that it directly trusts, thus causing the certificate validation process to fail. Note that a networked peer typically sends only its own certificate and that of its CA when it communicates with another peer, which may have a different direct CA. For example, in the network of *Figure 9*, a user of CA1 submits its certificate and the self-signed CA1 certificate when communicating with a user of CA3. However, users of CA3 do not generally have local access to all the cross-certificates and must contact a global directory to build a proper certificate chain to their issuing CA. Care must be also taken because the certificates for bi-directional cross-certification are typically stored as certificate pairs in a directory attribute different from that used for individual certificates. The requirement for an online, globally accessible directory of cross-certificates introduces interoperability issues if a client cannot access the directory or if the directory does not contain an up-to-date list of cross-certificates. Furthermore, users of a mesh PKI may require installation of application plug-ins because current commercial secure applications do not support cross certification, as they do not know how to access a global directory and process cross-certificates. The distribution process of such plug-ins to all clients in a large network can become a deployment issue.

In contrast to a full mesh, a partial mesh can also be built in which not all CAs are cross-certified. In this case, the ability to perform any-to-any certificate validation is not guaranteed. Meshes do not enable general path construction to be accomplished unless the necessary cross-certificates have been pre-established between one or more pairs of CAs positioned along the path. Meshes have the important advantage of being deployable in "bottom-up" direction without dependence on the prior availability of a top-level root CA: each EE is configured only with the self-signed certificate of its own CA (i.e., the one that acted as issuer for the EE).

It is also possible that a hierarchical PKI becomes part of a mesh. This is sometime known as *hybrid trust model* (*Figure 10*). In this case, only the TLCA of the hierarchy needs to cross-certify with the other CAs of the mesh.

Figure 10: Hybrid



In general, all trust models that make use of cross-certification present the following drawbacks:

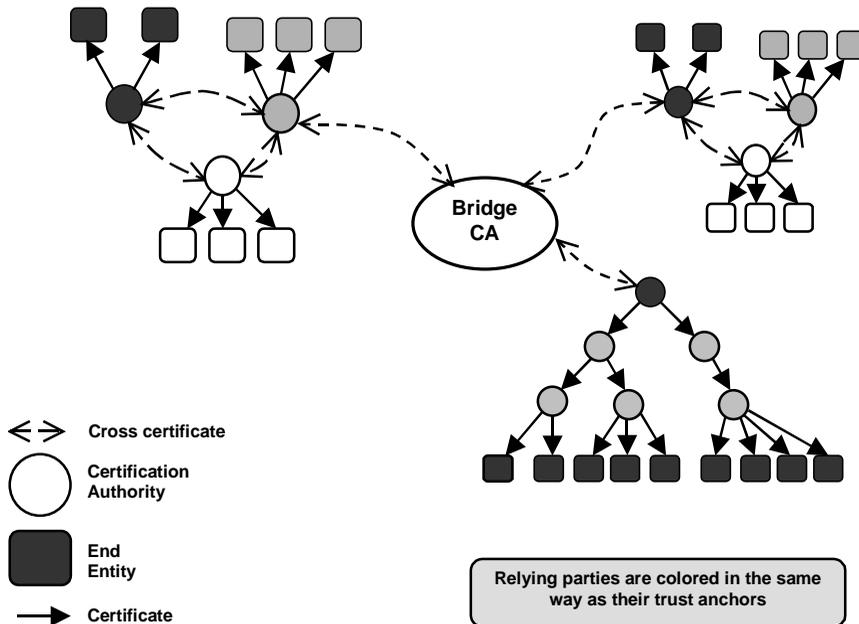
- increased complexity of the algorithms for certification path construction
- possible creation of unwanted trust paths through transitive trust
- decrease of the security level when a high-assurance PKI with restrictive operating policies is cross-certified with a PKI with less restrictive policies
- scarce or null support of cross certificates from commercial products.

The ICE-TEL web of hierarchies trust model (Chadwick et al., 1997) represents a form of hybrid model. Pairwise inter-hierarchy cross-certification, as in the hybrid model, serves well to link a small number of hierarchies. This model however does not scale well to larger numbers because it needs a number of cross-certificates equal to  $N^2 - N$ , where  $N$  is the number of hierarchies to be interconnected. Due to this problem, the Bridge CA model, described in the next section, was developed for the U.S. Federal PKI (Burr, 1998) and it reduces the number of needed cross certificates to  $N$ .

## Bridge CA

Another approach to the interconnection of PKIs through cross-certification is a hub-and-spoke configuration (Figure 11), also known as “bridge

Figure 11: Bridge



certification authority” (BCA). The main role of the BCA is to act as a central point to establish trust paths among different PKIs. In a bridge configuration, a “principal” CA (PCA) in each participating PKI cross-certifies with the central BCA, whose role is to provide cross-certificates rather than acting as the root of certification paths. Each CA can operate independently and under different certification policies. The BCA is not issuing certificates to the users, but only cross certificates to the principal CAs. Compared to the hybrid PKI model, where the number of cross certificates grows quadratically, in this environment the number of relationships grows only linearly with the number of PKIs.

Although this schema mitigates the political problem of the central trust point of hierarchical PKIs and reduces the number of cross certificates needed in generic mesh models, it still suffers from all the other problems of PKI models based on cross-certification. Furthermore, other political problems are raised by the operation of the BCA: the participating organizations need to agree with the BCA operators the certificate formats and the security policies in order to reduce the risk of unintended trust. This will most often be specified through certificate profiles as a part of the certificate policy and certification practices of the BCA and PCAs. This is discussed in the paper (Hesse & Lemire, 2002) that is based on the authors’ experience with a PKI interoperability tested built around a bridge certification authority that interconnects multiple PKIs based on CA products from several vendors (National Security Agency, 2001).

## CONCLUSION

This chapter illustrated various digital certificate generation and management schemas, by describing the different approaches proposed so far, together with their risks and vulnerabilities and the protections to be implemented. The reference to standards in this field is very relevant to let implementers achieve interoperability of certificate-based applications and systems (such as applications that manage electronic signatures or systems that use certificates in communication protocols). Attention has been paid to certificate extensions, as the ones defined by common profiles, because they are not just optional fields, but they are critical elements for more complex processing tasks such as certificate validation. Usage and application of the concepts and solutions presented in this chapter will provide a reference test bed to evaluate progresses in PKI and e-documents in the next few years.

## REFERENCES

- Arsenault, A., & Turner, S. (2002, July). *Internet X.509 Public Key Infrastructure: Roadmap*. IETF, Internet draft.
- Burr, W.E. (1998, September). *Public Key Infrastructure (PKI) technical specification: Part A – Technical concepts of operations*. Retrieved from the World Wide Web: <http://csrc.nist.gov/pki/twg/baseline/pkicon20b.pdf>.
- Chadwick, D.W., Young, A.J., & Kapidzic Cicovic, N. (1997, May/June). Merging and extending the PGP and PEM trust models: The ICE-TEL trust model. *IEEE Network*, 11 (3), 16-24.
- Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 6, 644-654.
- Ellison, C. (2002, April). Improvements on conventional PKI wisdom. In *Proceedings of 1st Annual PKI Research Workshop*, NIST (pp. 165-175). Retrieved from the World Wide Web: <http://www.cs.dartmouth.edu/~pki02/Ellison/paper.pdf>.
- Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., & Ylonen, T. (1999). *SPKI Certificate Theory*. IETF, RFC-2693.
- Ford, W., & Baum, M. S. (1997). *Secure electronic commerce*. Prentice Hall
- Freed, N., & Borenstein, N. (1996). *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. IETF, RFC-2045.
- Garfinkel, S. (1995). *PGP: Pretty good privacy*. O'Reilly & Associates.
- Hayes, J. M. (2001). Restricting access with certificate attributes in multiple root environments: A recipe for certificate masquerading. In *Proceedings of 17th Annual Computer Security Applications Conference*. Retrieved from the World Wide Web: <http://www.acsac.org/2001/papers/14.pdf>.

- Hesse, P. M., & Lemire, D.P. (2002). Managing interoperability in non-hierarchical Public Key Infrastructure. In *Proceedings of Network and Distributed System Security Symposium*. Retrieved from the World Wide Web: [www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/hesse.pdf](http://www.isoc.org/isoc/conferences/ndss/02/proceedings/papers/hesse.pdf).
- Housley R., & Polk, T. (2001). *Planning for PKI*. New York: John Wiley & Sons, Inc.
- Housley, R., Ford, W., Polk, W., & Solo, D. (1999). *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF, RFC-2459.
- Housley, R., Ford, W., Polk, W., & Solo, D. (2002). *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. IETF, RFC-3280.
- ITU-T Recommendation X.509 & ISO/IEC International Standard 9594-8. (2000). *Information technology - Open systems - The Directory: Public-key and attribute certificate frameworks*.
- Kohnfelder, L. M. (1978). *Towards a practical public-key cryptosystem* (Bachelor Thesis). Massachusetts Institute of Technology.
- Linn, J. (2000). *Trust models and management in Public-Key Infrastructures* (Technical report). RSA Data Security, Inc.
- Microsoft. (2001). *Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard*. Microsoft Security Bulletin MS01-017. Retrieved from the World Wide Web: <http://www.microsoft.com/TechNet/security/bulletin/MS01-017.asp>.
- Myers, M., Ankney, R., Malpani, A., Galperin, S., & Adams, C. (1999). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol*. OCSP, IETF, RFC-2560.
- National Security Agency. (2001). *Phase II Bridge Certification Authority Interoperability Demonstration Final Report*. Retrieved from the World Wide Web: <http://www.anassoc.com/Techno.htm>.
- Pfleeger, C.P. (1997). *Security in computing*. Prentice Hall.
- Ramsdell, B. (1999). *S/MIME Version 3 Message Specification*. IETF, RFC-2633.
- Rivest, R. L., & Lampson, B. (1996). *SDSI: A simple distributed security infrastructure*. Retrieved from the World Wide Web: <http://theory.lcs.mit.edu/~rivest/sdsi.html>.
- SET - Secure Electronic Transaction Specification*. (n.d.). Retrieved from the World Wide Web: [http://www.setco.org/set\\_specifications.html](http://www.setco.org/set_specifications.html).
- Webster's New Collegiate Dictionary*. (1980). Springfield, MA: G&C Merriam Company.

## Chapter III

# Smart Card Applications and Systems: Market Trend and Impact on Other Technological Developments

Gerald Maradan, STMicroelectronics, France

Pierre Cotte, STMicroelectronics, France

Thierry Fornas, STMicroelectronics, France

### ABSTRACT

*Securing data is becoming of the utmost strategic importance in today's digital environment. Open wide networks such as the Internet and inter-dependencies of modern systems have reshaped security requirements of smart card platforms. Smart card chips have been designed for 20 years to protect data and resist against attacks. Design mechanisms, cryptography, software implementation and certification process have all been introduced to provide efficient tamper resistant techniques against piracy. These techniques are re-used by a semiconductor industry demanding even more security. At the same time, smart card industry tries to address this demand and modify its positioning. This global convergence slightly impact new modern integrated systems.*

## INTRODUCTION

Securing data is becoming of the utmost strategic importance in today's digital environment. Open wide networks such as the Internet and interdependencies of modern systems have reshaped security requirements of smart card platforms.

In the 1980s, the first secure platforms were introduced by semiconductor industry to address smart card market needs. A smart card is a plastic card, embedding a silicon IC (Integrated Circuit). Today, smart cards are being used in a wide range of applications, from SIM (Subscriber Identity Module) cards for GSM (Global System for Mobile Communication), pre-paid memory cards for telephony and TV applications to transportation, communication, banking, health and identity cards. Since the late 1990s, GSM has been the market driver of smart card industry.

All these applications aim at protecting sensitive data. Therefore, security has shaped both hardware architecture and embedded software of smart cards. Beyond the technical features, a standard security evaluation methodology has been introduced, the Common Criteria (CC), released in 1996 in order to guarantee that the whole development process follows methodology rules. However, the environment is changing and the smart card market is at a watershed:

- Firstly, the communication revolution occurred: Internet, laptop computers and mobile phones led to the advent of the communication world. Thirty years ago, Marshall McLuhan made up the theory of “the Global Village” to describe his feeling that the world was getting everyday more interconnected and thus smaller and smaller (McLuhan, 1968). Indeed, today, anywhere in the world, people are able to communicate, to handle and exchange data. In this environment, securing data and communication becomes critical. Traditional software solutions to protect data turn out to be too weak to handle sensitive data (Pescatore, 2002). Technology providers propose alternatives and implement smart card techniques in modern platforms.
- Secondly, the digital convergence is on the track. Data transfers are moving from analog to numeric forms and digital home networks (DHN) are looming, inter-connecting into a network all devices in the house (set top boxes, televisions, DVD players, home servers, game consoles and computers) linked together through digital interfaces (USB – Universal Serial Bus, DVI – Digital Video Interface ...). The digital convergence will keep on. As a consequence, valuable content is now fully in a digital form and can be reproduced and redistributed wherever without any modification of its quality. Many industries, like music or movie ones, are at risk. Technology providers, coupled with standardization bodies try to find solutions for

content protection. We will describe in the following a representative case study based on SmartRight concept.

More generally in this chapter, we will provide ideas and trails to explain what makes the strengths of the smart card, exploring both technical security issues and market trends. At the same time, we will explore the state-of-the-art of hacking techniques and some tamper-resistance countermeasures that could redesign modern platforms. To conclude, we will try to catch a glimpse of smart card platforms' evolutions and their impacts on other modern integrated systems, focusing on content protection, e-commerce and introducing biometrics.

## Smart Card History

Security was not a major concern for the semiconductor industry two decades ago. The arrival of the first securechips closely relate to the advent of card technology.

In fact, bank credit cards have existed in the U.S. since the 1950s, storing information on the magnetic stripe embedded in the card. This technology is inexpensive but contains many drawbacks. The amount of data is still very small (only an identification number can be recorded) and security is not ensured since it is quite easy to duplicate the card with low cost machines. Information is written or read with techniques similar to those used to record or play an audio tape, and hackers can produce fake cards indistinguishable from real ones. Besides, in the initial scheme of U.S. credit cards, the time required to perform calculations was quite long, processors being deported partly to the reader, partly to a central calculator.

Taking into account these observations, Roland Moreno, in the 1970s, created the first smart card embedding a silicon chip. The idea of Roland Moreno was the following: this chip contains an identification number associated to a person and its available credit; the identification is performed through a PIN code (Personal Identification Number), typed by a user on a keyboard, and compared with a code embedded in the chip; access is then authorized or not; if the procedure fails, the card is deactivated (Svigales, 1987). The concept has been accepted immediately, but it was difficult to produce in an industrialized process. Indeed, silicon chips are complex and flimsy, traditionally packaged in solid structures. Smart cards represented a challenging opportunity, since they consist in packaging a thin chip in a plastic card of less than 1.5 mm<sup>2</sup>.

The first application of the smart card was the prepaid memory cards used for the payment of telephonic communications. The card embedded a simple non-volatile memory chip (a memory that could retain data without power supply) with an additional security circuitry to prevent people from modifying stored data. The phone card application exploded in the 1980s, first in the French

market then in the German one. In 1996, the overall number of prepaid phone cards reached 2 billion cards.

In order to cover new applications and really apply Moreno's ideas, smart card chips needed to combine both memories' capabilities and processing power. Therefore, smart card chips became secure micro-controllers with embedded memories storing both the Operating System and the application code, and with the CPU (Central Processing Unit) processing the code. Smart card platforms became a combination of a hardware micro-controller and software codes. Specific security features were implemented in order to hinder a fraudulent use of the smart card, like sensors, implemented to control external conditions of use of the card (check of external power supply and frequency).

In the end of the 1980s, French banks decided to adopt smart cards. Bankcards rely mostly on the well nigh impossibility to hack security mechanisms. As the smart card market grew, security became the key issue and the more complex task to tackle. As processing power introduced the possibility to perform mathematical calculations, smart card chips started to introduce the use of cryptography in semiconductor industry. The fraud was divided by 10 in three years compared with fraud when magnetic cards were used. As a consequence, VISA International and Europay definitely adopted the technology.

This success led to a wider use of smart cards in other applications like health care, pay TV and mobile communications. Microprocessor cards kept on gaining global acceptance throughout the world. In the late 1990s, the GSM SIM cards became the driving segment and smart cards became a strategic market in the semiconductor industry (Rankl, 1999).

## **Security Policy and Certification Scheme**

As the produced smart card volume increased, several new players decided to enter the market. Among the multiplicity of newcomers, it became more and more difficult for customers to feel confident with the security. Smart card key players endeavored to find solutions and finally proposed that independent security experts examine the cards, following security evaluation schemes. These schemes aim at covering IT (Information Technology) systems, hardware and software components. They are based on impartial evaluation, performed by independent experts and governed by national agencies to ensure conformity with the standards.

Formal Security evaluation techniques have existed for several decades but were confined to governmental applications.

Historically, the first significant scheme, called the TCSEC (Trusted Computer System Evaluation Criteria) has been introduced by the U.S. Department of Defense (DoD) for product evaluation since the early 1980s. The European Commission created the European ITSEC (Information Technology Security Evaluation Criteria) and adopted the version 1.2 in 1991. The ITSEC

aimed at covering both commercial and military applications. It was the first scheme used to assess the security of a smart card application. STMicroelectronics and Bull sponsored this evaluation, covering both the hardware chip and the software application. Several smart card applications, including bankcards, pay TV and health cards followed this model and achieved certification using this scheme.

At this time, several schemes still co-existed. Indeed, in 1993, the Canadian government issued the CTCPEC (Canadian Trusted Computer Product Evaluation). North American and European concepts decided to merge in order to provide criteria for the international community. The ISO started to work in 1990 and the v1.0 of a global standard and the Common Criteria (CC) was released in 1996. Finally, the Common Criteria are the result of extensive international efforts to align the source criteria from Canada (CTCPEC), Europe (ITSEC) and the U.S. (TCSEC). Now, it is widely used and corresponds to an ISO standard (ISO15408) ([commoncriteria.org](http://commoncriteria.org), 1999).

Let us introduce the basic notions of the Common Criteria process. The CC process starts with a security analysis defining:

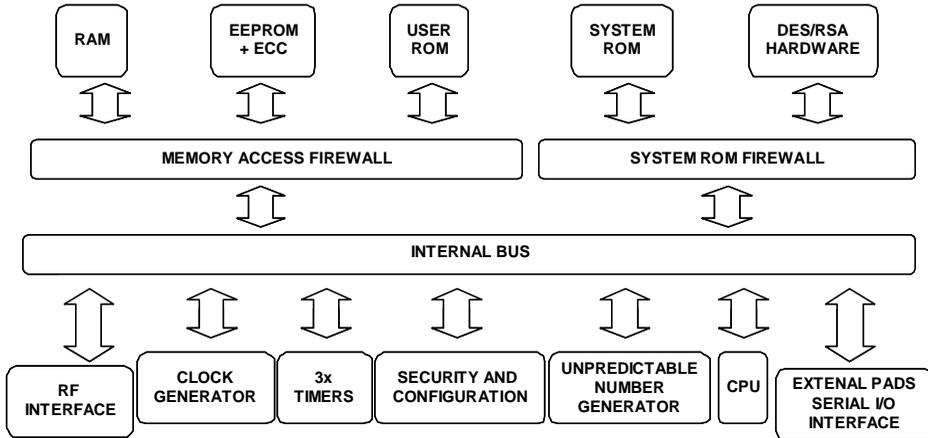
- The Target Of Evaluation (TOE), which is a part of the product or system subject to the evaluation
- The assets to be protected
- The threats to be countered
- The security objectives
- The Security Functional Requirements, the security functions to be implemented, and the Security Assurances requirements, the evidences that security functions have been correctly implemented

The result of this analysis should be written in a Security Target (ST). The ST can conform to a Protection Profile (PP). The PP is a subset of the Common Criteria and gathers standardized sets of security requirements for products or systems that meet similar security needs. A Protection Profile has been developed to address the major security requirements for Smart Card Integrated Circuits (Eurosmart, 2003).

The evaluator assesses then both the effectiveness and the correctness of the security mechanisms. It analyses the procedures, the methodology and performs security attacks on the TOE in order to establish a level of confidence of the system. A predefined CC scale, called the “Evaluation Assurance Levels” (EALs) ranks the level of confidence from the level EAL1 to the level EAL7.

The level of confidence the smart card industry achieves is generally the EAL4+ (or EAL augmented) level. For more information, refer to CC documents in [commoncriteria.org](http://commoncriteria.org) (1999) and Smart Card Protection Profile in Eurosmart (2003).

Figure 1: 8-bit secure micro controller architecture



Security of a platform is always a combination of several elements among which are the strengths of its technical features (hardware and software) and the security of both development and production environment. Formal security evaluation techniques provide an efficient method to control and measure broadly different security parameters.

## Smart Card Architecture Overview

The first IC embedded in a plastic card was based on a memory plus security logic added to prevent easy fraudulent modifications of its content. This architecture still exists, and is mainly used as prepaid memory cards for telephone applications.

The first memory type, initially introduced, used an EPROM (Electrical Programmable Read Only Memory), a memory requiring a high programming voltage supply. A new sort of non-volatile memory, the EEPROM (Erasable Electrical Programmable Read Only Memory) was then adopted, and did not require any additional power supply.

Let us analyse in more detail the structural design of the second type of smart card IC, based on a secure micro-controller (Rankl, 1999).

### Secure Micro Controller Cores

Two main architectures split the smart card market. The first one, the most widely used, is built around an 8-bit CPU based on the CISC (Complex Instruction Set Computer) architecture (see *Figure 1*). The core fetches an instruction from the memory, decodes it and performs the required operation. Then, it fetches the next instruction. The instruction set is very wide and

generally compatible with Motorola 6805 or Intel 8051 cores. Other silicon providers generally have added other instructions.

The second type of architecture is built around a 32-bit core, based on the RISC (Reduced Instruction Set Computer) architecture (see *Figure 2*). 32-bit secure micro-controllers provide enhanced processing capabilities. They appeared recently in the smart card market with the introduction of multi-application chips. The RISC architecture has a reduced instruction set. Operation management differs from CISC in that it is usually structured around a pipeline (Heath, 1995).

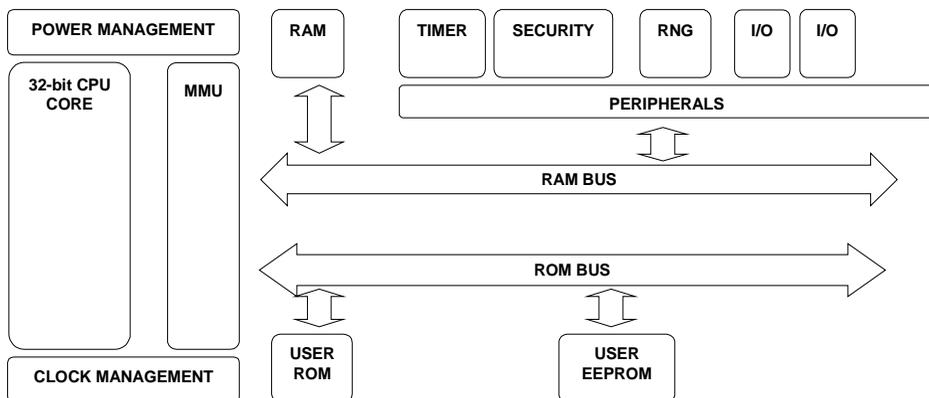
Both architectures present the same level of security, security techniques being relatively independent from the core.

Around the core, modern secure micro-controllers, whatever the CPU, host several macro cells to compose the chip among which are:

- Memories: RAM, ROM, EEPROM or Flash
- Firewalls: for memories or dedicated macro cells
- Timers and clock management module
- Random number generators
- Sensors
- Cryptographic hardware macro cells

Two I/O (Input/Output) pins are dedicated to ensure communication with an external reader, following the ISO-7816 standard (ISO7816-3, 1997). Contact-less cards take a significant market share in smart card industry. They do not require any electrical connection between the card and the reader, and an antenna hidden in the card allows transferring data within short distances.

*Figure 2: 32-bit secure micro controller architecture*



Several Radio Frequency solutions exist, but the chip architecture itself is based on the same architecture presented before.

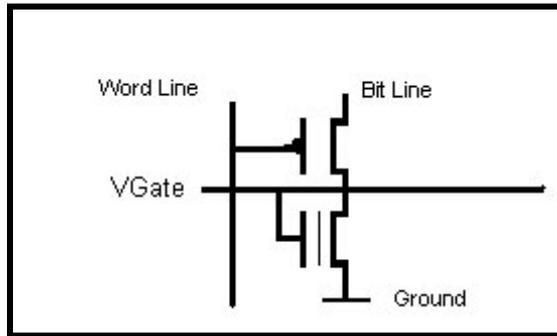
### *Memories*

To store the program code and data, various types of memories are implemented. Memories used are the RAM, the ROM and the EEPROM. Their sizes depend strongly on the application and on how compact the code is. The size is an important parameter, which defines the size of the chip and consequently its price.

Generally, the ROM is the most compact memory. For the sake of comparison, one bit of EEPROM is 1.5 times bigger than one bit of ROM. One bit of RAM is four times bigger than one bit of ROM (Rankl, 1999).

Let us analyse deeper technical features of memories:

- The ROM can only be read, but cannot be written. The data are hard-wired on the chip and cannot be modified once the chip is manufactured. As a consequence, no voltage is necessary to maintain the data.
- On the contrary, the RAM needs power supply to hold the data. The RAM is used to store variables of a program. Smart card industry uses SRAM (Static Random Access Memory). Six MOSFET (Metal Oxide Semiconductor Field-Effect Transistor) constitutes the SRAM. This type of memory acts as a bi-stable multi-vibrator and needs a continuous current to hold the data.
- The EEPROM is a NVM (Non Volatile Memory), i.e., a data can be maintained even with the absence of the power supply. Thus, each bit of the EEPROM can be modified during the life of the chip. This type of memory is much more complex than ROM or RAM memories. Two MOSFET compose the architecture of an EEPROM bit (see *Figure 3*). One transistor is used to select the bit. The second one is the storage transistor. It has a floating gate, which captures or discharges electrons on this gate. Electrons are able to go through the energy barrier between the oxide using the Fowler-Nordheim tunnelling effect, and depending on the tensions applied on the transistor. The state of the transistor will modify its threshold tension. In the programmed state, the transistor is blocked. No current can flow between its source and its drain. In the erase state, the current can flow between them. Finally, the data values are determined by the charge on the floating gates. Due to the current leakage of this gate, most silicon providers guarantee 10 years of retention of data or a capability to sustain 500, 000 write/erase cycles.
- Flash: some products contain Flash memories to replace ROM or EEPROM memories. A Flash memory is a non-volatile memory (NVM) with one transistor per cell. Like the EEPROM cell, the transistor is a MOSFET with

*Figure 3: EEPROM cell*

a floating gate with a thicker tunnel. The electrical erase is performed by Fowler-Nordheim tunnelling. Program is performed by hot electron transfer through the oxide (Rankl, 1999).

Firewalls are mainly designed to protect memory access. They can also be used to protect dedicated functions like cryptographic macro cells. They are hard-wired. Memories' firewalls allow partitioning of the memories. The Operating System maker, the one who develops the embedded software, defines the access rights from a partitioned memory to another one. Besides, he/she can manage the executability of this partition. A wide range of options is possible. Firewalls represent a strong security feature, hindering unauthorized downloads of software and faked code execution. Interrupt management capabilities complete the structure by interrupting the execution of the code and acting according to the security policy of the Operating System maker.

### *Timers and Clock Management Module*

Timers are generally based on 8- or 32-bit counters. They provide a time base with precise delay, necessary for the Operating System management and useful to implement security functions.

The clock management module is a clock generator providing the ability for the Operating System to select various clock configurations. This module, like most macro cells in the chip, is selectable by software. The circuit is driven by an internal clock, whose frequency is instable. It allows the OS to switch from one clock to another, providing an efficient tool against fraudulent program analysis. Internal clocks are generally used during code execution. The external clock is selected only when the application requires an exchange of data with an external reader. Internal clock frequencies usually range between 1 and 40 MHz.

### *RNG (Random Number Generators)*

The RNG (Random Number Generators) are widely used in smart card chips. Many mathematical calculations, but also hardware security mechanisms, rely on them. They were introduced at the beginning of smart card history mainly to generate cryptographic keys. Progressively, they have covered other functionalities, such as encryption of data stored in memories or internal wires encryptions. It is essential that the RNG be truly random in order to reduce the possibility for hackers to re-calculate numbers by finding out their mathematical law. RNG are designed using FSR (Feedback Shift Registers), synchronised by a clock and modified by a polynomial function. They are generally coupled with an analog source, in order to enhance the unpredictability of the RNG.

### *Sensors*

Under the generic term “security functions,” we mean to refer principally to the sensors implemented in secure micro-controllers. Two ranges of sensors are commonly implemented:

- The “integrity sensors” protect sensible data in the chip against physical intrusion. Indeed, the smart card chip is covered, at the upper layer, by a grid of sensors detecting physical modifications of this layer. Thus, invasive attacks are considered as an intrusion and the sensors triggers a hardware mechanism, immediately destroying all data contained in RAM and EEPROM memories. The card becomes unusable.
- The “environment sensors” control the external parameters of the chip, like voltage or frequency. External pads are monitored during the execution of the application. When these parameters are outside the range of the specifications, the sensor notifies the Operating System. The Operating System then acts according to its security policy. Indeed, hackers may try to modify external operating conditions in order to create a malfunctioning of the chip in order to get information about its behaviour.

### *Cryptography on Smart Cards*

Cryptography techniques are widely implemented in smart card applications. Algorithms and protocols ensure confidentiality of data, authentication, integrity and non-repudiation. We will not explain the concepts of cryptography but just provide fundamentals, while we reference details in Schneier (1997).

A cryptographic algorithm basically encrypts a text or decrypts a ciphered text. A key (a certain amount of bits) is required for encryption or decryption. There are two classes of key-based encryption algorithms: symmetric (or secret-key) and asymmetric (or public-key) algorithms. They act as follows:

- Symmetric algorithms use the same key both for encryption and decryption, whereas asymmetric algorithms use a different key for encryption and decryption. The decryption key cannot be derived from the encryption key.
- The asymmetric algorithms are used for key agreement, digital signature, and encryption. They are based on hard mathematical problems while symmetric cryptography is generally based on ciphering-block algorithms.
- Various cryptographic protocols mix the use of both classes of algorithms. Private-key algorithms are around one thousand times faster than public-key ones.

Depending on the required applications, some smart card chips may embed hardware macro cells dedicated to cryptography. Some high-end secure micro-controllers contain dedicated instructions and cryptographic capabilities inside the core itself. No additional macro cell needs to be implemented, hence making the chip more compact and computations more efficient. Smart card applications handle sensitive data: OS developers have naturally started to introduce cryptography techniques as soon as secure micro-controllers appeared. The first implementations were purely software. Hardware implementation became necessary to strengthen security.

Indeed, the basic intention of smart card applications is to protect sensitive data, such as keys. In order to manipulate these keys, to defend them, and to reassign them to the external world, a certain amount of processing power is required. With the increase of piracy, security needs to be stronger and data management has become more complex. The time necessary to perform such calculations makes sometimes crucial the introduction of dedicated blocks. Some computations, like modular arithmetic or multiplication over great numbers of bits, are not included in CPU instructions. Two main hardware cryptographic macro cells exist: hardware macro cells supporting private key cryptography and crypto-processors. They are now described separately.

### *Private Key Cryptography*

A hardware DES (Data Encryption Standard) supports the DES algorithm (NIST, 1999). This algorithm is a private-key algorithm based on a block ciphering system, which input is a 64-bit block of text and uses a 64-bit key to perform encryption. The result is a cipher text (or encrypted text) of 64 bits.

The DES was created in the beginning of the 1970s, and was adopted by the NSA (National Security Agency) in 1976. Despite years of cryptanalysis, DES has demonstrated its strength. Nevertheless, with the increase of processing power in these last years, it was possible to try all combinations of keys and retrieve it. The triple DES was introduced, performing three successive DES or DES inverse, and requiring two keys of 64 bits. In the 2001, a new algorithm, the AES (Advanced Encryption Standard) was adopted as the new standard for the private-key algorithms by the NSA (Daemen, 1998).

At the beginning of the 2000s, a DES operation could be executed within 20  $\mu$ s with a dedicated hardware, dividing by more than 200 the time required by a software implementation on a similar platform. The gain is tremendous. A dedicated design allows both pipelined implementations and optimised computations on 64-bit registers. Moreover, the CPU could execute parallel operations; sensitive DES calculations could be performed inside the macro cell. A sequential analysis of the CPU becomes trickier.

It is likely that AES dedicated hardware will be implemented in the near future. Software AES has not been introduced yet on smart card platforms.

### *Crypto-Processors*

Crypto-processors are designed to support asymmetric cryptography. They are arithmetic units performing operations over an important amount of bits (generally between 1024 and 2048 bits), such as exponentiations and modular operations. Public key algorithms necessitate a large amount of time compared with private-key ones (Schneier, 1997).

W. Diffie & M.E. Hellman introduced the concept of public-key in 1976. Their innovative idea relies on the fact that it was feasible to make up cryptography algorithms based on two different keys: one for encryption (the public key) and the other one for decryption (the private key). Anyone could then encrypt a message with the public key, but only the owner of the private key could decrypt it. This is a strong advantage over symmetric algorithms, whose well-known problem is to securely transmit the key to the receiver. Public-key cryptography overwhelms the problem.

In 1978, The RSA (Rivest Shamir Adleman) algorithm was presented. This is the well known and most used of asymmetric algorithms. It is based on the IFP (Integer Factorisation Problem), i.e., on the fact that given  $n$  and a product of  $p$  and  $q$ , which are primes, it is difficult to find  $p$  and  $q$  (Schneier, 1997).

Crypto-processors support RSA implementations, but also hash functions, like the Secure Hashed Algorithm (SHA) (NIST, 1994). An RSA encryption process is time consuming. Software implementations are nearly impossible to achieve, since several seconds would be necessary on a micro-controller. Hardware support is hence essential. Today, RSA calculation with a dedicated crypto-processor is performed within 300 ms.

A new type of cryptosystems, based on the Elliptic Curves Cryptography (ECC) is being introduced in the smart card industry. V. Miller & N. Koblitz first proposed these cryptosystems in the mid-1980s (Miller, 1986). In fact, Elliptic Curves cryptosystems can be classified into two classes: whether they are similar to the RSA systems or to the discrete logarithm based system. The techniques for computing elliptic curve discrete logarithms are more efficient than those used for classical discrete logarithms. Therefore, shorter key sizes are required to achieve the same level of security of simple public-key algorithms, representing a potential to build fast and secure algorithms (Blake, 1999).

## MAIN THRUST

In this section, we want to provide an overview of the main trends of the smart card market and the applications of smart cards within the globalization of modern systems. The evolution implies both a modification of the business model and the introduction of new software architectures such as 32-bit RISC MCU and the popular usage of Java language. Moreover, the convergence of applications (mobile phone e-payment and transport) are likely to merge and thus change the whole value chain for either operators' smart card vendors or chip suppliers. We will provide also an overview of state-of-the-art of attacks against modern platforms and tamper-resistance techniques to fight against hacking.

### Smart Card Market Trend

The first significant breakthrough started with prepaid memory cards for French and German telephony market in the mid 1980s. Smart cards based on micro-controllers found their first high volume opportunity by the end of the 1980s with the French banking system. However, the boom really arrived when the technology was incorporated into SIM cards for GSM mobile phones. Since the early 1990s, smart cards have become a very useful item for consumption throughout all of Europe, shortly followed by Asian countries (Rankl, 1999).

Despite its immediate acceptance in Europe, the U.S. smart card market has been very slow in taking off. The main reason is related to banking fraud, which was lower than Europe. Furthermore, concerning mobile telephony, the U.S. standard (CDMA) is based on a low-level pure software security instead of using a SIM card, which is the standard for handsets in Europe and Asia.

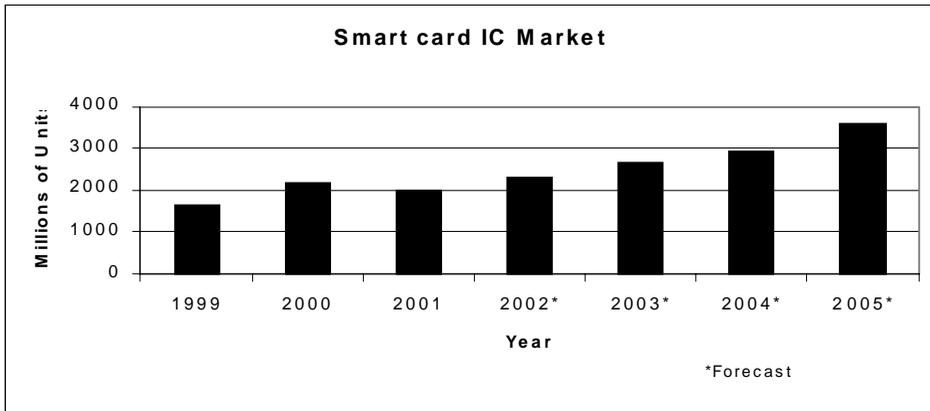
Even with few running applications, there is growing activity in smart cards in the U.S. with the high demand of security that appeared after the events of September 11, 2001. Indeed, government wireless operators, banking, and other corporations are requiring higher secure systems for both online (respectively off-line) identification and authentication, to prevent easy hacking from traditional magnetic stripe cards.

Before 2001, the industry of smart cards had enjoyed healthy growth rates in the range of 30%. Nevertheless, 2001 was certainly the worst year for this industry, with an average slump of more than 15% in term of revenue, especially for chip prices. Consequently, although more than two billion cards were issued in 2002, IC revenues have significantly fallen.

Signs of revival showed up in 2002, especially in term of volume delivery, with a growth of 11% up to 13% estimated by the main actors.

From the chip manufacturers' point of view, the smart card IC market also declined in 2001 with a high effect due to stock at operators and vendors side. However the IC demand is showing an increasing average of 18% each year, from 2.2 billion in 2001 to a forecast close to 5 billion in 2006 (see *Figure 4*).

Figure 4: Smart card chips shipment evolution forecast



Europe has dominated the industry both in term of production and usage. While Europe still covers more than 50% of the market, the trend shows clearly a shift towards the Asian area, with more than 30% compared with the current situation. Moreover, as explained above, the rise of both the U.S. and Chinese markets will certainly balance the predominance of Europe in the coming years.

European companies also dominate the production of smart cards. From “end-product” side, the two main actors, Schlumberger and Gemplus, represented more than 55% of the revenues in 2002. If we add the two other major players, Oberthur and G&D, the share will represent more than 80% of the market. Nevertheless, new Asian smart card manufacturers are likely to become major players in the coming years.

For ICs market, the three top companies in 2002, Infineon, Philips and STMicroelectronics, are also European ones and produce more than 75% of the worldwide smart card ICs. Here again, lots of newcomers, such as Hitachi, Atmel or Samsung, are becoming very active (see *Figure 5*) (Hirst 2002; Rouviere, 2002; eurosmart.com).

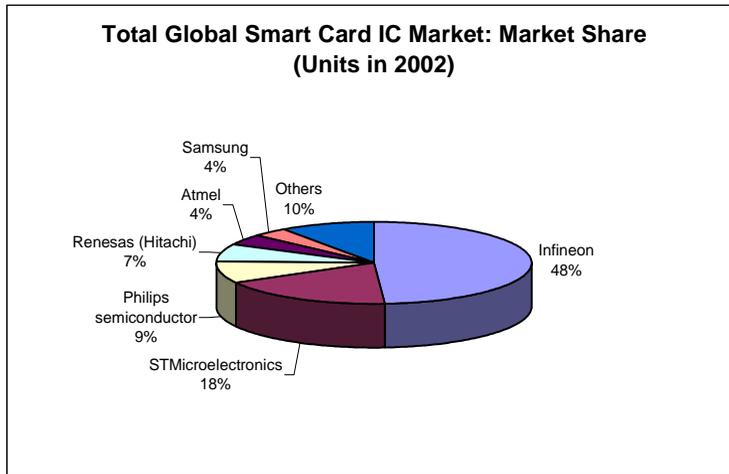
### *Value Chain*

The value chain of the smart card industry is moving. The historic chain is split as follows:

*Chip supplier <=> Smart card vendors <=> Operators as Card Issuers Chain*

But from some years ago, smart card manufacturers have a strong willingness to move up the value chain to a more service added value in term of revenues. Indeed, the market is becoming a commodity market, with power

Figure 5: Total global smart card IC market: Market share in units for 2002 (Source: iMarket McCulligan Analysis, 2003)



pushed by smart card vendors. This push was also strong from software houses, terminal system integrators, and other companies.

The value chain is now becoming fragmented and smart card manufacturers are losing a part of their power. New entrants (such as Aspects Software, Datacard, IBM, Trusted Logic and Goldkey) have entered the market and are creating quite a stir. This is one reason why smart card vendors now have to face two axes of competition which could squeeze them: on the one hand, new entrants from both Asia and Europe, and on the other hand, a dramatic price decrease from smart card buyers. European manufacturers have mainly focused on a volume market to secure their market share. With this evolution and the help of new entrants, chip suppliers could gain power to push complete solution offers under their own brand. Certainly, this is a big move and chip suppliers are strong enough to face the stage within a volume price dominant market. One of the examples is the acquisition of the Belgium Banking Card Issuer, Proton World, by the chip supplier STMicroelectronics. Particularly in the banking segment, STMicroelectronics is able not only to provide part of global solutions but also to propose exhaustive partnerships.

The future of the market and its value chain is going in the way of such interpenetration of value chain actors (Ubhey, 2003).

## **Distribution and Market Trends of Smart Card Applications**

Applications using smart cards are numerous and a lot of new ones are currently emerging. On one hand, mass-market applications are coming from

banking and GSM world usage. On the other hand, applications such as Pay TV, healthcare, transportation, PC network authentications, and government identification cards happen to be growing popular throughout the world (see *Figure 6*).

### *Mobile Phones and SIM Cards*

Most of smart card market revenue has come from the SIM card. The latter provides handset security (with the use of PIN code) and subscription information to users of mobile phones. In 2002, SIM cards represent more than 50% of card production with around 420 million units. Now, this segment is beginning to be saturated.

The overall growth of new customers is stagnating despite new opportunities like the ones emerging in China. Therefore, the market turns into a replacement market. For the first time, in 2001, this market, made of SIM issued to customers who switch from an operator to another, seeking better deals or wanting to upgrade their existing services, is equal to the number of SIM sold for new mobile customers. Replacement SIM will significantly outstrip SIM for new customers, especially in Western Europe, since operators are trying to increase their revenues per customer. Card vendors are focusing on applications and services in order to convince end-customers to upgrade their cards.

Even if SIM cards are based on micro-controllers, the EEPROM memory size is one of the key elements for GSM market. As a consequence, the importance for high memory products is increasing from 32 Kbytes of EEPROM to 64Kbytes and even up to 128Kbytes. This memory inflation closely relates to data. An example is the phone directory stored in the SIM card. New markets, like the Chinese market, are focusing on low-cost SIM cards with a use of EEPROM between 8Kbytes and 16Kbytes.

A second key for GSM market is the ability for operators to buy their cards from multiple suppliers and to expect that new applications will run on several cards. Java Cards have been designed to provide such interoperability. Java use is becoming more and more popular. Despite infant problems, interoperability is now a reality for many distributed systems in various application areas, such as public administrations, corporate information systems, industrial supply chains, and so on.

In fact, the main constraint is coming from TLC operators, who want to ensure new revenue generating services. Mobile commerce is one of the future services that sound promising in the coming years. To conclude, we can say that if smart card providers will not be able convince the industry that a SIM brings about a strong added value, TLC operators will consider SIM card business as a commodity.

### *Banking*

Banking is the first historical business area for smart cards. It represents a sizable portion of the whole market. France has experienced a significant fraud

cutback of 90% for financial transactions within a few years. And, almost all of the remaining 10% has come from transactions made outside France and using the magnetic strip card.

Banking cards have a brisk growth forecast of more than 30% in the coming years. The main reasons are both the EMV (Europay Mastercard Visa) migration and the product differentiation for banks.

EMV standard was created allowing smart cards from all the major brands to be accepted at all payment terminals. That is why Europay, Mastercard and Visa International have joined their forces to create the standard. Other competitors, such as American Express, have also adopted this norm. Card associations are pressing member banks in Europe to move to smart cards by 2005.

For other areas, like in the U.S. and in Asia, a lot of credit card issuers are coming on the market with sophisticated smart cards, which will stand out in their highly competitive market and also will allow acquiring new customers and retaining them.

One of the future popular applications is the Electronic Purse (named e-purse). This emerging module is intended to replace cash payments. The E-purse functionality is already in use in some western European countries, like Belgium.

### *Pay TV*

This is a very fast-growing market. Smart cards used in Pay TV serve as a removable security element and provide subscription information for its user. The key point is strongly linked to security to reduce hacking of broadcast TV diffusion.

More generally, Conditional Access (CA) technologies are of the utmost strategic importance in the digital TV environment, as key enablers of pay-TV business models. The CA market is promising to grow substantially, since it follows the growth of Set Top Box market, now the fastest growing consumer application in the marketplace (CAGR +25%, 2002 estimation). Furthermore, it is likely to change substantially over the coming years. The arrival of Personal Video Recorder (PVR) in the Set Top Boxes, the Video on Demand (VoD) through ADSL, Copy Protection, and Digital Rights Management (DRM) in home networks put CA actors under pressure to provide a broader range of technologies and services. The market could boom from 2005 on with new applications.

Indeed, the CA market is evolving with the convergence of the digital consumer equipment in the home (DTV, DVD recorders, VDSL modem...) and the security policies, especially to protect contents, mainly video from MPA (Motion Picture Association) and partially Audio (with MP3 popular usage), will certainly rely on evolution of CA systems.

### *Transport*

Public transport ticket payment has been identified as a major potential application in the development of the smart card market a long time before it was technically or commercially feasible. One key requirement is contact-less capability, because any scheme that requires passengers to insert a card into a terminal would not offer any advantage over traditional coin-operated ticket machines. In order to ensure smart card success in this market, the entire payment transaction must be completed during a small fraction of a second needed for the passenger to walk through the payment point without stopping.

Transportation operators have launched various experiments to evaluate all technological impacts of smart card usage. Major rollouts or launches are planned in 2003 in London, Paris, Tokyo and Singapore. Almost every major city is at least considering the usage of chip card in mass transit.

Most of the projects have two ways to use chip cards. The first uses just a memory chip as single trip ticket and the second uses a microprocessor card for multi applications. Why such an approach? Although essential, contact-less operation is not enough. Nobody wants to see his/her wallets filled with a proliferation of dedicated cards. In practice, it means that, for example, paying for a metro ticket becomes a transaction where the ticket payment is made by debiting an e-purse account on a same smart card which could also be used to pay newspaper or a piece of cake.

### *Specific Secure Applications: E-Government*

The events of September 11, 2001 triggered exceptional interest in usage of smart cards. It has boosted a lot of pilot projects in the U.S. in the fight against terrorism. A good example is the U.S. Department of Defense (DoD), which is in process to release more than 4 million smart cards to military, civilian personnel and outside contractors. Those smart cards are used for Identification and Authentication allover DoD agencies.

ID (identity) applications will become the dynamic application, but it will probably take some time to come up, simply because government programs are developing gradually.

Moreover, the usage of biometrics is also linked to such applications in order to strengthen security. It is particularly true to boost airport security. Pilots at Amsterdam's Schilphol Airport are running with passenger authentication after inserting a smart card and verifying the identity through iris recognition. This process firstly spectacularly reinforces security process at gate entrance, and secondly, it should reduce the waiting time for boarding.

Nowadays, national ID cards are on projects all over the world, mainly in Western Europe and in dedicated Asian countries. The same process is ongoing for driving licenses.

In the same way, health cards are already used in Germany and France. The main advantage of such cards is the improvement of administrative procedure time schedules and, thus, a drastic impact on cost reduction for all paper documents linked to the process flow. Two kinds of cards coexist: one as social security ID and another one for health professionals.

### *Multi-Application Cards*

A multi-application card holds several kinds of applications in one physical card. For example, a transportation card can be combined with an e-purse and an ID company badge. Products are already available, since new 32-bit chips were designed to target the market. Nevertheless, there are still a lot of legal problems linked to the operator's responsibility before a mass-market launch. In other words, this implies strong agreements, for example, between telecommunication and banking operators, to interoperate.

Anyway, the market seems to be ready to move towards new business models and some very interesting trials are on the field, such as in transport systems.

### *Pay Phones*

Pay Phone cards are basic cards with only a secure memory inside. They are used to replace coin and reduced theft fraud in a public area. It is a very high volume market with low prices. The market is becoming mature and prices are slowing down. In some countries such as Mexico and China, this market is strongly soaring.

### *Open Network Security*

One of the big challenges of the smart card industry consists of a move from dedicated networks (such as banking networks or GSM) to an open world (based on Internet). In this environment, smart card connections, standardized according to ISO 7816, are now a bottleneck both for communication speed and bandwidth.

Recently, dedicated smart card products appeared on the market integrating an USB (Universal Serial Bus) port. Such a module could either work directly on USB network within a computer or as a classical smart card ISO connection.

The popular usage of RF (Radio Frequency) networks is also a challenge for future generations of smart cards.

Historically, European actors mainly drive the market. Asia is progressively involved in the business and the U.S. is waking up slowly but strongly. After a decrease in revenue, especially for ICs suppliers, the market is starting again with a strong war on prices.

The business model is evolving; on one hand with new type of applications like multi-application cards that imply partnerships between operators from

different segments (banking & telecommunication, for example) and, on the other hand, with the structure of the market chain, which is also changing with a more added value put on software and services around the card itself. This is why some IC manufacturers are coming from the hardware world to offer to more advanced “added solution” offers.

## **Operating System Evolution Trend**

A smart card Operating System is constituted by the embedded programs, which together with the chip features, controls and supervises program processes. Evolution of the Operating Systems in the last 10 years has been characterized by a profound change: from proprietary monolithic Operating System, the tendency has moved towards layered Operating System, with more and more refinements.

At the beginning of the 1990s, smart card Operating Systems were very few. This was mainly due to the limited memory capacities of the existing chips at that time. Operating Systems were monolithic: they were structured as a collection of Assembler or C routines, compiled and implemented on ROM. Although the code was characterized by low-level complexity, its monolithic structure required deep know-how to perform any functional modifications, and only at a considerable expense.

Over the past few years, advances in chip technology and chip software technology have been phenomenal. The capacities of chips ROM and EEPROM have become tremendous, and, in the meantime, advanced co-processors, performing fast computing operations, have come out. Smart card ICs can also run portable Operating Systems, much like Windows, which support high-level application programming languages (e.g., Java, C++, Visual C/Basic). These advanced capability features are driving factors for smart card market acceptance: they allow a simplification and a standardization of the application development phase, and therefore imply reduced costs.

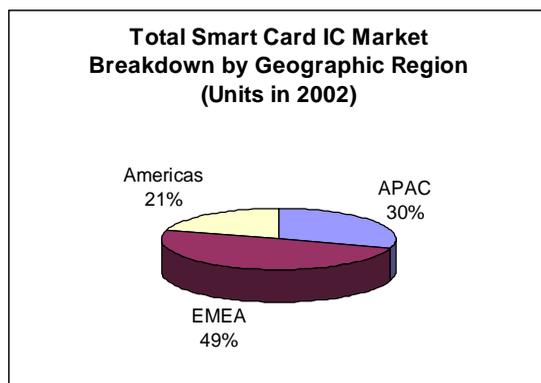
### *Business Requirements*

New business requirements for smart cards stand in providing the most customized services on the cards in the most cost-effective manner. One of the most important requirements for an issuer turns out to be that the same card must support applications from different application providers. This new concept is called multi-application smart card.

What are the characteristics of an “open” multi-application smart card? Dictated by the market evolution, characteristics of an “open” multi-application smart card, which support industry standards for both hardware and software, can be drawn as followed.

True “open” multi-application smart cards will have the following characteristics:

Figure 6: Total smart card IC market distribution (ST internal source)



- They will run a non-proprietary operating system widely implemented and supported.
- No single vendor will specify the standards for the operating system and the card's use.
- The cards will support a high-level application programming language (e.g., Java, C++) so issuers can supply and support their own applications as well as applications from many other vendors.
- Applications can be written and will operate on different vendors' multi-application smart cards with the same API (Application Programming Interface).

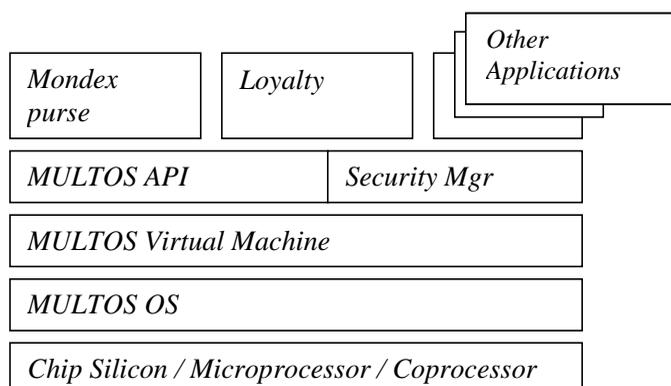
These advancements have had a dramatic effect on the smart card industry and the applications that run on smart cards. Now the challenge is to choose the correct open multi-application smart card technology and chip technology to use in the program.

Java seems to be the Operating System that satisfies most of the market requirements: portability across a wide range of chip platforms, support of selectable levels of security, and facilitation of partnership developments. Furthermore, Java can operate on a smart card's IC chip and, at the same time, can support the terminal where the card is used and can be used to support the application server.

Let us describe the main features of a set of different Multi-Application Platform Cards:

- *MULTOS Card (see Figure 7):* MULTOS stands for MULTi-application Operating System. The NatWest Development Team as a secure platform

Figure 7: MULTOS OS structure



for the Mondex electronic purse developed it in the mid-90s (DatacardGroup, 2001). MULTOS includes everything needed for a multi-application smart card the operating system, the virtual machine (abstract layer) which sits on top of the operating system, and a card security manager that determines what can and cannot be loaded onto the card.

- *JAVA™ Card (see Figure 8)*: The specifications for Java Card come in three parts:
  1. The Java™ Run-time Environment
  2. The Java™ Card Virtual Machine
  3. The Java™ Card API

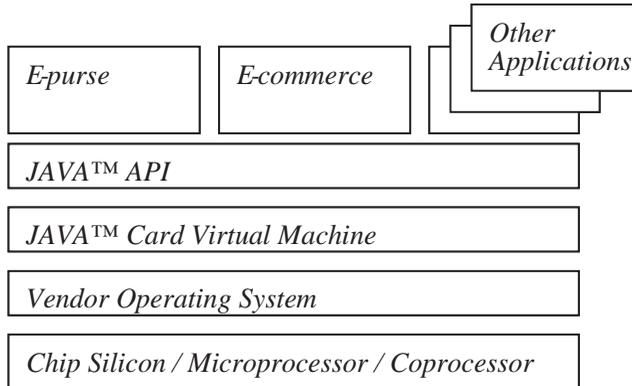
Java™ Card is not an Operating System, but a series of specifications that define how a Java VM can run on any vendors' operating system.

- *Windows for Smartcard (WfSC) (see Figure 9)*: Announced in 1998, and developed and supported by Microsoft™ WfSC includes an operating system, a WfSC Virtual Machine, and an API.
- *“Open Platform” Card (see Figure 10)*: “Open Platform” builds on the card platforms mentioned above, and the major difference is the addition of an Open Platform Application Programming Interface (API) implemented on top of these other card platforms. This API adds a standard application programming interface and a standard security manager to these different cards.

## Piracy and Main Threats

As the smart card market increases, piracy represents more and more a loss of money for the whole industry, which is quite impossible to assess. Besides, a

Figure 8: JAVA™ card structure



multiplicity of new applications are entering the market, such as e-commerce, network access or multi-applications. Smart cards deal with applications with highly valuable content to protect. Hacking a card can be of high interest economically in some fields, particularly in banking or Pay TV segments.

It is usually assumed that fraud cannot be completely eradicated. Indeed, the level of security cannot compromise economic viability of the application, simply because increasing security implies increasing the overall cost of the system. A common occurrence is: “anything can be hacked, it just depends on your time and money.” In other words, piracy exists but it must be weakened as much as possible.

Pirates or hackers know quite well the functioning of standard platforms. Let us take the example of PayTV system. In the case of broadcast via satellite, the content is sent over the air, but it is not possible to control who views it, since no physical return path exists. The mobility of decoders and the difficulty to

Figure 9: WfSC card structure

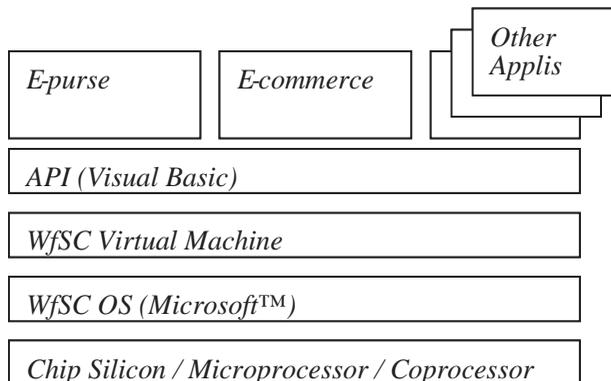
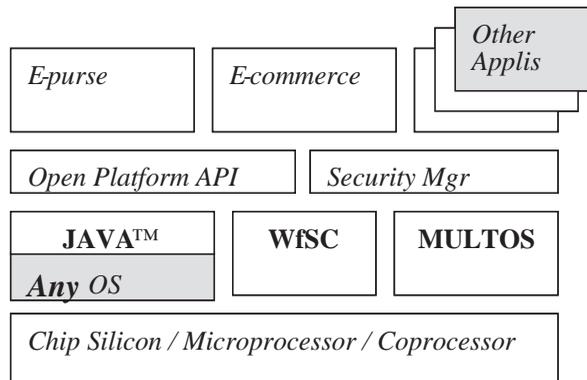


Figure 10: Open platform card structure



detect fraud give more time to the hacker to breach the application, explaining why, since 1994, all types of smart card chips have been hacked and the high level of piracy of decoders (Kômmerling, 1999).

Banking does not have this constraint since authentication generally requires a connection to a server. GSM operators can also cut the line if an abnormal use is detected.

Let's analyze the main threats and techniques used to extract sensitive data and software from the chip. We will try to give a good overview of well-known attacks on silicon chips, but we won't be exhaustive. All information below is public. We distinguish two ranges of attacks:

- The first one deals with invasive attack techniques: they modify physically the chip in order to extract information
- The second one deals with non-invasive attack techniques: they provide sensitive information without any physical modification of the chip.

### *Invasive Attacks*

Reverse engineering is one of the most well known techniques for in-depth decryption of the behaviour of a chip. The method consists of rebuilding the electronics structural circuitry from the layout. While understanding software code requires computer knowledge, reverse engineering needs specific design electronics knowledge and skills, and much more expensive equipment.

In a first step, the card body is heated up in order to remove the chip module. The chip is a very thin silicon structure (a few hundred microns thick) with transistors and wires, made of many layers, built on top of one another. Each one must be dissolved in appropriate chemical "baths" in order to let physically

appear the underneath layer. At each step, we photograph the chip, with dedicated tools. The progressive elimination of layers must be done accurately so not to destroy the lower ones. Indeed, transistor connections are implemented in the lower layers. A complete observation of the connectivity allows rebuilding electronics functions (see *Figure 11*). Custom tools, specifically designed by semiconductor groups and research centres, exist to extract, automatically, the layout from photography.

Reversing the memories is particularly interesting since the code is partly implemented in ROM. ROM code masks are doped, so, under the gates additional implants are located, whose state represents the content of the bit.

Micro-probing and FIB (Focused Ion Beam) workstations are very efficient once some layers have been removed. Micro-probing workstations contain mainly a special optical microscope. Associated with sub-micron probes, it is possible to apply voltage on specific wires (like bus or I/Os) and to monitor parts of the circuit. Full stations can be sold within a budget of around a few thousand dollars.

However, the most useful instrument is certainly the FIB workstation. It has been produced commercially for approximately ten years, primarily for large semiconductor manufacturers. The first usage of FIB has been in the semiconductor industry for defect analysis, circuit modification or mask repair. FIB systems operate in a similar fashion to a scanning electron microscope (SEM). But, rather than a beam of electrons, and as the name implies, FIB stations use a finely focused beam of gallium ions that can be operated at low beam currents for imaging or high beam currents for site specific sputtering or milling. So it is possible, with a sub-micron resolution, to remove material on a chip, modify the inter-connections or deposit metal (generally platinum), creating new connections. Hackers use this powerful tool to reconnect test fuses, create contacts in the chip or perform reverse engineering.

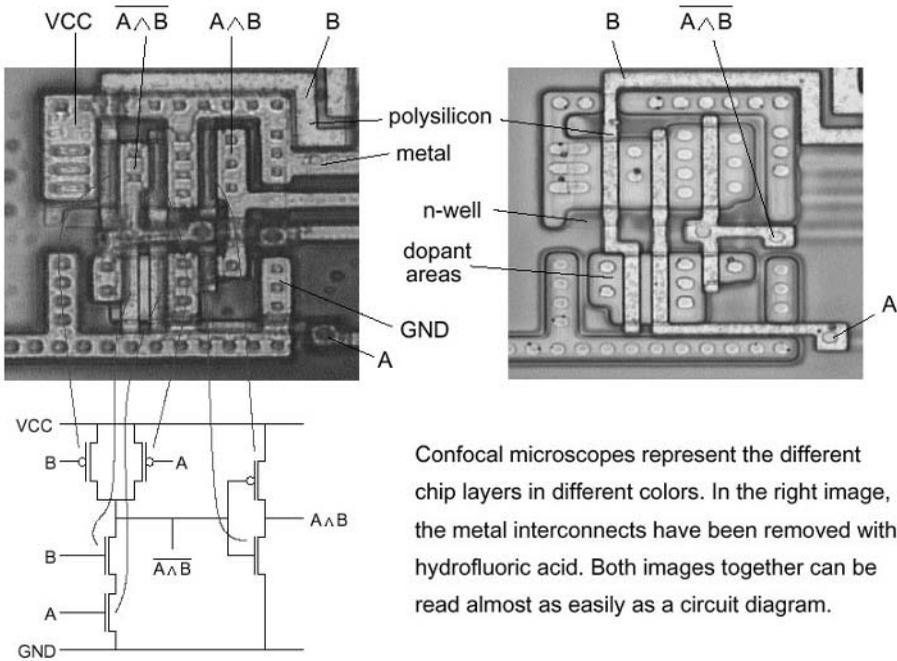
Other techniques and tools exist to monitor ICs. Even if it is long and knotty to understand the functionality of a macro cell of several thousand gates, it is possible to get information about new macro cells, security mechanisms or software code.

### *Non-Invasive Attacks*

Non-invasive attacks are particularly powerful. A system can be abused at a low cost without destroying the chip.

These attacks exploit the inherent architecture of semiconductor chips made of thousands of transistors, the basic element of these systems. The transistor acts as a switch and its state depends on the current passing between its Gate and its Drain. Therefore, the consumption of a transistor indicates the state and a pick of consumption betrays a change of the state. The CPU and the logic macro cells', being a combination of transistors and flip-flops, macro-characteristics are strongly dependent from power consumption and switches.

Figure 11: Reverse engineering of a logic cell



The glitch attack consists of generating a malfunction to mislead the code execution. The fault causes one or more flip-flops (made of transistors) to adopt the wrong state. Consequently, it can trick the smart card circuit into skipping a portion of an algorithm, missing an instruction branch or bypassing cryptographic sophisticated barriers. The goal can also be to dump the memory or to corrupt sensitive data while they are transferred on the bus. Glitch techniques consist in putting a little hiccup into the normally steady clock input, on the power supply pin or on the frequency input. Glitch attacks exploit a fault generation technique.

Fault generation has been efficiently exploited since September 1996. Boneh, Demillo & Lipton, from Bellcore, announced a new type of cryptanalytic attack, which received widespread attention (Boneh, 1997). They stated that it was possible to find the secret key of public-key cryptosystem both by using differential cryptanalysis theory and generating faults. E. Biham & A. Shamir extend this attack to private-key cryptosystems (Biham, 1997). Thus, one can induce a fault at a random bit location in one of the registers at an intermediate stage in a cryptographic computation and repeat the experience with the same clear text and key, injecting the fault in the same temporal window. Applying several times the method, it is feasible to deduce intermediate sub-keys leading to the secret key. Several practical methods were presented, making this attack very efficient against all cryptosystems.

Other non-invasive attacks exploit the electrical consumption of the chip. We provide in the following two examples what is sometimes called in the literature side-channel attacks (Kocher, 1998; Lash, 2002).

The first is the SPA (Simple Power Analysis). In this case, only monitoring the consumption of the chip using an oscilloscope reveals interesting information. Indeed, the amount of power consumed depends on the CPU instructions performed. Large features such as a DES, AES or RSA computation can be detected. In *Figure 12*, an analysis of the consumption shows the 16 rounds of the DES. If no countermeasure is implemented, it is possible to estimate when key bits or sub-keys bits are transferred on the bus.

The second example is correlated to SPA (Single Power Analysis). P. Kocher, J. Jaffe & B. Jun published, in 1998, a method to perform a more dangerous attack, called DPA (Differential Power Analysis) (Kocher, 1998). It is based on a statistical analysis of the power consumption issued from measures a large number of computations with the same key. The DPA for DES proceeds as follow (Lash, 2002):

- The hacker runs the encryption algorithm for several random values of plaintexts
- He chooses a target bit at the output of a S-box
- He makes an hypothesis on the involved key bits
- The attacker performs a statistical analysis of the results for all inputs. The form of the final trace is characteristic if the hypothesis is correct
- He repeats the same techniques to retrieve all key bits.

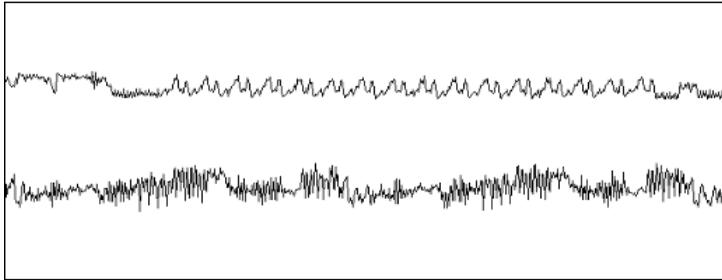
Rather than measuring consumption using an oscilloscope and a computer, it is also possible to simulate VHDL hardware model and exploit the power analysis consumption simulation results.

We present all basics techniques of attacks. Some are very efficient. We point out that techniques discovered to breach smart card chip can be reproduced in other systems. Specific counter-measures have been developed to resist these attacks.

## **Tamper-Resistance Techniques and Impacts on Platform**

Secure micro-controller architecture evolution is tightly dependent on new hacking techniques. An analysis of the architecture presented shows the basic elements to ensure protection, like sensors, memory firewalls, and cryptographic macro cells. (We presented also evaluation schemes). However, to fight against “hacking,” the semiconductor industry has to adapt and propose other solutions. We propose to go further in describing other tamper-resistance mechanisms such as:

*Figure 12: Current analysis of a DES operation of a smart card chip*



- Side-channel attack protection
- Software configuration integrity
- Hardware design protection
- Memory and bus encryption
- Custom macro cells approach

When side-channel attacks appeared, cryptography was supported by crypto-processors or hardware DES yet, driven by dedicated software libraries. Hardware firewall protections ensure integrity of these libraries, the only ones able to control these macro cells. Operations can be eavesdropped. Software libraries were redesigned taking into account consumption weaknesses, mainly using randomization of data. All sensitive data, transferred on the bus, are mixed using random techniques reducing consumption correlations. Besides, some macro cells were re-designed following the same techniques. Sensitive data transiting internally are mixed with random data and non-linear pieces of algorithms were recalculated and redesigned in order to allow random data injection without modifying the overall algorithm. Coupled with other techniques, correlations capabilities of side-channel attacks have been reduced considerably.

Device software usually has access to the data that needs to be protected. In order to avoid malicious code to be downloaded and executed, a secure boot performs several verifications to guarantee the integrity of the chip. A common occurrence in OS development is to never trust the hardware. Checks are performed regularly in order to be sure that code and hardware are not corrupted. Techniques exist to destroy sensitive information in case of possible attempts.

Reverse engineering of the chip has always been a problem. Most smart card silicon providers used primarily shield protection. A metal, covered with integrity sensors, is dedicated to protect the upper layer of the chip. In case of physical intrusion, the card destroys the secrets. However, with the evolution of

design techniques and thinner technology, the number of layers has increased both with the number of transistors available in the chip. When performing a reverse engineering, it is often uncertain to associate a high level metal wire with the good transistor. Beside, the progressive elimination of layers damages the lower ones. Decrypting a complete design becomes a brainteaser. Moreover, the most advanced designs mix up all the digital parts in glue logic. All transistors are randomly mixed physically in the layout. Final handed layout modifications complete the protection, rendering it almost impossible to reverse a complete block. Nevertheless, memories are not mixed with the digital parts. Encryptions and scrambling are performed. Memories are not regular matrixes and data stored are randomized hiding the true values. Following the same scheme, the bus of the CPU is not only part of the glue logic of the chip, but also encrypted randomly at each cycle clock.

Another approach increases security. Since most smart card chips are standard platforms, some “OS makers” have decided to get a differentiated product to avoid cloning of the cards. Indeed, once a code is breached, it is possible to use standard smart cards and download a code, reproducing the behavior of the hacked card. Developments of hardware macro cells or full custom chips appear in the marketplace in order to strengthen security.

Security is an issue in many industries. Initiatives are generalizing, particularly in the consumer world. The main industry hit by hacking deals with Pay TV systems. It is, however, considered as the strongest security system existing today. Nevertheless, in Italy in 2002, 3 million hackers were estimated to watch digital TV using a hacked card, representing \$2.5 billion losses for Pay TV operators. Smart card chip increases its security, but hackers exploit also security breaches in the Set Top Box itself. For example, the Operating System of the Set Top Box, contained in a flash memory, is dumped and re-programmed in order to bypass smart card authentication. In this case, there is no need to insert a card in the decoder.

The JTAG (Joint Test Action Group) port is also used to analyze the code and re-programmed boot programs in the Set Top Box. Furthermore, the emergence of PVR (Personal Video Recorder) technology has created a problem. Since the movie can be recorded in a digital form in the disk, it becomes a high valuable asset that needs to be protected.

Mobile industry now also suffers from the same hacking techniques. Hacking on terminal mobile phones that attack the operating system used in the handset chip are already commonplace. The identification included in the handset can be re-programmed in order to bypass the SIMLOCK (this is a protective mechanism in the Operating System handset that restricts a certified SIM card with a given mobile). Indeed, some operators used to subsidy handset manufacturers in order to propose competitive packages (subscription plus handset) to users. Once the SIMLOCK is bypassed, it is possible to get a handset at a competitive price and then to choose the subscription operator.

Both GSM handset chips and Pay TV decoders hosted mainly two chips: the first one is a SOC (system on chip), a chip embedding several processors and functionalities; the second one is a flash memory, storing the Operating System. System integration and data processing. These were privileged in order to reduce the price of the system. However, these chips were deprived of high security features.

The trend is changing. New security features are now included within these systems:

- Flash memories are secured with firewall techniques
- Hardware DES and AES inside the system on chip allows data encryption
- Secure Access Channel secures the link between the smart card chip and the system on chip
- Encryption techniques protect the link between the flash and the processor
- Unique identification numbers are inserted in the system
- Public key cryptography algorithm is being introduced.

The description is not exhaustive. Other obfuscation techniques and software counter-measures exist or are being developed.

## FUTURE TRENDS

In this last part, we discuss the new security requirements imposed by both the digital convergence of the consumer market and the advent of open networks. We will try to provide ideas about what roles the smart card industry can play in future integrated systems within consumer platform and e-commerce, providing a case study of a solution, called SmartRight, for content protection. To conclude, we will endeavor to introduce biometrics and bring out long-term trends of the smart card industry.

### Consumer Market and Security

Content (movie, song, video game, software...) is delivered to the home from many sources (see *Figure 13*). Outside the boundaries of the home, we distinguish two main domains of protection. The first one deals with contents transferred through satellite, terrestrial, cable or ADSL for digital TV. It is traditionally protected by Conditional Access Systems. The second one deals with the Internet, which opened a second channel of distribution of content, delivered to computers. This channel is protected by DRM (Digital Rights Management) systems.

Once it arrived in the home, content is decrypted and the consumer can use it without any restriction. The domain inside the home is not protected. However,

the digital form of content and interconnections through devices represent a new threat. Today, most of devices in the house are still interconnected via analog links, but the fast development of digital interfaces in the house will lead to a generalization of Digital Home Networks (DHN): all devices will be interconnected and the content, like movies, will transfer from one device to the others. The emergence of both Digital Video Recorder (DVR) and the Personal Video Recorders (PVR) in the Set Top Boxes will allow people to record movies, coming from broadcasters, in a digital form that could be re-distributed over the Internet. Consequently, a third domain, inside the boundaries of the home, needs to be protected.

On top of the current industry-scale piracy, the fast download of digital content through the Internet had dramatic consequence in the music industry. The IFPI (International Federation of Phonographic Industry) recently reported that global sales is losing 5% each year since 2001 with a loss of about \$5 billion (compared to \$37 billion in sales) due to hacking (Berman, 2003). The movie industry is following the same trend. The MPAA (Motion Picture of America Association) reported more than one million illegal downloads performed in 2002. With the advent of DHN and new technologies, high quality movies could suffer a massive piracy. Content protection is a main concern for content providers. They are pushing all technology providers to find secure solutions. The consumer industry, in accordance with silicon providers, multiplies the initiatives in order to propose efficient solutions.

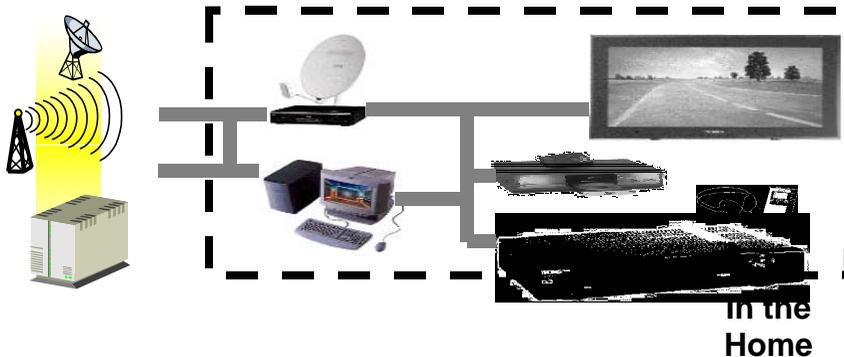
A second trend is happening in the consumer market. The evolution toward the DHN is on the way. But, what will be the home gateway of the DHN? This hub is likely to be a device being able to receive and re-distribute all types of contents inside the home. Three devices could play this role: the set top box, the PC and the game console.

Security is a key element of the main stake of tomorrow's consumer market evolution.

Computer designs have neglected security on their chips, relying on DRM solutions to protect the Internet channel (DRM intends to protect content by associating an access right to the content). Nevertheless, these solutions are widely attacked because their reliance on cryptographic software modules is unable to make up for hardware weaknesses (Allan, 2002).

Intel and Microsoft are determined that the PC will be the hub of the future home network. If entertainment is a key application, DRM is going to be the critical enabling technology. The PC has then to do DRM or risk being displaced in the home market. They are trying to propose solutions. The TCPA, which stands for the Trusted Computing Platform Alliance, is an initiative led by Intel. Their target is to build "a new computing platform for the next century that will provide for improved trust in the PC platform." Palladium is software done. It provides a computing platform on which you can't tamper with the applications,

Figure 13: Consumer environment



and where these applications can communicate securely with the vendor. The targeted application is DRM. Content providers should be able to sell DVDs, decrypted by the platform. Obviously, an anti-copying system should be included. They also will be able to sell you CDs that you'll only be able to play three times. Other marketing possibilities will open up. TCPA/Palladium will also make it much harder for you to run unlicensed software (TCG, 2003).

The Microsoft's Xbox game console is another example of a platform that tries to implement security. Nevertheless, the first version suffers weaknesses. A MIT graduate student took tools left over from his just-completed Ph.D. thesis on symmetric multiprocessing (SMP) systems and applied them in breaking some security features of Microsoft's Xbox (Huang, 2002). His efforts, which began just few days after Xbox went on sale, could potentially open the Xbox platform to other operating systems and for support of nonstandard peripherals.

The platform didn't include the traditional traps used in the smart card industry, but rather tried to re-think security techniques. Thus, the platform included built-in booby traps intended to prevent software not authorized from running on the machine. They are also used to implement DRM (digital rights management) and to provide each Xbox with a unique, traceable serial number.

Techniques includes RC-4 encryption (RC-4 is a private-key cryptography algorithm), non-working decoy code in the system's Flash ROM, and a "secret boot block" hidden within one of the system's application-specific integrated circuits (ASICs). The use of a symmetric algorithm is a problem when one party is not trusted. Besides, one of the security features was based on the high throughput of the high-speed bus. But it was possible to intercept communications between the device's north-bridge and south-bridge chips, exposing the decryption key and the contents of the boot. Sensitive operations were done in hardware, but it was possible to perform reverse engineering. No protection technique used by secure microprocessors was used. Nevertheless, security is now a big concern in consumer industry.

There's no doubt that security is on its way to being implemented in all future consumer platforms.

## **Case Study: E-Content and Smartright Proposal**

The efforts of industry are also concerned with e-content protection inside the home. Standard bodies and the MPAA are leading the battle.

Thus, the DVB (Digital Video Broadcasting), the European standard body of digital TV, has issued toward the community a call for proposals for Content Protection & Copy Management Technologies (dvb.org, 2001). The DVB request aims at defining a common technical framework that will be able to protect all types of content within the home network. One of the requirements of the DVB Content Protection and Copy Management (DVB-CPCM) is to provide an end-to-end protection, i.e., from the point of content origination to the point of content consumption. Several companies answered the DVB. In the following, we propose to analyze the initiative created by Thomson to deal with content protection, called SmartRight (smartright.org, 2001) and submitted by a consortium of companies to the DVB. We propose, in the following, to develop SmartRight basics as a case study representative of consumer security new market demand and using a smart card.

### *Hardware Architecture*

The architecture uses the conditional access principles. SmartRight architecture is based on smart card technology. At the entry point of the home network, the device scrambled or kept scrambled the content. The content moves scrambled over the network until it is descrambled and rendered by another smart card to the consumer device.

The management of content protection uses data packets called Local Entitlement Control Message (LECM) inside the flow of content. They encapsulate the content scrambling keys and the access rights following the traditional scheme of MPEG-2 management in main pay TV systems. Cryptography mechanisms secure the LECM over the network. The DVB-CPCM defines several protection levels. They are dependent on the usage state of the content: copy control not asserted, copy once, copy never or copy no more. The behavior of the system and its cryptographic mechanisms depends on this state. The concept of a Personal Private Network (PPN), composed of a set of devices is introduced. A set of security rules governs the life of smart cards and checks the coherence of the network activity. Once entered, the content is handled and protected. Inside this protected home network, the referenced architecture defines three types of consumer devices (see *Figure 14*):

- The access devices
- The presentation devices
- The storage units

And two types of cards are considered:

- The converter cards
- The terminal cards.

The access devices receive the source of content from the external domain and deliver it to the PPN. It can be any distribution means inside the home network. The presentation devices are any devices rendering the protected content. A typical example is a Digital TV. It can also export the content to proprietary copy protection systems. The storage units are anything that records the content. No additional card is required. The content remains scrambled while stored.

Some devices may play several roles: a typical example is a Set Top Box with a PVR (access device and storage unit). A converter card is associated with an access device. It includes the SmartRight module that creates the LECM. If the access device is a Set Top Box, the conditional access card can embed also this module. The terminal card is associated with the presentation device. This card decrypts the LECM and, according to the usage state, sends back to the presentation device a control word to authorize or not the descrambling of the content.

### *Content Protection Management*

The DVB-CPCM defines several levels of content protection, chosen by the content provider:

- “Copy control not asserted:” the user is authorized to view the content and to make as many copies as he wants. This is the “copy-free” state.
- “Copy once:” the content can be copied and used only inside its associated PPN.
- “Copy never” and “Copy no more:” the content cannot be recorded but only viewed once rendered by a presentation device.

Two different PPN can share only “copy-free” contents. Besides, as a PPN is associated to the smart cards, it can encompass several physical homes. In order to provide a consistent end-to-end protection, the content is managed by the LECM. A specific network key protects this data structure. This key is unique and created randomly at the PPN creation. The content to be protected can be video or audio, but also images, web pages or software. In this system, no assumption is made on the content type.

Let us describe now the flow of the content in the Pay TV system in Europe. The MPEG-2 transport stream arrives scrambled in the set top box. It is scrambled using the DVB Common Scrambling Algorithm (DVB-CSA). Then, Entitlement Control Messages (ECMs), containing the Control Words (CW) and a brief description of the content, are extracted and sent to the conditional access smart card. Once decrypted, the descrambling key (or Control Word) is sent to the set top boxes in a secure way in order to render the content. In the SmartRight system, the LECM management depends on the content received by the PPN. If it is a MPEG-2 content, then the conditional access module of the converter card will securely decrypt the ECM. To ensure the content protection, the SmartRight module will then re-encrypt it into a Local ECM (LECM). The LECM replaces the ECM at the entry point of the PPN. The content received by the PPN can also be yet protected by a DRM and another content protection if it is in a pre-recorded format like a DVD or simply free-to-air. In these cases, the rights of the content are extracted and the content will be converted in the SmartRight format. The LECM replaces the protection data structure at the entry point of the PPN.

The SmartRight algorithm used to scramble the content is still under discussion. The DVB-CSA could be one of the candidates, since it proved its robustness in the past.

Consequently, the converter card creates the LECM and the terminal card extracts the control word enabling the rendering of the content.

All the relevant information for copy protection broadcasting inside the home is carried by the LECM. Its format follows the MPEG-2 transport stream format. The data structure is encapsulated in the `private_data_byte` structure of the Private section defined in the ISO 13818-1 standard (ISO13818-1, 1996).

The LECM is divided into two sections. The first one is the protected section. It contains the descrambling (control words) and view only information that will be described deeper in the following. It is encrypted for “private copy” and “view only” modes.

The second part is the plain section, never encrypted. It contains mainly the following information:

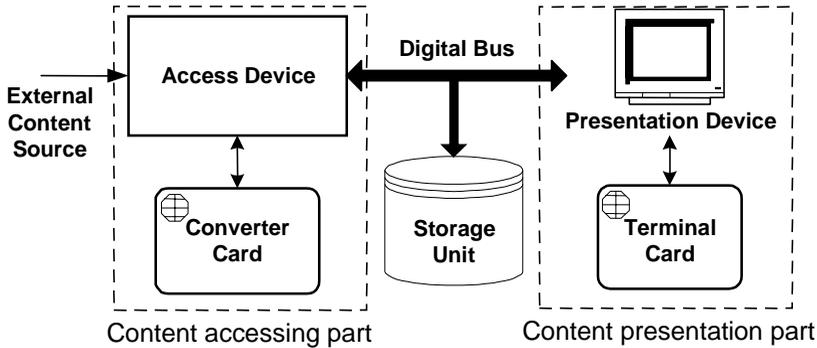
- The content type
- The usage state
- The 128-bit encrypted LECM key.

A 160-bit integrity check is associated with the LECM.

The content protection management over the network relies on cryptographic schemes.

A unique network key, present only in terminal cards, is created randomly at the PPN creation. The converter card manages the LECM. It picks at random a LECM key when a new device is connected or upon request of the DRM/CA

Figure 14: Consumer devices and cards in SmartRight system



module. The converter card then transfers it in a secure way (using public key cryptography) to one of the terminal cards, which encrypts it with the network key. It returns to the converter card the resulting encrypted LECM key. When creating a LECM, the converter card uses the LECM key to encrypt the protected section of the LECM and puts the LECM key encrypted with the network key in the plain text section of the LECM.

These encryptions use symmetric cryptography. The integrity check of the LECM is done before the encryption of the LECM. This is a 160-bit hash value computed using the Secure Hash Algorithm (SHA-1) (Krawczyk, 1997). The management of LECM and its content depends on the usage state of the protected content.

When the content received is in “copy free” state or in “private copy” state, the content descrambling information of the LECM is the Control Words (CW). The view only section contains random values.

In “copy free” mode, the LECM remains fully in clear. So, once received by the terminal card, the CW is extracted and the content is descrambled in the presentation device.

When the content received is in “private copy” mode, the protected section is encrypted with the symmetric algorithm using the LECM key. Once in the terminal card, the encrypted LECM key is decrypted using the secret network key. The protected section of the LECM can then be decrypted, giving the access to the CW. A presentation device of another PPN cannot decrypt the LECM.

In “view only” mode, the Converter card and the Terminal card uses the HMAC-SHA-1 algorithm (RSA, 2001). This algorithm specifies the use of the Hash function combined with HMAC as a keyed authentication mechanism. The goal is to provide both data origin authentication and data integrity for packets sent between the two parties. This is an 80-bit secret key authentication algorithm.

When “view only” content is received, the Converter card picks at random two values: X and Y. The content descrambling information is a combination of X and the CW. The view only information is Y. The encrypted LECM is sent to the Terminal Card. Once received, the LECM is decrypted. The Terminal card cannot retrieve the CW. It issues a random W and sends it back to the Converter Card. This one computes  $f(W)$  using Y as the secret key,  $f$  being the message authentication process using the HMAC-SHA1 algorithm. It combines  $f(W)$  and X and sends back the data to the Terminal Card. The Terminal Card verifies W using the secret key X sent previously and retrieves Y. It can then extract the CW to descramble the content. The protocol is shortly described in *Figure 15*. Another PPN cannot decrypt the LECM. A recorder that wants to replay the content is unable to respond to the correct challenge since it doesn’t know the view only information.

### *PPN (Private Personal Network) Infrastructure*

The network key is the secret data shared by a PPN. It is held by the Terminal cards. The Terminal cards have three states (see *Figure 16*):

- Virgin: it has no network key.
- Progenitor: it has a network key and can send it to a Virgin card.
- Sterile: it has a network key but cannot send it to a terminal module.

When a new device is connected to the PPN, its Terminal card is Virgin. If it is the first Terminal card in the PPN, it picks at random a network key and becomes Progenitor. Else, one Terminal card in the PPN detects the new card and is able to send it the network key in a secure way. When the key is transmitted, this card becomes Sterile.

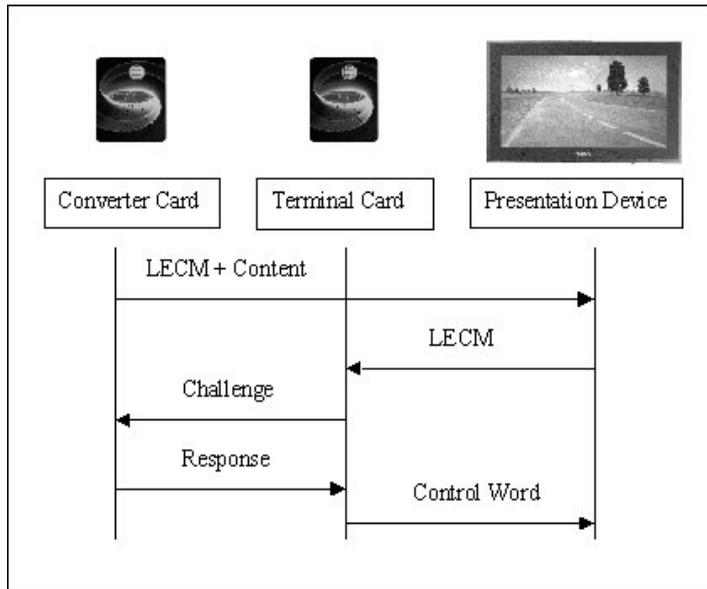
A card can be initialised to Virgin state upon request of the user. *Figure 9* shows the different states of the Terminal card.

Each Terminal card has one asymmetric key pair, contained in a Terminal certificate. Besides, the Terminal and the Converter cards hold a Certification Authority public key, used to check the validity of other certificates.

Before transferring the network key, the Progenitor and the Virgin modules exchange and check their certificates. Then, the network key is sent, encrypted with the Virgin card’s public key. The chosen asymmetric cryptographic algorithm is the RSA with 1024-bit key, described in PKCS#1 v2.1 (RSA, 2001). The same algorithm is used when the encrypted LECM key is transferred to a Converter card. Besides, the RSA with 1024-bit is used to sign certificates.

Progenitor and Sterile cards contain a network identifier. This one is the hash value of the network key, computed using the SHA-1. It allows checking the coherence of the PPN when a change or user request occurs. The number of devices inside the PPN is limited by a network size data, stored permanently

Figure 15: Protocol view only



in the Progenitor or Sterile cards. This number is decremented each time a new device is connected. When the counter is null, the PPN has reached its maximum size.

SmartRight technology was one of the numerous proposals submitted in response to the DVB Call for Proposal for Content Protection and Copy Management Technologies. It is an example of solution for e-content protection. It's likely that the future standard of content protection will come from a general consensus of the whole industry and will result from a technical compromise between various solutions.

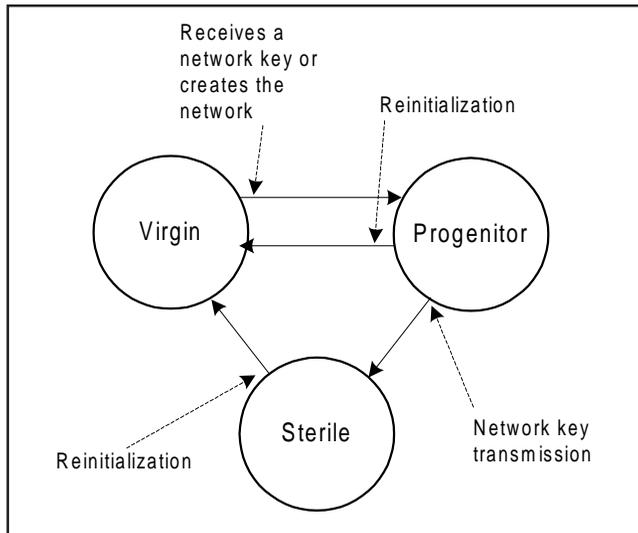
Secure solutions are developing outside the traditional scope of the smart card industry. Constraints are different so innovative solutions will emerge. These solutions will lead, in the coming years, to the protection of the full digital chain.

## E-Commerce Trends

E-commerce evolution is directly linked to Internet expansion through PC and Wireless's tremendous accessibility. This new mass market is an opportunity to stretch the application field, with secure real-time processing of high volumes of data.

This expansion offers exiting opportunities to smart card business through these innovative applications, which loom up on the field. These new kind of applications require:

Figure 16: Terminal card states



- Data storage capacity (video, music, maps, graphics, database, applets, phonebook, personalization)
- Mono-directional flow for application with high volume of data requesting real time playback, (e.g., music, video)
- Bi-directional isochronous flow for streaming applications with high volume of data exchange in both directions (e.g., data encryption for connecting to banking application, contents right control { watermark check }, encryption/decryption of voice or IP packets)
- More and more security and cryptographic capabilities for covering confidentiality, integrity, and authentication needs.

In this context, the smart card has turned out to be the essential vector for generating digital signatures, performing mutual authentications, and also ciphering, essential operations for online transactions.

However, introducing smart cards within an Open Network market is a real stake. Market acceptance is a key factor that can be reached through the following statement: Smart cards must achieve a high performance secure access to the Open Network, without extra cost.

The smart card success factors are summarized as follows:

- Smart card security brought to PC or Mobile world
- Low cost implementation: no more smart card readers -

- Increase of the communication speed
- Dual (or more) mode smart cards (for ISO compatibility)

A pending crucial success factor of fruitful commercial applications stretch is consumer confidence towards security and confidentiality of the traditional payment models. Therein smart cards overcome weaknesses of traditional specific applications, and enter the tremendous Internet applications mass market. Here the smart card can strengthen the global application through robust security features and appropriated PKI algorithms.

The first step to link a smart card to the market of Open Networks has been to adapt the ISO communication protocol to the PC one: ISO (International Standardisation Organisation) is the traditional communication protocol used for smart card, when USB protocol is dedicated to computer world.

The first solution has been to develop an ISO/USB reader link between smart card specific protocol and Open Network business. The main drawback of such a product is the electronic complexity of the device, and therefore the inherent cost. The global solution remains very expensive regarding the expected market acceptance: acceptance from consumer, acceptance from PC manufacturer (if integrated within PC), or acceptance from application provider (if solution pushed by financial groups).

To thoroughly fit this new business model, the idea is to integrate within smart cards new communication protocols, and not in the reader any more. This solution allows a direct connection to the PC, just through galvanic contacts. Smart cards become bilingual, and even more! As a consequence, the cost has been dramatically reduced (by a factor of 30!).

To underpin smart card acceptance within innovative applications, which for Internet will be mainly based on securing audio and video, speed enhancement boundary cannot be a limitation factor. Choice of new smart card protocols for Open Network must be dictated by the ciphering/deciphering capabilities for streaming services: speed enhancements of 12 Mbps (Mega bits per second) are sought-after for multimedia (audio, photo, low speed video, games) and telephony, modem; high speed applications of up to hundreds of Mbps are planned in a very near future:

- Multimedia (digital video, video conference, DVD),
- Telecom and datacom applications,
- High-speed transmissions.

Smart card has to evolve and adapt itself in that changing environment: communication channel, with respect to transfer rate, size and protocol, has to move from ISO 7816 T=0.

The current studies led by the 3<sup>rd</sup> Generation Partnership Project (3GPP) evaluate the following protocols from the smart card side:

- Universal Serial Bus (USB)
- Multi Media Card Protocol (MMC)
- Enhancements of ISO/IEC 7816 T=1

In this context, USB protocol is particularly adapted: USB 2.0 specification defined speed enhancement up to 480 Mbps (USB High Speed).

A product integrating the USB protocol has been designed to fulfill the requirements of the market evolution (STMicroelectronics in collaboration with Schlumberger, followed by other chip manufacturers): able to operate in ISO mode as well as in USB mode, this product can be used both on computer terminals and on traditional smart card terminals (payment, banks, health, identification and so on). Thanks to this solution, the terminal, previously an intelligent device is now reduced to the minimum, i.e., an USB connector, without the need for an external clock the smart card chips were requiring. To achieve this goal, it has been a challenge for chip manufacturers to generate an internal clock for USB devices with an accuracy of a few percent and this, without any external devices (oscillators, capacitors...).

Through this smart card USB device, the Internet world is opening its door to a new field of secure and fast commercial transactions: the user simply slots this card into its dumb connector, and the system recognises it as a USB peripheral. Out of the connector, it fits in the user wallet, ready for use in smart card terminal (access, banking, loyalty, and transport applications), giving users all the advantages of a traditional ISO card. This highly versatile digital ID is ideal for e-commerce and secure Internet access. The USB smart card will become your ID for all computer transactions and will be mandatory in terms of IT security. All these applications at the lowest cost for the PC manufacturers... and for you, because a PC with USB terminals will be at no additional cost!

## **Biometry Introduction**

For all secure applications, a problematic is the same: how to authenticate users with security and convenience? Current security solutions are based on something you know (PIN codes, Passwords) or something you have (keys, smart cards). But both of them could be stolen, borrowed, lost or even rented. High security solutions must be a combination of both something you know and also something you have, like self-fingerprinting. It is the best suitable approach for security and convenience. Biometrics is based on something you have.

### *Biometrics Market*

The authentication product market is rapidly blooming and the main technical trend for biometrics is the use of fingerprints. First, applications are access control for computers, network security point of sales and government national ID cards. A second phase will certainly be supported by e-commerce roll out. E-commerce and Internet usage are growing markets for security but they are not alone. Telecommuting communication and data flow for customers or subcontractors are also pushing security requirements for confidentiality and true identifications for “online” networks.

Indeed, tomorrow, once every appliance will be connected on a network, those appliances will be use to conduct business and financial transactions. They will represent an opportunity for new business models in order to enhance transaction security and ensure non-repudiation. Security and non-repudiation will only fully be achieved when biometrics will be used to authenticate users. In addition to the high psychological impact from September 11, 2001 events, biometrics applications are promising a great future. Up to today, the market was mainly for identification systems used by government agencies such as the FBI (Federal Bureau of Investigation).

With price points for biometrics systems going down and better means to authenticate people, it is expected that the market in 2005 will exceed one billion US\$.

Biometrics technologies are based on two things: firstly on the behavior and secondly on physiological:

- Behavioral systems are based on voice written signature or keystrokes that are unique but timely variable. This is why behavior systems are not widely used due to poor probability results.
- Physiological systems are better due to unique and permanent definition (finger, hand, eye, face or vein). The best compromise used today in term of quality and cost is, with no doubt, fingerprint with silicon sensors usage.

It is more secure than voice; for instance, a voice can be faked with a simple recording. It is easier to use than iris recognition (Gifford, 1999). It consumes much less power than fingerprint optical technology. Most of the fingerprint systems sold in 2000 were optical sensor-based. In 2004, the market share of silicon based-system should be 80%. *Table 1* compares different biometrics technologies. ++, + and - symbolize the degree of importance of the function, according to the technology.

The main leaders in this silicon fingerprint sensors are STMicroelectronics and Infineon.

### *Fingerprint Recognition*

Fingerprint (Prabhakar, 2003) technology is based on the concept of minutia. Minutia are characteristic points of the ridge and valley: ending points and bifurcations are minutia points that are used to create a fingerprint template. In the U.S., 8 minutiae points are sufficient to legally match two fingerprints.

The benefits of using fingerprint templates are:

- Its size. It is much smaller (around 250 bytes) than a full fingerprint image and can be stored in small memories or smart cards.
- Its security. It protects privacy since it is not possible to reconstruct a fingerprint image out of a fingerprint template.

The fingerprint identification process is divided in two phases: enrollment and verification. Every biometrics system contains an enrollment process:

1. The sensor captures the fingerprint image
2. The biometrics algorithm extracts the template from the fingerprint image
3. The fingerprint template is stored in a memory

To authenticate a user:

1. The sensor captures the fingerprint from the live finger
2. The biometric algorithm extracts the template
3. The biometric algorithm matches the stored template with the template from the live finger
4. Access is granted only if matching is positive

### *Biometrics Challenge*

Biometrics systems are facing different challenges, which are:

- Negative public perception: some people, for psychological and/or cultural aspects, are reluctant to use sensors such as fingerprint for cleanness, or iris recognition for eye “burning” fear. It is mainly a question of education and communication.
- Lack of understanding of the technology, which is quite new and implies some guideline rules to follow before a mass system launch. Again, education is very important before going to biometrics systems.
- Lack of standards: like other high technologies, there are no real worldwide standards. Some software layers are standardized as BIOAPI. Nevertheless, a few companies have starting to discuss this, and it seems that European countries and the U.S. are on the way to find compromise in order to improve the international security level in the fight against terrorism.

Table 1: Biometrics technology comparison

Technology	Security	Ease of use	Compactness	Low Power	Low cost
Silicon Fingerprint	++	++	+	++	+
Optical Fingerprint	+	++	+	-	+
Voice	-	+	++	++	++
Face	-	+	+	-	++
Eye	+	+	+	+	+
Dynamic Signature	+	-	-	+	+

- Product performance-reliability: one of the biggest question marks, which is too often forgotten, is the industry capability to deliver a reliable system, especially if you have a system with thousand of uses a day on the same sensor! Nowadays, silicon sensors could guarantee more than one million uses for a single sensor.
- An appropriate balance between FAR (False Acceptance Rate) and FRR (False Rejection Rate) is fundamental both for security and convenience of the application:
  - i. FAR = someone gets in but is not supposed to get in = security risk
  - ii. FRR = someone is supposed to get in but the system does not let him to get in = inconvenience
- System considerations:
  - No biometrics system will work for all users all the time. Biometrics is based on physiological aspects of human body.
  - Any biometrics system typically includes a backup.
- Prices of biometrics applications is always linked to the level of security you want to reach. However, with more and more deployments of biometrics systems, especially based on silicon, the price decrease should be fast enough to enhance fingerprint technology diffusion

### *Biometrics and Smart Cards*

Biometrics complements smart cards. It is more secure and more convenient than a PIN. And smart cards complement biometrics: storing template on smart cards eliminates privacy concern. If we go further on the duplicity new technology such as Matching On Card (MOC), it seems to be interesting. So, you have the verification process embedded within the card itself. This is why smart card, associated with biometrics, offers a good solution for e-commerce security. It is the best financial level security with consumer level convenience.

The biometrics market will probably go through an impressive growth for the coming years. Up to today, the market was mainly used for police identification systems.

Biometrics is based on something we have, like physiological or behavioral aspects.

Physiological fingerprint recognition with silicon sensor technology is the best candidate with a good balance between quality and solutions price.

The main benefits we could see now are:

- Prevention of unauthorized use of:
  - Cellular phones, computers, cars or Point of Sales (POS) terminal
- Authentication for:
  - Electronic transaction, email or building access
- Replacement of:
  - PIN code, password keys — biometrics cannot be lost, left at home or forgotten!

It is a new and exciting technology, which could rapidly change our everyday life for our security and convenience. Moreover, there is a lot of synergy between smart card and biometrics use, as we could see in applications like national ID and access control systems.

## Future Smart Card Features

Smart card form factor (e.g., smart card physical aspect) has been evolving regarding evolution of market trends, in line with the evolution of society. For example, change from magnetic stripes to embossed chips within the plastic card is the response to security demands and guarantees the protection of private life: emphasis has been put on security. Therefore, to pretend knowing in which direction will evolve smart cards form factor, one has to know market requirements evolution. As shown in the different sections of this work, smart card has and will always follow society's evolution.

New smart card form factors have emerged in the last decade: USB smart card token, which seems to be the essential vector for e-commerce, and SIM plug-in for wireless handsets. Anyway, smart cards need to expand its application field by being the answer to innovative applications. That philosophy implies endowing smart cards with battery, fingerprint sensor, flexible memory, dynamic display, wireless antenna, a keyboard, and — why not — a microphone (see *Figure 17*)?

### *Battery on Smart Card*

The main innovation for smart cards would be their autonomy: embedding an on board source of energy, smart cards would mutate from “slave” status to

“master” status. By the way, it would pry open the door of numerous innovative applications requiring traditional terminal independency.

### *Temporal Counter Embedding on Smart Card*

Society’s evolution led industrials to work on new application items; DVD downloads, VoD (Video on Demand) and content protection are some examples. In the consumer market environment, content (movies, music or software) is downloaded and consumed at the same time. But it can be stored and replayed. Hardware time stamping prevents from replaying the content.

Smart cards turned out to be the essential vector for generating secure cryptographic operations, meanwhile restricting applications duration exploitation by controlling time factors.

Therefore, a register containing an appropriated time variable would be implemented within a smart card (see *Figure 18*) (Patent, 2002). This variable would evolve regularly and continuously in the course of time, thanks to an embedded source of energy (accumulator, capacitor, micro-battery, micro-electronic process...).

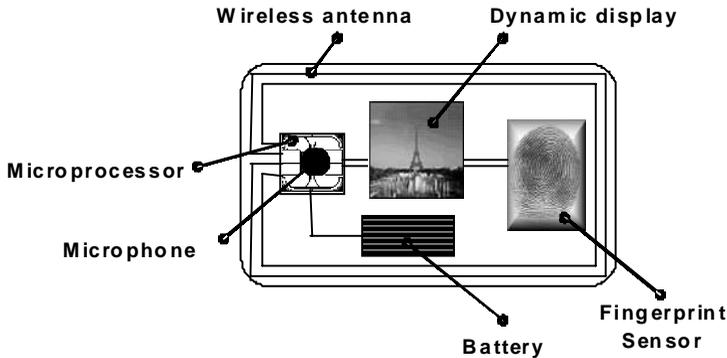
### *Biometric Sensor on Smart Card*

A growing list of vendors believe there is a market for a product that combines a fingerprint sensor with a standard smart card. Such a card could help in identifying cardholders within most of the terminals already deployed on the field.

Why a fingerprint sensor in smart card? With the sensor incorporated in the smart card, emphasis can be put on security of the whole application. This step forward can conduct innovative electronic commerce applications, access control, etc, without having to invest in fingerprint readers. Moreover, security of the global application lays on the fact that any identification data and any process of the information remain within the smart card: no piracy could intervene at any step of the identification operation.

The main concerns on which industry are working are the thickness of the biometric sensor, with a view to meet ISO standards, and the robustness of the sensor embedded in a plastic card. And of course the price of such a device, increasing the global price of the smart card. But even if before the cost was important, due to the Sept. 11 attacks price and security have now changed places in priority.

These functional modifications offer to the user the best control as possible of his card, of the embedded services, and of the embedded private data. By becoming a small autonomous system, pro-active and secure, the smart card is the answer to increasing demand in trust, guarantee, comfort, and ergonomic use for real-world transactions.

*Figure 17: Smart card embedded technical features*

But modifying the card with a view to develop a small autonomous terminal implies deep modifications of the emplacement of the modules. Such a revolution in smart card evolution has to be driven by an evolution of the standards. This will blaze a trail for new coming industrial killing applications.

## CONCLUDING REMARKS

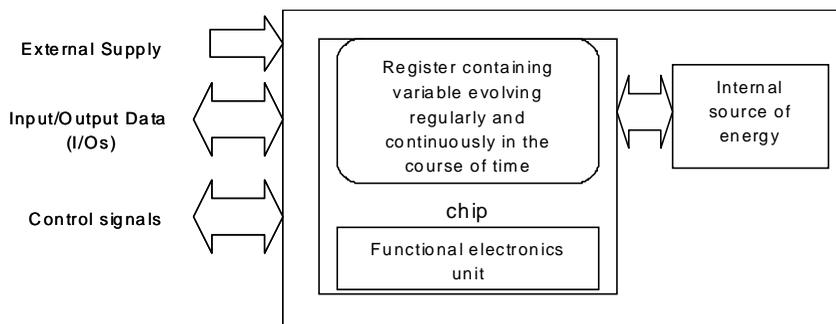
Performance has been the driving factor of the last decades in the semiconductor industry. Integrated circuits have both increased their processing capabilities and integrated even more functionalities on chips. Smart card chips were islands focused on security rather than computing power. Consequently, other architectures suffered from security weaknesses that need to be corrected.

Market demand for security is increasing in many sectors and smart cards face a multiplicity of new booming applications to address, modifying both its architecture and its use in new environment. At the same time, piracy experience keeps on increasing and inventing efficient techniques to hack new systems. Security techniques, initially limited to the smart card industry, are being re-used in many other sectors, from consumer mass-market products to governmental applications.

Furthermore, the advent of networks made devices inter-connected, exposing major applications to PC world insecurity. E-commerce, high speed Internet and content protection are new technologies demanding strong security features. Smart card experience is re-used, but most platforms possess their own particularities, making it essential to analyse vulnerabilities and innovate in order to propose efficient solutions.

The success and the fast deployment of new services imposed by the digital convergence will be effective if security flaws are lessened. A key to success

Figure 18: Smart card as a secure module



is to find a balance between software, which allows flexibility, and hardware, providing efficient built-in mechanisms.

The world is interconnected. The whole chain needs to be secure; one of the simplest rules is “security is not stronger than its weakest link.”

## REFERENCES

- Allan, A. (2002). *Digital right management software: Perspective*. Retrieved from the World Wide Web: <http://www.gartner.com>.
- Berman, J. (2003). *Commercial piracy report 2003*. Retrieved from the World Wide Web: <http://www.ifpi.org/site-content/library/piracy2003.pdf>.
- Biham, E., & Shamir, A. (1997). *Differential fault analysis of secret key cryptosystem*. CS0910. Haifa, Israel: Electrical Engineering Department, Technion, Israel Institute of Technology.
- Blake, I., Seroussi, G., & Smart, N. (1999). *Elliptic curves in cryptography*. London: Cambridge University Press.
- Boneh, D., DeMillo, R., & Lipton, R. (1997). On the importance of checking cryptographic protocols for faults. In W. Funny (ed.), *Advances in Cryptology-Eurocrypt'97*, Volume 1233 of *Lecture Notes in Computer Science* (pp. 37-51). Berlin: Springer-Verlag.
- commoncriteria.org. (1999). *Common criteria for information technology security evaluation*. Part 1, 2 and 3, version 2.1. Retrieved from the World Wide Web: <http://www.commoncriteria.org/>.
- Daemen, J., & Rijmen, V. (1998). *The Rijndael block cipher: AES proposal: First AES Candidate Conference*. Retrieved from the World Wide Web: <http://csrc.nist.gov/CryptoToolkit/aes/>.

- DatacardGroup. (2001, May). *The transition to multi-application smart cards with post issuance personalization capabilities*. DatacardGroup, Version 1.0.
- dvb.org. (2001). *DVB technical module sub-group on copy protection technologies*. Call for Proposal, Revision 1.2. Retrieved from the World Wide Web: <http://www.dvb.org>.
- Eurosmart. (2003). Retrieved from the World Wide Web: <http://www.eurosmart.com>.
- Gifford, M., McCartney, D., & Seal, C. (1999). Networked biometrics systems: requirements based on iris recognition. *BT Techno.Journal*, 17 (2).
- Heath, S. (1995). *Microprocessor Architectures RISC, CISC and DSP* (2nd ed.). Butterworth-Heinemann.
- Hirst, C., & Phillips, A. (2002). *Microcontroller secure smart cards – Research brief*. Retrieved from the World Wide Web: <http://www.gartner.com>.
- Huang, A. (2002). *Keeping secrets in hardware: the Microsoft Xbox case study*. Retrieved from the World Wide Web: <http://web.mit.edu/bunnie/www/proj/anatak/AIM-2002-008.pdf>.
- ISO13818-1. (1996). ISO/IEC 13818-1: Information Technology, Generic coding of moving pictures and associated audio. Retrieved from the World Wide Web: <http://www.iso.org>.
- ISO7816-3. (1997). *ISO/IEC 7816-3: Electronic signals and transmission protocols*. Retrieved from the World Wide Web: <http://www.iso.org>.
- Kocher, P., Jafe, J., & Jun, B. (1998). Differential power analysis. In *Advances in Cryptology—CRYPTO '99* (LNCS 1666) (pp. 388-397).
- K mmerling, O., & Khun, M.G. (1999). Design principles for tamper-resistant smartcard processors. Retrieved from the World Wide Web: <http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf>.
- Krawczyk, H., Bellare, M., & Canetti, R. (1997, February). *HMAC: Keyed-hashing for message authentication*, RFC 2104.
- Lash, T. (2002). *A study of power analysis and the advanced encryption standard*.
- McLuhan, M., Fiore, Q., & Agel, J. (1968). *War and peace in the global village*. New York: Bantam Books.
- Miller, V.S. (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology. Crypto '85* Berlin: Springer-Verlag.
- NIST. (1994). FIPS Publication 180-1: Secure Hash Standard. Retrieved from the World Wide Web: <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- NIST. (1999). FIPS PUB 46-3: Data Encryption Standard. Retrieved from the World Wide Web: <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>.

- Patent. (2002, December). *Patent Referenced 0216481: Security Module compounded with functional electronic unit, register with temporal variable, and battery.*
- Pescatore, J. (2002). *Software Security is soft security: Hardware is required.* Research Note. Retrieved from the World Wide Web: <http://www.gartner.com>.
- Prabhakar, S., & Jain, A. (2003). Retrieved from the World Wide Web: <http://biometrics.cse.msu.edu/fingerprint.html>.
- Rankl, W., & Effing, W. (1999). *Smartcard handbook* (2<sup>nd</sup> ed.). John Wiley & Sons.
- Rouviere, C., Schmitt, P., & Mouzon, L. (2002). Smart cars, identity crisis. BNP Paribas Equities.
- RSA. (2001). *PKCS#1: RSA Cryptography Standard, Version 2.1* (Draft 2). RSA Laboratories. Retrieved from the World Wide Web: <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/>.
- Schneier, B. (1997). *Applied Cryptography* (2<sup>nd</sup> ed.). New York: John Wiley & Sons.
- smartright.org. (2001). *SmartRight, DVB-CPT-714*. Retrieved from the World Wide Web: <http://www.smartright.org>.
- Svigales, J. (1987). *Smart Cards: The new bank cards*. Macmillan.
- TCG. (2003). *Trusted Computing Group (TCG): Main Specification version 1.1.a*. Retrieved from the World Wide Web: <http://www.trustedcomputing.org>.
- Ubhey, A. (2003). *Potential impacts of a changing value chain*. Frost and Sullivan. Retrieved from the World Wide Web: <http://www.frost.com>.

**SECTION II:**

**AUTHORIZATION  
FRAMEWORKS**

## Chapter IV

# A Flexible Authorization Framework<sup>1</sup>

Duminda Wijesekera, George Mason University, USA

Sushil Jajodia, George Mason University, USA

### ABSTRACT

*Advances in application areas such as Internet-based transactions, cooperating coalitions, and workflow systems have brought new challenges to access control. In order to meet the diverse needs of emerging applications, it has become necessary to support multiple access control policies in one security domain. This chapter describes an authorization framework, referred to as the Flexible Authorization Framework (FAF), which is capable of doing so. FAF is a logic-based framework in which authorizations are specified in terms of a locally stratified rule base. FAF allows permissions and prohibitions to be included in its specification. FAF specifications can be changed by deleting and inserting its rules. We also describe FAF's latest additions, such as revoking granted permissions, provisional authorizations, and obligations.*

### INTRODUCTION

Traditionally, access control plays an integral part in overall system security. Over the years, many different access control models have been developed, and discretionary and mandatory access control models have received consid-

erable attention. Discretionary access control is based on having subjects, objects, and operations as primitives and policies that grant access permissions of the form  $(s,o,a)$ , where subject  $s$  is allowed to execute operation  $a$  on object  $o$ . Mandatory access control is based on having clearance levels for subjects and classification levels for objects as primitives and policies that grant accesses to subjects whose clearance levels dominate those of the objects they access. These models have been used in the commercial and military domains, and implemented in operating systems, database management systems, and object-oriented systems.

Advances in application areas bring new dimensions to access control models. The needs to support multiple access control policies in one security domain, Internet-based transactions, cooperating coalitions, and workflow systems have brought new challenges to access control. In response, new access control models are being proposed to address these emerging needs.

Large numbers of access control models proposed over the years (Dobson, 1989) have been developed with a number of pre-defined policies in mind and thereby have introduced a sense of *inflexibility*. Two alternatives accommodate more than one access control model simultaneously. The first is to have more than one access control mechanism running at the same time, one for each policy. The second is to make access control an application responsibility. The first alternative calls for every application to be closely bound to its access control module, which decreases their portability. The second alternative requires all applications to enforce a consistent access control. Additionally, the responsibility of enforcing access control is vested in applications; it will not impose the same rigorous standards of verification and testing imposed on system code.

Consequently, both alternatives are undesirable. This can be seen by considering a number of access control policies that have been used over the years (Castano, 1995). A popular policy is the *closed world* policy, where accesses that cannot be derived from those explicitly authorized are prohibited. A rarely used alternative is the *open world* policy, where accesses that are not explicitly denied are permitted. Some policies include explicit prohibitions in terms of negative authorizations. This, coupled with generalizations and specializations of these policies to structures such as subject and object hierarchies (Bruggemann, 1992; Rabitti, 1991), yields numerous combinations. Hence, custom creation of policy enforcement mechanisms or passing of these complications to applications is practically infeasible.

One of the solutions for this problem has been to develop flexible authorization models (Jajodia, 2001b), where the flexibility comes from having an access control model that does not depend on any policies or meta policies, but is capable of imposing any of them specifiable in the syntax of the model. One of the main advantages of this approach is that access control can now reside within the system, yet it is able to impose application-specific policies. Given that

there is a need for flexible access control models, the following requirements would be desirable:

- **Expressibility:** It must be possible to model not only existing policies, such as *closed world*, *open world*, and *denials take precedence policies*, but also policies of emerging applications, such as *provisions* and *obligations* (to be discussed shortly).
- **Decoupling Policies from Mechanisms:** The primary need for flexibility is to obtain a policy-independent framework. Hence, policies expressible in such a framework must be enforceable using generic enforcement mechanisms.
- **Conflict Resolution:** Having a flexible framework may invite conflicting policies and, consequently, the framework must be able to facilitate their resolution.
- **Efficiency:** Due to the high frequency of requests coming to access control systems, their processing must be fast. Thus, efficient and simple mechanisms to allow or deny access requests are crucial.

In this chapter, we describe a logic-based framework to specify authorizations in the form of rules referred to as the *Flexible Authorization Framework (FAF)*. In FAF, authorizations are specified using Prolog style rules. These rules may include both positive and negative authorizations. By placing syntactic restrictions on authorization specification rules, FAF ensures that every specification has a unique stable model. In addition, every FAF specification is complete. That is, for every authorization request, FAF either grants it or denies it.

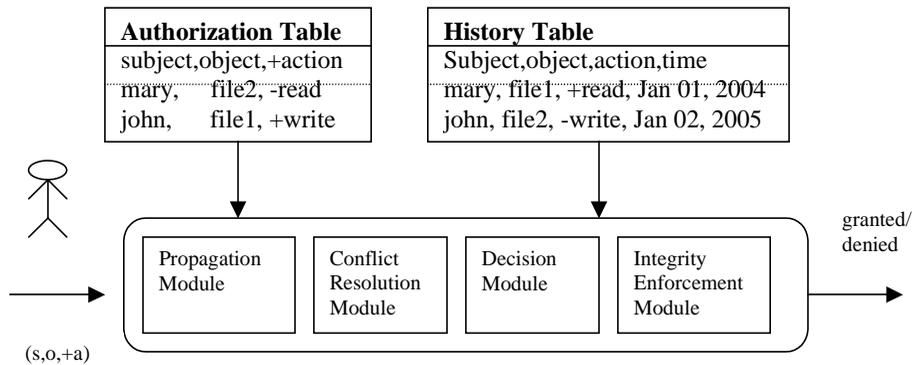
The flexibility in FAF comes by not having any pre-defined meta-policy such as the closed world or the open world policy. The former prohibits underivable permissions and the latter permits underivable prohibitions. In fact, such meta-policies can be specified as FAF rules, making FAF applicable to a large number of application scenarios. Furthermore, by materializing FAF rules, FAF derivations can be made efficient.

Due to the changing nature of applications, it may be necessary to change the rules that specify an authorization policy applicable to an application. We later describe how FAF specifications can be changed by changing the FAF rule base, and how these can affect the materialization.

The dynamic nature of applications may also require the flexibility to revoke already granted permissions. The effect of such permission revocation, including its effect on the materialization, is described.

Not all authorizations are absolute in the sense that an authorization may depend upon the subject satisfying some condition to obtain the access. As an example, a website offering electronic loans may require a potential borrower to register and prove her credit-worthiness to obtain a loan. Once the loan is

Figure 1: FAF system architecture



granted, the borrower is obligated to pay back the loan. The latter is an example where an authorization is granted based on an obligation — a requirement that needs to be met by a subject after the access has been granted. An extension of FAF that incorporates provisions and obligations is described.

## FAF: THE FLEXIBLE AUTHORIZATION FRAMEWORK

The Flexible Authorization Framework (FAF) of Jajodia et al. (Jajodia, 2001; Jajodia, 2001b) is a logic-based framework to specify authorizations in the form of rules. It uses a Prolog style rule base to specify access control policies that are used to derive permissions. It is based on four stages that are applied in a sequence, as shown in *Figure 1*. In the first stage of the sequence, some basic facts, such as authorization subject and object hierarchies (for example, directory structures) and a set of authorizations, along with rules to derive additional authorizations, are given. The intent of this stage is to use structural properties to derive permissions. Hence, they are called *propagation policies*. Although propagation policies are flexible and expressive, they may result in *over specification* (i.e., rules could be used to derive both negative and positive authorizations that may be contradictory). To avoid conflicting authorizations, the framework uses *conflict resolution policies* to resolve conflicts, which comprises the second stage. At the third stage, *decision policies* are applied to ensure the completeness of authorizations, where a decision will be made to either grant or deny every access request. This is necessary, as the framework makes no assumptions with respect to underivable authorizations, such as the closed policy. The last stage consists of checking for integrity constraints, where

all authorizations that violate integrity constraints will be denied. In addition, FAF ensures that every access request is either honored or rejected, thereby providing a built-in completeness property.

FAF syntax consists of terms that are built from constants and variables (no function symbols), and they belong to four sorts: subjects, objects, actions, and roles. We use the notation  $X_s$ ,  $X_o$ ,  $X_a$ , and  $X_r$  to denote respective variables belonging to them, and lower case letters such as  $s$ ,  $a$ ,  $o$ , and  $r$  for constants. For predicates, FAF has the following:

- A ternary predicate  $cando(X_s, X_o, X_a)$ , representing grantable or deniable requests (depending on the sign associated with the action), where  $s$ ,  $o$ , and  $a$  are subject, object, and a signed action term, respectively.
- A ternary predicate  $dercando(X_s, X_o, X_a)$ , with the same arguments as  $cando$ . The predicate  $dercando$  represents authorizations derived by the system using logical rules of inference [modus ponens plus rule of stratified negation (Apt, K. R., 1988)].
- A ternary predicate  $do$ , with the same arguments as  $cando$ , representing the access control decisions made by FAF.
- A 4-ary predicate  $done(X_s, X_o, X_a, X_t)$ , meaning subject  $X_s$  which has executed action  $X_a$  on object  $X_o$  at time  $X_t$ .
- Two 4-ary predicate symbols  $over$ , that takes as arguments two object terms, a subject term, and a signed action term.  $over$  takes as arguments a subject term, an object term, another subject term, and a signed action term. They are needed in the definitions of some of the overriding policies.
- A propositional symbol  $error$  indicating violation of an integrity constraint. That is, a rule with an  $error$  head must not have a satisfiable body.

Other terms and predicates are necessary to model specific applications. In our examples, we use constants  $AOH$ ,  $ASH$  to denote the authorization object and subject hierarchies, respectively. We use a ternary predicate  $in$ , where  $in(x, y, H)$  denotes that  $x < y$  in hierarchy  $H$ . For example,  $in(usr\backslash local, usr, AOH)$  says that  $usr\backslash local$  is below  $usr$  in the authorization object hierarchy  $AOH$ .

Obtaining a locally stratified logic program requires a stratification of rules. FAF is stratified by assigning levels to predicates (literals) as given in *Table 1*, and the level of a rule is the level of its head predicate.<sup>2</sup> As a logic program, any FAF specification gets a local stratification with the level assignment to predicates, as the level of a head predicate is not less than levels of predicates in its body. For any FAF specification  $AS$ ,  $AS_i$  denotes the rules of belonging to the  $i^{\text{th}}$  level.

Because any FAF specification is a locally stratified logic program, it has a unique stable model (Gelfond, 1988), and a well-founded model (as in Gelfond & Lifshitz, 1988). In addition, the well-founded model coincides with the unique stable model (Baral, 1992; Jajodia, 2001b). Furthermore, the unique stable model

Table 1: Strata in FAF specifications

Level	Stratum	Predicate	Rules Defining Predicate
0	AS <sub>0</sub>	hie-predicates rel-predicates done	base relations base relations base relations
1	AS <sub>1</sub>	cando	body may contain done, hie- or rel- predicate
2	AS <sub>2</sub>	dercando	body may contain done, cando, dercando, hie- or rel- predicate. Occurrence of dercando predicate must be positive
3	AS <sub>3</sub>	do	When the head is of the form do(-,-,+a) the body may contain dercando, cando, done, hie- or rel- literals.
4	AS <sub>4</sub>	do	When the head is of the form do(x,y,-a) the body contains only $\neg$ do(x,y,+a)

can be computed in quadratic time data complexity (van Gelder, 1989). See Jajodia (2001b) for details. Following Jajodia (2001b), we use the notation  $M(AS)$  to refer to this unique stable model of specification  $AS$ .

Access requests are frequent in systems. Therefore, processing them using a rule base requires long execution times. In addition, changes to access control specifications are relatively few. Therefore, to optimize processing of these requests, a materialization *architecture* has been proposed in Jajodia (2001b), where instances of derived predicates are maintained. To be able to incrementally update computed materializations upon changes to specifications, Jajodia (2001b) maintains a materialization structure that associates each instance of valid predicates with the rules that directly support its truth. Because predicates belong to strata as stated in *Table 1*, the materialization structure can be constructed in levels corresponding to them.

## MATERIALIZATION

In FAF, an access request must be presented to the rule base in terms of a query of the form  $?do(s,o,+a)$ . In computing the results of this query, the rule-base produces some instances of literals such as *do*, *dercando*, *cando* and possibly others because the rule execution engine needs to backtrack through rule chains. Conversely, if all valid instances of these literals are known, then the execution of the query  $?do(s,o,+a)$  is much faster because backward chaining is eliminated. To facilitate this efficiency, Jajodia (2001b) constructs a materialization structure that stores all valid instances of literals. We now present the materialization structure given in Jajodia (2001). In the following description, we use the notation  $head(r)$  and  $body(r)$  for the head and body of rule  $r$ , respectively.

**Definition 1: (Materialization Structure)**

The materialization structure for an authorization specification  $AS$  is a set of pairs  $(A,S)$ , where  $A$  is a ground atom in the authorization specification language and  $S$  is a set of (indices of) rules whose head unifies with  $A$ .

Definition 2 gives the relationship among a specification, its stable model semantics, and the materialization structure.

**Definition 2: (Correctness of Materialization Structures)**

Let  $AS$  be an authorization specification and let  $MS$  be a materialization structure. We say that  $MS$  *correctly models*  $AS$  if for any pair  $(A,S) \in MS$ , the following conditions hold:

- $A \in M(AS)$  (i.e.,  $A$  belongs to the model of the authorization specification).
- For each  $A \in M(AS)$ , there is at least one pair  $(A,S) \in MS$ .
- For all rules  $r$  such that  $q$  is the most general unifier of  $head(r)$  and  $A$ ,  $r \in S$  iff  $body(r)\theta$ 's existential closure is true in  $M(AS)$ .

According to Definitions 1 and 2, a materialization structure that correctly models an authorization specification  $AS$  contains a pair  $(A,S)$  for each atom  $A$  that is true in the (unique stable) model of  $AS$ , where  $S$  contains indices of the rules that directly support the truth of  $A$ . When instances of atoms are added to or deleted from a specification  $AS$  by adding or removing rules, corresponding changes need to be reflected in its materialization structure so that the updated materialization structure is correct with respect to the updated model. Either adding or removing indices to  $S$  for the set of supporting rules reflects that update. In this situation, an atom will be deleted from the materialization only when its support  $S$  becomes empty. The materialization structure is changed using two algebraic operators  $\oplus$  and  $\otimes$ . Operators  $\oplus$  and  $\otimes$ , respectively, add and remove a pair  $(A, S)$  to/from a materialization structure, and are defined as follows:

**Definition 3: ( $\oplus$  and  $\otimes$ )**

Let  $MS(AS)$  be a materialization structure,  $A$  a ground instance of a literal, and  $S$  a set of rules. Then:

$$\begin{aligned}
 MS(AS) \oplus (A,S) &= MS(AS) \cup \{(A,S)\} \text{ if } \neg \exists (A,S') \in MS(AS) \\
 &= MS(AS) - \{(A,S')\} \cup \{(A, S' \cup S)\} \text{ otherwise.} \\
 MS(AS) \otimes (A,S) &= MS(AS) \text{ if } \neg \exists (A,S') \in MS(AS) \text{ such that } S \cap S' \neq \emptyset \\
 &= MS(AS) - \{(A,S')\} \text{ if } \exists (A,S') \in MS(AS) \text{ such that } S \subseteq S' \\
 &= MS(AS) - \{(A,S')\} \cup \{(A,S' - S)\} \text{ if } \exists (A,S') \in MS(AS) \text{ such that } S \cap S' \\
 &\neq \emptyset \text{ and } S' \neq S
 \end{aligned}$$

Given a materialization structure MS of an authorization specification AS, the model  $M$  of AS is then the projection over the first element of the pairs, written  $M = \prod_1(\text{MS})$ .  $\text{MS}_i$  and  $M_i$  denote the materialization structure and the model at stratum  $\text{AS}_i$ , respectively.

## Computing the Materialization Structure

Computing the unique stable model of an authorization specification AS is an iterative process that at each step  $i$  computes the least model of  $\text{AS} \cup M(\text{AS}_{i-1})$ , where  $M(\text{AS}_{i-1})$  is the least model of the stratum  $\text{AS}_{i-1}$ . The materialization algorithm presented next follows this process.

### Algorithm 1: [The Materialization Algorithm]

*The Base Step of Constructing  $M_0$ :* *hie-*, *rel-*, and *done* predicates are the only ones present in  $\text{AS}_0$ . Hence,  $M_0$  is constructed as the union of these base relations, where  $I_A$  is the set of (indices of) rules that support  $A$ .  $\text{MS}_0 = \{(A, I_A) : A \text{ is a hi, rel- or a done fact}\}$ .

*Inductive Case where  $A_{n+1}$  has no Recursive Rules:* Suppose that we have constructed  $\text{MS}_n$ , and the stratum  $\text{AS}_{n+1}$  does not have recursive rules. Then  $\text{MS}_{n+1}$  is defined as follows, where  $\underline{c}$  refers to a ground instances of the predicate  $p(\underline{x})$ :

$\text{MS}_{n+1} = \oplus\{(p(\underline{c})), \{r\} : r \text{ is a rule in } \text{AS}_{n+1} \theta \text{ is grounding, } \text{head}(r)\theta = p(\underline{c}) \text{ and } \text{MS}_n \Vdash \text{body}(r)\theta\}$

*The Inductive Case where  $A_{n+1}$  has Recursive Rules:*

We use a differential fix-point evaluation procedure as follows:

1. Split the body of each rule  $r \in \text{AS}_n$ , where the first set denoted  $D_r$  contains all the recursive literals and the second set, denoted  $N_r$ , contains all the non-recursive literals of  $r$ . Evaluate the conjunction of the non-recursive literals against  $\prod_1(\text{MS}_0 \cup \dots \cup \text{MS}_n)$ , the materialized model of all strata up to and including  $n$ . Store the result as a materialized view  $V_r$ . Rewrite  $r$  as the rule  $r_{rew}$  given by  $\text{head}(r) \leftarrow V_r \wedge \{A : A \in D_r\}$ . Let  $\text{tr}(\text{AS}_n)$  be the set of all rules  $\{r_{rew} \mid r \in \text{AS}_n\}$ .  $\text{tr}(\text{AS}_n)$  and  $\text{AS}_n$  are logically equivalent [see (Jajodia, 2001b) for the proof]. Hence, we compute the materialization with respect to  $\text{tr}(\text{AS}_n)$  instead of  $\text{AS}_n$ .
2. Let MS be any materialization structure. Define the *program transformation*  $\Phi(\text{AS}_{n+1})$  as follows, where  $\theta$  is a grounding substitution:

$$\Phi(AS_{n+1})(MS) = \{(p(\underline{c}), \{r\}) \mid r_{rew} \in \text{tr}(ASv), \text{head}(r_{rew})\theta = p(\underline{c}), \forall r \cup \Phi_1(MS_n) \Vdash \text{body}(r_{rew})\theta\}$$

3. The set of all materialization structures is a complete lattice with respect to subset inclusion, and the operator  $\Phi(AS_n)$  is monotone and continuous on that lattice. Therefore, by Knaster-Tarski theorem (Tarski, 1955), it follows that  $\Phi(AS_n)$  has a least fixed point. Define  $MS_{n+1}$  to be  $\oplus \text{LFP}(\Phi(AS_n)(\emptyset))$ , where  $\text{LFP}(\Phi(AS_{n+1})(MS))$  denotes the least fixed point of  $\Phi(AS_{n+1})$ .

We can use the above algorithm to materialize any FAF specification, all the way from  $AS_0$  to  $AS_4$ . In using the materialization algorithm as stated, we need to apply the base step at strata 0, recursive steps at strata 2 and 4, and the non-recursive steps at strata 1, 3, and 5. Then, the computation of decision and integrity conflict views would become faster. The following theorem proved in Jajodia (2001b) states that the above procedure is sound and complete.

**Theorem 1: (Correctness of the Materialization Structure)** Let  $AS = \cup\{AS_i; 0 \leq i \leq 4\}$  be an authorization specification, and  $MS_i$  be the materialization structure for stratum  $AS_i$ . Then,  $\cup\{MS_i; 0 \leq i \leq 4\}$  correctly models  $\cup\{AS_i; 0 \leq i \leq 4\}$ .

**Proof:** The proof follows from the fact that each step  $i$  of the construction ensures that  $MS_i$  correctly materializes  $AS_i$ . See Jajodia (2001b) for details.

## CHANGING FAF SPECIFICATIONS

This section discusses the problem of maintaining the materialization of authorization specification upon changes. Changes can be due to changes in user/objects/hierarchical relations ( $AS_0$ ), as well as to modifications in the authorization ( $AS_1$ ), derivation ( $AS_2$ ), and decision ( $AS_3$ ) views, and hence can affect any of  $AS$ 's strata. From the stratification of the program, we are guaranteed that changes to a stratum  $AS_i$  cannot affect stratum below it. The materialization update process exploits this property by incrementally determining the possible changes to the materialization of  $AS_i$  and, iteratively, their effects on the materialization of stratum  $i+1$ . We consider insertion and deletion of new facts or rules into/from the authorization specification. We do not consider updates that modify already existing facts and rules. This is not a restriction, because modifications can be realized in terms of deletions and insertions.

### Inserting Facts

When a new fact is introduced in the authorization specification (stratum  $AS_0$ ), the materialization structure of every stratum might need to be modified.

In principle, the entire authorization specification might change; in practice, it is often the case that the new fact does not have any impact on the extension of any derived predicate. This happens, for example, when the inserted atom is a history fact, whose presence does not fire any derivation rule in addition to those that were already fired before the insertion.

Let us consider the insertion of a base fact  $\delta$  in a *hie-*, *rel-*, or *done-* relation. The process proceeds stratum by stratum, from 0 to 3, computing the required changes. It stops at the first stratum for which no change is recorded (if the model at  $i$  is not affected, neither will the models of strata above  $i$ ). Let  $AS^{old}$  denote the authorization specification before the insertion and  $MS^{old}$  denote its materialization. The new materialization  $MS^{new}$  is defined as follows:

**Step 0:**  $MS_0^{new} = MS_0^{old} \oplus \{(\delta, \_)\}$ . If  $MS_0^{new} = MS_0^{old}$  then terminate the process.

**Step 1:** Let  $CANDO^{new}$  be the set of authorization rules in  $AS_1$  whose body contains at least one literal (either positive or negative) that may be unified with the inserted fact, that is,  $CANDO^{new} = \{cando(\text{tuple}) \leftarrow L_1 \& \dots L_n \text{ s.t. for some } i = 1, \dots, n, \text{ literal } L_i \text{ unifies with } \delta\}$ . Intuitively,  $CANDO^{new}$  is the subset of  $AS_1$  whose extension is potentially affected by the insertion. We then compute the materialization of these rules against the old ( $MS_0^{old}$ ) and the new ( $MS_0^{new}$ ) materializations of the lower stratum and compare them. The materializations are computed as described in the previous section.

Let  $\Delta_{MS_1}^{new}$  be the materialization of  $CANDO^{new}$  evaluated against  $MS_0^{new}$ , and  $\Delta_{MS_1}^{old}$  be the materialization of  $CANDO^{new}$  evaluated against  $MS_0^{old}$ . Compute  $\Delta_{MS_1}^+ = \Delta_{MS_1}^{new} \otimes \Delta_{MS_1}^{old}$ , the set of pairs to be added to  $MS_1$ . Compute  $\Delta_{MS_1}^- = \Delta_{MS_1}^{old} \otimes \Delta_{MS_1}^{new}$ , the set of pairs to be removed from  $MS_1$ . Set  $MS_1^{new} = MS_1^{old} \otimes \Delta_{MS_1}^- \oplus \Delta_{MS_1}^+$ . Let  $M_1^{new} = \Pi_1(MS_0^{new} \cup MS_1^{new})$ , and let  $M_1^{old} = \Pi_1(MS_0^{old} \cup MS_1^{old})$ . If  $MS_1^{new} = MS_1^{old}$  terminate the process.

**Step 2:** Let  $\Delta_{M_1} = (M_1^{new} - M_1^{old}) \cup (M_1^{old} - M_1^{new})$  be the set of atoms in  $M_1$  whose extension has been changed as a consequence of the update. Compute  $DERCANDO^*$  as the set of rules in  $AS_2$  whose firing is potentially affected by the insertion. Note that in the definition of  $DERCANDO^*$  we must take into account the presence of recursion. In addition to the rules whose body contains a literal defined in the lower strata and whose truth could have changed in the corresponding models, we must also consider those rules that possibly depend on the update through recursion. To perform this dependency check, we refer to the original rules in  $AS_2$ . Every time  $AS_2$  is materialized, a new rewritten version of the potentially affected rules are constructed, since the materialization of the conjunction of non-recursive literals might change.

$\text{DERCANDO}^* = \text{dercando}(\text{tuple}) \leftarrow L_1 \ \& \ \dots \ \& \ L_n$  s.t. for some  $i = 1, \dots, n$ , literal  $L_i$  unifies with an atom in  $\Delta_{\text{MS1}}$  or with the head of a rule in  $\text{DERCANDO}^*$ , or  $\text{dercando}(\text{tuple})$  unifies with the head of a rule in  $\text{DERCANDO}^*$ . Let  $\Delta_{\text{MS2}}^{\text{new}}$  and  $\Delta_{\text{MS2}}^{\text{old}}$  be the materializations of  $\text{DERCANDO}^*$  with respect to  $M_1^{\text{new}}$  and  $M_1^{\text{old}}$ , respectively. These materializations can be computed as described in step 3 of the previous section.

Compute  $\Delta_{\text{MS2}}^+ = \Delta_{\text{MS2}}^{\text{new}} \otimes \Delta_{\text{MS2}}^{\text{old}}$ , the set of new derivations made possible by the insertion of  $\delta$ . Compute  $\Delta_{\text{MS2}}^- = \Delta_{\text{MS2}}^{\text{old}} \otimes \Delta_{\text{MS2}}^{\text{new}}$ , the set of derivations blocked by the insertion of  $\delta$ . Set  $\Delta_{\text{MS2}}^{\text{new}} = \Delta_{\text{MS2}}^{\text{old}} \otimes \Delta_{\text{MS2}}^- \oplus \Delta_{\text{MS2}}^+$ . Let  $M_2^{\text{new}} = \Pi_1(\text{MS}_0^{\text{new}} \cup \text{MS}_1^{\text{new}} \cup \text{MS}_2^{\text{new}})$  and let  $M_2^{\text{old}} = \Pi_1(\text{MS}_0^{\text{old}} \cup \text{MS}_1^{\text{old}} \cup \text{MS}_2^{\text{old}})$ . If  $M_2^{\text{new}} = M_2^{\text{old}}$ , terminate the process.

**Step 3:** Derived predicates defined in stratum  $\text{AS}_3$  are non-recursive. Thus, we can follow the same technique discussed in step 1, referring to the facts in the already computed sets  $M_2^{\text{new}}$  and  $M_2^{\text{old}}$  to evaluate the derivation increments and decrements.

Some comments on the effectiveness of the method discussed above are in order. First, we comment on the *frequency* of the updates. While updates to *hie*- and *rel*- predicates are not likely to be frequent, updates to *done* may be very frequent (the main reason for materializing the model of the specification is that access requests are far more frequent than administrative requests). One may assert that, given that the history is recorded using the *done* predicate, any access to the system on which the authorization specification is defined may result in an insertion of a new *done* fact, implying a change to the specification and, as a consequence, changes to the specifications may be far greater in number than the access requests. The correctness of the above-stated procedure, stated in Theorem 2, is proved in Jajodia (2001b).

**Theorem 2: (Correctness of the insertion procedure)** Let  $\text{MS}^{\text{old}}$  be the materialization structure that correctly models  $\text{AS}$ . When a base fact  $\delta$  is inserted, the update procedure transforms  $\text{MS}^{\text{old}}$  into  $\text{MS}^{\text{new}}$  such that  $\text{MS}^{\text{new}}$  correctly models  $\text{AS} \cup \{\delta\}$ .

**Proof:** See Jajodia (2001b)

The following Lemma given in Jajodia (2001b) shows that updating FAF rules takes only polynomial time.

**Lemma 1: (Complexity of the insertion procedure)** Suppose that  $\text{AS}$  is an authorization specification whose base relations predicates (*hie*, *rel*, *done*) are all part of the input, and that  $A$  is any atom being inserted. Our update procedure has polynomial data complexity.

**Proof:** See Jajodia (2001b).

## Inserting Rules

The iterative process we have introduced to handle the insertion of base facts in the materialized authorization specification is general enough to handle also inserting rules as well as deleting both facts and rules.

### *Inserting Cando and Do Rules*

When a rule is inserted in stratum  $AS_1$ , (resp. stratum  $AS_3$ ) (i.e., in a stratum without recursion) the process described above is applied, starting from step 1 (resp. step 3), and letting  $CANDO^{new}$  (resp.  $DO^{new}$ ) (i.e., the sets of rules potentially affected by the insertion) contain exactly the inserted rule. The algorithm checks whether the new rule allows the derivation of new atoms. If the answer is yes, the update to the considered stratum is propagated to the upper levels; otherwise, the algorithm stops.

### *Inserting Dercando Rules*

If a rule is inserted in stratum  $AS_2$ , then the update process is started at step 2, letting  $DERCANDO^* = \text{dercando}(\text{tuple}) \leftarrow L_1 \& \dots \& L_n$  such that for some  $i = 1 \dots n$ , literal  $L_i$  unifies with the head of the inserted rule or with the head of a rule in  $DERCANDO^*$ . Intuitively,  $DERCANDO^*$  contains all the rules that could fire because of the presence of the new rule introduced. Correctness of the approach comes from the fact that recursion can only be positive; thus, the immediate consequence operator  $AS_2$  is monotonic. Hence, the authorizations that become true on the basis of the new rule cannot block any previously firing rule in the same stratum.

It is easy to see that insertion of *cando* and *dercando* rules into an authorization specification preserves the (polynomial) data complexity results we have obtained earlier.

## Deleting Facts and Rules

We distinguish deletion of base facts, deletion of rules in strata that do not allow recursion, and deletion of rules in the potentially recursive stratum.

### *Deleting Base Facts*

To handle the deletion of a base fact  $\delta$  from the authorization specification, we apply the update method starting from step 0 and letting  $MS_0^{new} = MS_0^{old} \otimes \{(\delta, \_)\}$ . This step can obviously be executed with polynomial data complexity.

### *Deleting Cando and Do Rules*

The deletion of a rule from  $AS_1$  (or  $AS_3$ ) is taken care of with a method that is symmetric to the insertion: the set  $CANDO^{new}$  ( $DO^{new}$ , respectively), containing exactly the removed rule, is considered as the set of rules potentially affected

by the update, and its materialization  $\Delta_{MS_1}^-$  ( $\Delta_{MS_3}^-$  respectively) is computed.  $D_{MS_1}^-$  contains the derived atoms that were supported by the removed rule. There is no need to calculate  $\Delta_{MS_1}^{new}$  and  $\Delta_{MS_1}^+$ , because  $\Delta_{MS_1}^+$  is always empty given that removing a rule from  $AS_i$  cannot cause new atoms to be added to  $MS_i$ . Thus,  $MS_i^{new} = MS_i^{old} \otimes \Delta_{MS_i}^-$  is computed. If  $\Pi_1(MS_i^{new}) \neq \Pi_1(MS_i^{old})$ , then the control is passed to the next step. Note that it is not necessary to look at  $M_i$  (which also contains atoms in the models of lower strata), since nothing has changed at levels below  $i$ . Note also that facts that were supported by the deleted rule will still belong to the model if there are other rules supporting them. A fact is removed from the materialization structure (and hence from the model) only when its set of supporting rules becomes empty.

Deleting *cando* and *do* rules is a polynomial process because computing the materializations  $\Delta MS_1^-$  and  $\Delta MS_3^-$  is polynomial (as materializing a rule has polynomial data complexity by our previous results). Computing  $MS_i^{new} = MS_i^{old} \otimes \Delta_{MS_i}^-$  is also polynomial (in fact quadratic in the sizes of the relations involved). Checking if  $\Pi_1(MS_i^{new}) \neq \Pi_1(MS_i^{old})$  is also polynomial.

### Deleting Dercando Rules

Deleting *dercando* rules is based on a technique similar to the approach adopted in the corresponding step of the insertion algorithm. In this case, recursion is taken into account for defining a set  $DERCANDO^*$  as follows:

1.  $r \in DERCANDO^*$ , where  $r$  is the rule to be deleted.
2. If  $DERCANDO^* = \text{dercando}(\text{tuple}) \leftarrow L_1 \ \& \ \dots \ \& \ L_n$  s.t. for some  $i = 1, \dots, n$ , literal  $L_i$  unifies with the head of a rule in  $DERCANDO^*$  or  $\text{dercando}(\text{tuple})$  unifies with the head of a rule in  $DERCANDO^*$ , then  $r' \in DERCANDO^*$ .

$DERCANDO^*$  contains all the rules possibly connected to the deleted rule. For any rule in  $DERCANDO^*$ , its head unifies with the head of some rule whose firing might depend on the head predicate of the deleted rule. Instead, the deletion does not have any effects on the model of  $AS_2$  -  $DERCANDO^*$ . It is important to note that  $DERCANDO^*$  contains an *overestimate* of the set of rules to be deleted. Some rules in  $DERCANDO^*$  may not actually need to be deleted.

Let  $NEWDERCANDO^* = DERCANDO^* - \{r\}$ . The materializations  $MS_{DERCANDO^*}$  and  $MS_{NEWDERCANDO^*}$  of  $DERCANDO^*$  and  $NEWDERCANDO^*$ , respectively, are computed. Only the pairs  $MS_{DERCANDO^*} \otimes MS_{NEWDERCANDO^*}$  are effectively removed from the materialization structure.

The complexity of computation  $DERCANDO^*$  is obviously quadratic in the size of our authorization specification. By our previous complexity result on materialization, we can see that the materializations  $MS_{DERCANDO^*}$  and  $MS_{NEWDERCANDO^*}$  can be computed with polynomial data complexity. Finally,

computing  $MS_{\text{DERCANDO}}^* \otimes MS_{\text{NEWDERCANDO}}^*$  is also immediately seen to be polynomial. Now consider an example FAF with the following rules:

$$\begin{aligned} r_1: & \text{dercando}(tuple1) \leftarrow \text{dercando}(tuple2) \\ r_2: & \text{dercando}(tuple1) \leftarrow \text{non-recursive-body} \\ r_3: & \text{dercando}(tuple3) \leftarrow \text{dercando}(tuple1) \\ r_4: & \text{dercando}(tuple4) \leftarrow \text{dercando}(tuple2) \\ r_5: & \text{dercando}(tuple2) \leftarrow \text{non-recursive-body} \end{aligned}$$

Assume that the existential closures of the bodies of non-recursive rules are all satisfied, and that  $tuple1$ ,  $tuple2$ , and  $tuple3$  are ground, distinct tuples. Thus:

$$MS_2 = \{(\text{dercando}(tuple2), \{r_5\}), (\text{dercando}(tuple1), \{r_1, r_2\}), (\text{dercando}(tuple3), \{r_3\}), (\text{dercando}(tuple4), \{r_4\})\}$$

Suppose the system security officer (SSO) requests that rule  $r_1$  be removed.

$$\begin{aligned} \text{DERCANDO}^* &= \{r_1, r_2, r_3\}, \text{ and } \text{NEWDERCANDO}^* = \{r_2, r_3\}. \\ MS_{\text{DERCANDO}}^* &= \{(\text{dercando}(tuple1), \{r_1, r_2\}), (\text{dercando}(tuple3), \{r_3\})\} \\ MS_{\text{NEWDERCANDO}}^* &= \{(\text{dercando}(tuple1), \{r_2\}), (\text{dercando}(tuple3), \{r_3\})\} \\ MS_{\text{DERCANDO}}^* \otimes MS_{\text{NEWDERCANDO}}^* &= \{(\text{dercando}(tuple1), \{r_1\})\}. \end{aligned}$$

Thus, only  $(\text{dercando}(tuple1), \{r_1\})$  is removed from the materialization structure, which then becomes  $MS_2^{\text{new}} = \{(\text{dercando}(tuple2), \{r_5\}), (\text{dercando}(tuple1), \{r_2\}), (\text{dercando}(tuple3), \{r_3\}), (\text{dercando}(tuple4), \{r_4\})\}$ .

The head of  $r_1$  unifies with the body of rule  $r_3$ . In the incremental approach, we would have  $r_3 \in \text{DERCANDO}^*$ , and we therefore remove the pair  $(\text{dercando}(tuple3), \{r_3\})$  from  $MS_2$ . However, this would not be necessarily correct, since the atom  $\text{dercando}(tuple1)$  could be derived through the nonrecursive rule  $r_2$ , and hence  $\text{dercando}(tuple3)$  would still be supported by  $r_3$ .

Because of the problem illustrated, the easiest thing to do in case of deletion of  $\text{dercando}$  rules to recompute the materialization for stratum  $AS_2$ . After that, the materialization of  $AS_3$  can be updated incrementally. Note that the materializations of  $AS_0$  and  $AS_1$  remain unvaried and hence do not need to be recomputed.

## REVOKING ACCESS PERMISSIONS GRANTED BY FAF

Due to the evolving nature of information systems, permissions once granted may need to be revoked for legitimate reasons. Revoking access

permissions in a discretionary access control model that is specified by a stratified rule base is the subject of this section. In revoking access permissions derived from a stratified rule base, we have to address two main issues. The first is to explore the semantics of revoking permissions. The second is to address limitations imposed by the representational framework used to specify access permissions and the effect that removing one permission has on others. The work reported here appears with details in Wijesekra (2001)

Intuitively, the semantics of permission removal must address the history and cause(s) of their original granting. Possibilities here are that they could have been given directly, or granted by some other subject that has the authority to grant them, or were granted based on the same subject already having some other permissions.

The semantics of permission removal associated with conditional grants and granting permissions with grant options are discussed in detail in Hagstrom (2001). We do not address this issue here.

Complementary to the semantics of removing permissions with grant options, there are many specification methods for access control. Some of these methods permit explicit prohibitions (i.e., negative permissions) instead of having some built-in default closed world policy of denying permissions unless they are explicitly granted. In addition, the permission specification methodology may introduce additional dependencies not resulting from intuitive semantics of removal addressed in Hagstrom (2001). The following examples show two sets of rules granting access permission  $a$  on object  $o$  to subjects  $s_1$  and  $s_2$ .

## Representational Issues in Permission Removal

Below, the ternary predicate  $canExec(s,o,a)$  means the subject  $s$  is allowed to execute action  $a$  on object  $o$ . Consider the following two options for specifying the granting of  $s_1$  and  $s_2$  the permission to execute  $a$  on  $o$ :

First Option

$dercando(s_1,o,a).$

$dercando(s_2,o,a) \leftarrow dercando(s_1,o,a).$

Second Option

$dercando(s_1,o,a).$

$dercando(s_2,o,a).$

The difference between the first and second option in the above example is that the first only lists rules to derive permissions and the second lists their conclusions. The question now is *if permission to execute  $a$  on object  $o$  is withdrawn from subject  $s_1$ , should subject  $s_2$  retain its permissions?* One may argue that the first option explicitly states conditional permissions, and consequently, when subject  $s_1$  loses permission  $a$  on object  $o$ , so must subject  $s_2$ .

We call this *deep removal*. Conversely, a case can be made that conditionals such as those listed in the first option are a concise way of stating their final conclusions and hence should not be used in backward propagation of permission removal. We call this option *shallow removal*. We refer to such issues arising as a consequence of the representational mechanism as representational dependencies. Our objective is to provide a representation-independent permission removal.

If authorizations are derived using a rule base and an access permission that is a consequence of a given set of rules is removed, then either the rule base must be changed to reflect the removed permission or the new set of permissions may not be consequences of the old rule base, thereby introducing inconsistency. Consequently, we choose to minimally *alter* the rule base to reflect removed permissions — in a sense of minimality described shortly — so that the consistency of the rule base will not be lost due to permission removal. We illustrate the issues involved in the following example.

### Shallow vs. Deep Removal

1. The Original Specification:
  - dercando( $s_1, o, a$ ),
  - dercando( $s_2, o, a$ )  $\leftarrow$  dercando( $s_1, o, a$ ).
  - dercando( $s_3, o, a$ )  $\leftarrow$  dercando( $s_2, o, a$ ).
2. Rules after Shallow Removal of canExec( $s_2, o, a$ ) :
  - dercando( $s_1, o, a$ ),
  - dercando( $s_3, o, a$ )  $\leftarrow$  dercando( $s_1, o, a$ ).
3. Rules after Deep Removal of canExec( $s_2, o, a$ ):
  - dercando( $s_1, o, a$ ).

Shallow removal of permission  $a$  from  $s_2$  on  $o$ , dercando, in the given example results in a rule base where dercando( $s_1, o, a$ ) and dercando( $s_3, o, a$ ) can be inferred but not dercando. The deep removal of dercando results in removing all consequences of it, resulting in only having dercando( $s_1, o, a$ ) in the rule base. Nevertheless, deep removal, while appearing to be logical, has the undesirable side effect of being explicitly dependent on stated rules and is not invariant under rule derivations. In the example, the rule canExec( $s_3, o, a$ )  $\leftarrow$  dercando( $s_1, o, a$ ) is a rule derived from dercando( $s_2, o, a$ )  $\leftarrow$  dercando( $s_1, o, a$ ) and dercando( $s_3, o, a$ )  $\leftarrow$  dercando( $s_2, o, a$ ). If it were present in the original rule base, dercando( $s_3, o, a$ ) would still be valid in the rule base with the deep removal of dercando( $s_2, o, a$ ). Thus, one way to remove the dependence of deep removal's semantics on the representative rule set is to deductively close the rule base before applying the removal operation (such as altering rules to reflect the removal). But then, the difference between shallow and deep removal disappears. This is shown in the

example whereby deductively closing the rule base and then deeply removing  $\text{dercando}(s_1, o, a)$  neither  $\text{dercando}(s_1, o, a)$  nor  $\text{dercando}(s_3, o, a)$  is removed, but only  $\text{dercando}(s_2, o, a)$ .

Now we discuss the issue of shallow removal of any literal  $A$  at any stratum  $AS_i$  for  $i > 0$  from any specification  $AS$ , which should result in a new specification  $AS^{\text{new}}$ , whose model  $M(AS^{\text{new}})$  differs from the model of  $AS$ ,  $M(AS)$ , because of:

1. The absence of  $A$ ,
2. The presence of all the literals whose derivations were all blocked, in  $M(AS)$ , by some negative literal that unifies with  $A$ . This can happen when a positive literal is removed and therefore its negative counterpart becomes true due to the rule of stratified negation.

Issues involved in doing so are illustrated in the next example, where  $s_1, s_2,$  and  $s_3$  are subject constants,  $X_s, X_o,$  and  $X_a$  are subject, object, and action variables, respectively, and  $AOH$  is the object hierarchy.  $o, o'$  are object constants satisfying  $o <_{AOH} o'$ , and  $a$  is an action constant.

### Issues in Shallow Removal of Permissions in FAF:

$$\text{in}(o, o', AOH) \leftarrow \quad (1)$$

$$\text{cando}(s_2, o, a) \leftarrow \text{in}(o, o', AOH) \quad (2)$$

$$\text{dercando}(s_3, o, +a) \leftarrow \neg \text{cando}(s_2, o, +a) \quad (3)$$

$$\text{dercando}(X_s, o, +a) \leftarrow \text{cando}(X_s, o, +a) \quad (4)$$

$$\text{dercando}(s_1, o, +a) \leftarrow \text{dercando}(s_2, o, +a) \quad (5)$$

$$\text{do}(X_s, X_o, +a) \leftarrow \text{dercando}(X_s, X_o, +a) \quad (6)$$

$$\text{do}(X_s, X_o, -a) \leftarrow \neg \text{do}(X_s, X_o, +a) \quad (7)$$

From rules 1 through 7, subjects  $s_1$  and  $s_2$  are allowed to perform the operation  $a$  on object  $o$ . Now, shallow removal of  $\text{cando}(s_2, o, a)$  from  $M(AS)$  requires the following:

**Derivation Chain 1:** When the predicate instance  $\text{cando}(s_2, o, a)$  is removed, the consequent of rule 2 should become false. Because the antecedent of rule 2,  $\text{in}(o, o', AOH)$  is true by rule 1, the rule itself needs to be removed.

**Derivation Chain 2:** Due to  $\text{cando}(s_2, o, a)$  being removed, rules 4 and 5 cannot be used to derive  $\text{dercando}(s_1, o, a)$ . Because no other rules can be used to derive  $\text{dercando}(s_1, o, a)$ ,  $\text{do}(s_1, o, a)$  cannot be derived either, as rule 6 is the only one that can be used to derive an instance of  $\text{do}$ . Hence,  $\text{do}(s_1, o, -a)$  can be derived from rule 7.

**Derivation Chain 3:** The only way  $do(s_2, o, a)$  can be derived is by using rule 6, but it requires a derivation of  $dercando(s_2, o, a)$ . Notice that  $dercando(s_2, o, a)$  can only be derived by using rule 4, which requires  $cando(s_2, o, a)$  as an antecedent. Thus, when  $cando(s_2, o, a)$  is removed,  $do(s_2, o, a)$  cannot be derived. Hence, by the non-monotonic rule 7,  $do(s_2, o, -a)$  can be derived.

**Derivation Chain 4:** Removing  $cando(s_2, o, a)$  makes the antecedent of rule 3 true and hence,  $dercando(s_3, o, a)$  becomes true. Hence,  $do(s_3, o, a)$  becomes true due to rule 6.

Before the shallow removal of  $cando(s_2, o, a)$ , subjects  $s_1$  and  $s_2$ , but not  $s_3$ , are allowed to execute the operation  $a$  on object  $o$ . After shallow removal of  $cando(s_2, o, a)$ ,  $s_3$  is permitted operation  $a$  on object  $o$  but not  $s_1$  and  $s_2$ .

As stated,  $cando(s_2, o, a)$  has to be false in any new model  $M(AS^{new})$ . Also, derivation chain 5 is an example of a blocked literal due to the removal of another literal in rule 3. Nevertheless, consider the following rules, derived from those in the example. For example, rules 2, 4 and 6 derive rule 8 below.

$$\begin{aligned} do(s_2, o, +a) &\leftarrow in(o, o', AOH) & (8) \\ dercando(s_1, o, +a) &\leftarrow in(o, o', AOH) & (9) \\ do(s_1, o, +a) &\leftarrow in(o, o', AOH) & (10) \\ do(s_1, o, +a) &\leftarrow & (11) \\ do(s_2, o, +a) &\leftarrow & (12) \\ do(s_3, o, -a) &\leftarrow & (13) \\ cando(s_2, o, +a) &\leftarrow & (14) \end{aligned}$$

Rules 2, 4 and 6 derive rule 8. If we use informal reasoning similar to that used in derivations chain 1 through 4 of the example, derived rule 8 will still be admissible even after removing  $cando(s_2, o, +a)$ . Consequently, if derived rules 8 through 14 were included in the original rule set,  $s_2$  will still be permitted to access object  $o$  after removing  $cando(s_2, o, +a)$ . Thus, to obtain representation-independent semantics for permission removal, we must consider all derived rules. Hence, we formalize the notion of *shallow removal with minimal changes* as follows.

**Definition 1: (Shallow Removal with Minimal Changes)**

An authorization specification  $AS^{new}$  is a shallow removal of literal  $A$  from  $AS$  with minimal changes if it satisfies the following conditions:

1.  $\mathcal{D}(AS^{new})$ , the deductive closure of  $AS^{new}$ , must not contain any ground instance  $A\theta$  of  $A$ . That is,  $\mathcal{D}(AS^{new}) \cup \{A\theta : \text{where } A\theta \text{ is a ground instance}\} = \emptyset$ .

2. Every rule  $r$  (and hence atomic instance) that is derivable from  $AS$  but has no terms unifying with  $A$  must be derivable from  $AS^{new}$ . That is,  $D(AS^{new}) \cup \{r \text{ is a rule where no term in } r \text{ unifies with } A\theta\} \subseteq \mathcal{D}(AS^{new})$ , where  $A\theta$  is an instance of  $A$ .
3.  $\mathcal{D}(AS^{new})$  must contain every rule (and hence atomic instance) derivable from  $\mathcal{D}(AS) - \{r \in \mathcal{D}(AS) \text{ where the head of } r \text{ is an instance of } A\}$ . That is,  $\mathcal{D}(AS) - \{r \in \mathcal{D}(AS) \text{ the head of } r \text{ is an instance of } A\} \subseteq \mathcal{D}(AS^{new})$ .

The intent of Definition 1 is to remove all instances of the removed literal (as given in condition 1), to retain rules that did not involve the removed literal and derivable before permission removal (as given in condition 2 of Definition 1), and to be able to derive all rules that were blocked in the original derivation due to the presence of the removed literal (as given in condition 3 of Definition 1). To show that Definition 1 captures the motivation provided at the beginning of this section in terms of models, we show that any  $AS^{new}$  that is a minimal removal of  $AS$  satisfies these properties.

**Theorem 3: (Equivalence of Syntactic and Semantic Definitions)** If  $AS^{new}$  is a specification that is a shallow removal of literal  $A$  from  $AS$ , then the unique stable models  $M(AS^{new})$  and  $M(AS)$  satisfy the following properties:

- For all ground instances  $A\theta$ ,  $A\theta \in M(AS^{new})$ .
- $M(AS) - M(AS^{new}) \subseteq \{A\theta : A\theta \text{ is a ground instance of } A\}$ .
- For any rule  $r$  that does not have any term unifying with  $A$ , if  $r \in M(AS)$  then  $r \in M(AS^{new})$ .

Conversely, any pair of specifications  $AS$  and  $AS^{new}$  satisfying the above properties where  $A \in M(AS)$  satisfies shallow removal of  $A$  from  $AS$  to obtain  $AS^{new}$ .

**Proof:** See Wijesekra (2001).

Next, we show some desirable consequences of our definition of removal. First, we show that for any specification  $AS$  and literal  $A$ , a new specification  $AS^{new}$  can be found so that  $AS^{new}$  minimally removes  $A$  from  $AS$  in the sense of Definition 1. Second, we show that the result of a minimal removal is immune from the presence of a derived rule and it is unique up to derivable ground literals.

**Procedure 1: (Procedure for Shallow Removal)** Given a specification  $AS$  and a literal  $A$  to be removed, the following procedure produces a specification  $AS^{new}$ , which is a shallow removal of  $A$  from  $AS$ :

1. If the literal to be removed is  $\text{do}(s,o,-a)$ , where the action term is negative, then add  $\text{do}(s,o,+a)$  as a new rule. Otherwise, proceed as follows.
2. Replace any rule  $r$  of AS that has a literal unifying with  $A$  by all of its ground instances. Name the new specification  $AS'$ .
3. For any pair of rules  $r_1$  and  $r_2$  of  $AS'$ , where a ground instance  $A\theta$  of the literal  $A$  is in the tail of  $r_1$  and the head of  $r_2$ , replace all ground occurrences of  $A$  in the tail of  $r_1$  by the body of  $r_2$  recursively as follows:
  - a. Let  $B_0$  be  $AS'$ . For every integer  $n$ , define  $B_{n+1}$  from  $B_n$  as follows:  
Let  $C_n$  be the set of all derived rules obtainable by replacing all ground occurrences of  $A\theta$  in the tail of  $r_1$  by the body of  $r_2$  for some rules  $r_1, r_2 \in B_n$ , where the head of  $r_2$  and a literal in the body of  $r_1$  are the unifying ground instance  $A\theta$  of  $A$ . Let  $B_{n+1}$  be  $B_n \cup C_n$ .
  - b. Define  $AS''$  as  $\cup\{B_n : n \geq 1\}$ .
4. Now, remove all rules whose heads are an instance of  $A$  in  $AS''$ . Name the specification  $AS^{\text{new}}$ .

$AS'$  obtained in step 2 is logically equivalent (i.e., have the same deductive closure) to the original specification  $AS$ , as this step adds all instances of rules that have a literal unifying with  $A$ . Also,  $AS''$  obtained in step 3 is equivalent to  $AS'$ , as this step adds rules derivable from  $AS'$ .

**Theorem 4: (Correctness of the Shallow Removal Procedure)**  $AS^{\text{new}}$  constructed in Procedure 1 is a minimal shallow removal of  $A$  from the specification  $AS$ .

**Proof:** See Wijesekera (2001).

Theorem 5 shows that the construction given in Procedure 1 is independent of the choice of rules  $AS$  selected to represent  $\mathcal{D}(AS^{\text{new}})$ .

**Theorem 5: (Independence from Derived Rules and Uniqueness of  $AS^{\text{new}}$ )** Suppose  $AS_a$  is  $AS \cup \{r\}$  where the rule  $r$  can be derived from  $AS$ , and  $AS^{\text{new}}$   $AS_a^{\text{new}}$ , results from minimal shallow removals of literal  $A$  from  $AS$  and  $AS_a$ , respectively. Then, for any ground literal  $p$ ,  $p \in \mathcal{D}(AS^{\text{new}})$  iff  $p \in \mathcal{D}(AS_a^{\text{new}})$ . Furthermore, if any  $AS_c^{\text{new}}$  and  $AS_d^{\text{new}}$  are minimal shallow removals of literal  $A$  from  $AS$  then,  $p \in \mathcal{D}(AS_c^{\text{new}})$  iff  $p \in \mathcal{D}(AS_d^{\text{new}})$ . (Here,  $AS_a$ ,  $AS_c$ , and  $AS_d$  refer to different specifications and should not be confused with strata of  $AS$ , which are denoted by  $AS_1$ ,  $AS_2$ ,  $AS_3$  and  $AS_4$ .)

**Proof:** See Wijesekera (2001).

Given an authorization specification  $AS$  and a literal  $A$  to be removed, Procedure 1 constructs  $AS^{new}$  that is a minimal shallow removal of  $A$  from  $AS$ . The construction of  $AS^{new}$  as stated is quite inefficient because it introduces all ground instances of rules at the first step. A method that removes that inefficiency is described in Wijesekra (2001).

## ADDING PROVISIONS AND OBLIGATIONS TO FAF

Traditional static access control policies that provide only “yes/no” decisions in response to user access requests are too inflexible to meet the complex requirements of newer applications such as business-to-business or business-to-consumer applications.

To illustrate, consider a loan application and management (payment collection, etc.) system. It allows users to initiate a loan application process if they are already registered in the system. If they are not already registered, they are given an opportunity to register with the system by supplying the necessary information and, if this step is successful, they are given permission to proceed with the loan application process. Note that here the initiation of the loan application is not a statically assigned permission to the users. Users are given the permission to apply for a loan as long as they satisfy some conditions; if they do not satisfy these conditions, they are given a chance to perform certain actions to satisfy them.

Continuing with the example, assume a loan application is approved. In this case, the applicant will have access to the funds under the condition that the user agrees to pay off the loan according to a certain payment schedule. Here again, such a condition is different from a statically assigned permission in the sense that the user promises to satisfy certain obligations in the future, and the system needs to be able to monitor such obligations and take appropriate actions if the obligations are not met.

From the example, we see that policies in many applications are complex, and a system requires flexible and powerful mechanisms to handle conditions and actions before and after certain decisions (access to the loan funds by the applicant in the example). Since the two sets of conditions and actions are conceptually different and require different management techniques, we distinguish between them by calling them *provisions* and *obligations*, respectively. Intuitively, provisions are specific actions to be performed before the decision is taken, and obligations are actions that should be taken after a favorable decision is taken. In this section we formulate these concepts and incorporate them in FAF. We also present the essential structure of the formal model, and refer to Bettini (2002a, 2002b) for details.

Because we model obligations as actions that must be fulfilled after the access control decision is made, the system needs to monitor the progress of

obligation fulfillment. Furthermore, if obligations are not fulfilled, the system must be able to take compensatory actions. For these purposes, we introduce compensatory actions. For the remainder of this section, we use the abbreviation PO for provisions and obligations.

We represent by two disjoint sets of predicate symbols  $\mathbf{P}$  and  $\mathbf{O}$  that are also disjoint from the set of predicate symbols  $\mathbf{Q}$  allowed in the policy rule specification language. On the contrary, the set of variable and constant symbols  $\mathbf{V}$  and  $\mathbf{C}$  admitted in the predicates is the same as that used in the policy rules. The predicate symbols in  $\mathbf{P}$  and  $\mathbf{O}$  may be of any nonnegative arity.

An atom is either one of the symbols  $\leftarrow$ ,  $\perp$  or a predicate  $P_i(t_1, \dots, t_k)$  with  $P_i \in \mathbf{P}$  or  $O_i(t_1, \dots, t_k)$  with  $O_i \in \mathbf{O}$  and each  $t_i$  is either a constant from  $\mathbf{C}$  or a variable from  $\mathbf{V}$ . When not clear from the context, we distinguish these atoms from those in the policy by calling them PO-atoms. Then, a PO-formula is either a PO-atom, or a disjunction of PO-formulas, or a conjunction of PO-formulas. A PO-atom is ground if it is variable-free, and a PO-formula is *ground* if each of the atoms in the formula is ground. An interpretation  $I$  of a PO-formula is a mapping from each ground atom to the constant *True* or *False*, with the atoms  $\leftarrow$  and  $\perp$  mapping to the constants *True* and *False*, respectively. The satisfaction of a PO-formula is defined inductively on the structure of the formula, as usual, considering ground atoms as a basis and the conjunction and disjunction operators that appear in the formula. Later we give a detailed syntax for specification of obligations, here simply given as a predicate.

For each policy rule  $R_i$  in  $R$  there is an associated PO-formula, denoted by  $F(R_i)$ , representing the PO for that rule. We also impose the intuitive constraint that each variable appearing in  $F(R_i)$  must appear in the body of  $R_i$ . Note that because these predicates are not part of the policy rule specification (the datalog program), they do not appear in its model  $M_R$ . An example, a FAF specification with provisions and obligations, is as follows:

$$Q_1(x) \quad \leftarrow \quad Q_2(x,y), Q_3(y) \quad :O_1(s,x,y) \quad (15)$$

$$Q_2(a,b) \quad \leftarrow \quad P_1(b) \quad :P_1(b) \quad (16)$$

$$Q_3(b) \quad \leftarrow \quad : \quad (17)$$

$$Q_1(y) \quad \leftarrow \quad Q_4(z,y,c) \quad :P_2(y,a) \wedge P_3(a) \wedge O_2(y,c) \quad (18)$$

$$Q_4(c,a,c) \quad \leftarrow \quad : \quad (19)$$

To deal with provisions and obligations in policy rules, we define the *Global Provision and Obligation Set (GPOS)* for an atom  $Q$ . Intuitively, a GPOS represents the alternative sets of POs that must be satisfied to derive  $Q$  in the policy rules. For example, consider the derivation of  $Q_1(a)$  using the rules in the example above. There are two ways to derive it, either by rule 15 and 16, or by 18 and 19. By *collecting* all the PO formulas, we get the GPOS for  $Q_1(a)$  to be  $(P_1(b) \wedge Q_1(s,a,b)) \vee (P_2(a,a) \wedge P_3(a) \wedge O_2(a,c))$ , representing the two possible

derivations of  $Q_1(a)$ . The GPOS of a given FAF specification can be computed in a way similar to the construction of the materialization structure.

Because each policy decision may involve different derivations and hence invoke different PO for the user and system, selecting a certain *minimum* set of POs in GPOS is an interesting issue. A number of possibilities should immediately become clear. For example, we may assume that there is a pre-known preference hierarchy on POs that we could represent by associating a weight to each provision and obligation predicate; a larger numerical weight means the predicate is more difficult to satisfy. In this case, we may simply choose the set of POs that is minimum in weight from the GPOS. Various issues are involved in such a selection scheme, but a basic algorithm is reported in Bettini (2002b).

As an example, consider an online store that allows its customer to purchase by registering and further allows the customer to upgrade her registration to a *preferred customer*. These two policies are stated in the first two rules. The next two rules state that the purchase price of an item is \$100 for a non-preferred customer and \$80 for a preferred customer. Thus, a customer has the choice of remaining in the non-preferred category and paying \$100 or registering as a preferred customer and paying \$80 per item. Further, suppose there is a one-time cost of \$10 to register as a non-preferred customer and to pay a \$20 fee for upgrading. Then, it is preferable to buy as a non-preferred customer for a one-time-only purchase, but to become a preferred customer for multiple purchases. This is so because the cost of the one-time purchase is \$80 after paying a one-time fee of \$30, as opposed to paying \$100 after paying a registration fee of \$10. The algorithm reported in Bettini (2002b) provides this computation.

cando(item,s,+buy)	← in(contract,Contracts)	: register(s,customer)}(20)
dercando(item,s,+buy)	← cando(item,s,+buy)	: upGrade(s,prefCust) (21)
do(item,s,+buy)	← dercando(item,s,+buy)	: payFees(s, \$80) (22)
do(item,s,+buy)	← cando(item,s,+buy)	: payFees(s, \$100) (22)

## Detailed Specification of Obligations

As we have seen above, a policy decision is taken when the user satisfies a sufficient set of provisions and accepts the required obligations. A non-trivial task involves monitoring accepted obligations and taking appropriate actions upon fulfillment and defaulting, respectively. For example, if the user agrees to pay a monthly fee for services as an obligation, the system should monitor this *obligation fulfillment* and, in case of failure, take necessary compensating actions. Such compensating actions could range from decreasing the *trustworthiness* of the user, replacing *unfulfilled* obligations with (perhaps more costly) alternatives, and/or taking punitive actions such as informing relevant authorities of the default or terminating the policy in force. To replace obligations with more stringent ones, the user needs to be informed of changes in contractual

obligations. Similarly, for obligations fulfilled as promised, it may be appropriate that a (positive) compensating action should be taken, such as acknowledging payment of monthly fees and thanking the user, and perhaps rewarding the user's good deeds by upgrading her trustworthiness. Before explaining how we can associate these actions with obligations, we introduce minor extensions to the syntax of obligation expressions.

As syntactic sugar, we allow [for  $x = 1$  to  $n$   $O(x)$ ] to be an obligation when  $O(x)$  is an obligation definition with a free integer variable  $x$  and  $n$  is an integer. In addition, we specify [If  $p$  then  $O$ ] to be a conditional obligation provided that  $p$  is a predicate formed by Boolean combinations of predicates and  $O$  is an obligation. The semantics of [If  $p$  then  $O$ ] is given by the evaluation of the condition  $p$ : if it evaluates to true, then obligation  $O$  must be fulfilled; otherwise, not. The truth of  $p$  is evaluated using classical truth tables.

The obligation specification given below forces the customer to pay back her loan in 36 installments provided that the loan is not cancelled by the *system* before a pay period. The payment can be done in two ways, either by paying on time (i.e., fulfill the obligation *payByDate*) or paying by an extended payment date (i.e., fulfilling the obligation *payByExtendedDate*).

```
[for n = 1 to 36
  [if(¬received(loanCancelled,customer,t,loan)^(t<30n))
    [payByDate(customer,30n+5,monthlyPayment)∨
     payByExtendedDate(customer,30n+15, monthlyPayment+100) ] ] ]
```

The value  $n$  refers to the  $n^{\text{th}}$  payment period, of which there are 36. We assume time is given in days, so that  $30n+15$  refers to that many days after the policy is enforced. Syntactically the obligation is constructed by using the *for* statement, but it is equivalent to the conjunction of 36 different conditional obligations, one for each of the 36 values the variable  $n$  can take. Each conditional construct is applied to the disjunction of the obligation predicates *payByDate* and *payByExtendedDate*.

To specify the actions associated with fulfillment and defaulting, we attach to an obligation expression a *fulfillment action specification* and a *defaulting action specification* respectively. The reason for attaching the action specification to a possibly complex obligation expression rather than to each atomic obligation is easily explained by an example. Suppose a policy decision was taken upon satisfaction of a VPOS containing a set of provisions and two obligations, the first requiring a payment by January 1, 2003, and the second a payment by February 1, 2003. Defaulting and, in particular, fulfilling actions will most likely be different for the single obligations and for the global one (the conjunction of them). For example, a reward may be given to the user if all the obligations were honored, while none is given for each single one. We propose the following syntax for fulfilling and defaulting clauses to obligations.

OBL ::= [ OBL Name  
 Definition: obligationExpression  
 FUL: ActionList  
 DEF:<obligationExpression, ActionList> ]  
 ActionList ::= [Action List: A<sub>1</sub>, ..., A<sub>n</sub>]

To specify consequences of accepting an obligation and consequently monitoring its fulfillment by the system, we introduce the notion of *action*. Actions are activities performed by the system to manage policy rules and monitor obligations. Common actions are those involving sending information to users or to other system components.

We represent actions by special predicates having any number of parameters. Sending actions are specified by the predicate *send* with at least three, but possibly more, parameters. The first parameter is the action name, the second parameter is the recipient's identity, and the third is the time at which the action is to be executed. Obligations and action terms may contain both variables and constants (of the appropriate type) as parameters. An example of a sending action is *send(loanCancelNotice, system, Jim Lee, 2003-Jan-14:07:30, loan451)*, specifying that the system should send a message *loanCancelNotice* at 7:30 on 2003-Jan-14 to the customer Jim Lee to inform him that his loan identified by *loan451* was cancelled. The *received* predicate becomes true as the effect of action *send*. Hence, in the example, the action *send(loanCancelNotice, system, Jim Lee, 2003-Jan-14:07:30, loan451)* will make true the predicate *received(loanCancelNotice, system, Jim Lee, 2003-Jan-14:07:30, loan451)*. This semantic interpretation implies that actions take effect immediately (i.e., action propagation takes no time).

The following specification refers to an obligation whose name is *payByDate*, having six parameters. The fulfilling component *FUL* includes two actions to be performed by the system: the first is to provide the customer with an acknowledgment, and the second is intended for the system itself to increase the reliability score of the customer.

OBL *payByDate*  
 Definition: *payByDate*(customer, loan, time, payment, penalty, upScore, downScore)  
 FUL: [Action List: { *send*(acknowledgeReceipt, customer, time, loan, payment), *adjustReliability*(system, time, customer, upScore) } ]  
 DEF: < [ OBL *payByExtendedDate*  
 Definition: *payByExtendedDate*(customer, time, payment+penalty),  
 [Action List: { *send*(reminder, customer, time, loan, payment+penalty),  
*adjustReliability*(system, time, customer, -downScore) } ]

Accepting the obligation *payByDate* entails the following consequences:

1. If the obligation is fulfilled, then an acknowledgment will be sent to the customer by the system and the reliability of the customer will be increased by the amount *upScore* by the system.
2. If the original obligation is not fulfilled, then the customer is obliged to fulfill a new obligation *payByExtendedDate*, and she will receive a *reminder* message from the system to do so, and the reliability of the customer will be decreased by the amount *downScore*.

In the example above, the defaulting clause DEF of the obligation *payByDate* uses an obligation *payByExtendedDate*, which requires its own definition. Consequently, the pair {*payByDate*, *payByExtendedDate*} forms an obligation chain of length 2, in the sense that the definition of *payByDate* uses *payByExtendedDate*. Our specification constraint requires that such obligation chains be of finite length.

Long-term success of a system with obligations increases with the user community honoring its obligations. For example, banks with lower loan default rates are less likely to get into financial difficulties. Similarly, user histories of obligation fulfillment need to be utilized in future dealings with them. This may be done by assigning a numerical measure of *trustworthiness*, referred to as reliability rating similar to the *credit rating* used by lending institutions in the United States. The ternary predicate *reliable*, where *reliable(subject, score, time)*, is true in a policy rule when *score* is the reliability rating of subject *subject* at time *time*. To update the reliability ratings of subjects, our system has a special action term *adjustReliability*, where *adjustReliability(time, subject, score)* adjusts the reliability score of the subject *subject* by  $\pm$ *score* at time *time*. Here,  $\pm$ *score* is a non-negative real number.

The following specification reports a policy rule using reliability rating. Indeed, it states that a customer can be automatically approved for a loan provided that her reliability rating is higher than 7.2 and that her income is at least three times that of the monthly payment arising out of the proposed loan.

```
access(customer,loan,approve) $\leftarrow$ reliable(customer,score,time),(score>7.2),
    monthlyIncome(customer,income),
    computePay(customer,loan,monthlyPayment),
    (income>3.monthlyPayment).
```

A system that depends on users to fulfill their obligations must monitor them. Monitoring involves considering all complex obligations and sending all required messages at times specified for each fulfilling and defaulting clause. In addition, monitoring must ensure that users properly adhere to conditional obligations. In

principle, the monitoring system should check at each instant if a defaulting or fulfilling of an obligation has occurred and take appropriate action. However, this is clearly a very inefficient solution; therefore, one of the objectives of monitoring is to derive an appropriate *schedule* of when to monitor the occurrence of such events. For example, if an obligation imposes a payment within a month from the signing of a contract, the system will derive the appropriate absolute deadline at the time of signing and will insert it as a guarding time in the schedule. A detailed obligation monitoring algorithm is given in Bettini (2002a).

## CONCLUSION

New and changing application requirements can be satisfied by having an authorization server that can simultaneously enforce multiple security policies. The Flexible Authorization Framework (FAF) of Jajodia et al. (2001b, 2001) provides such a capability. FAF specifies complex application-specific authorization requirements as a collection of rules in a stratified rule base. Therefore, every specification has a unique, well-founded model. Firing a backward chaining rule base in response to every authorization request may slow down any FAF-based authorization engine. To address this issue, Jajodia (2001b) proposed materializing the rules so that their conclusions could be looked up in a table.

To adapt to application dynamics, in this chapter FAF has been extended to incorporate rule changes and revoking once-granted permissions. We have shown how to minimally alter once-materialized conclusions of FAF rules so that the rule changes will be reflected in the materialization.

FAF has been further extended to include authorizations based on subjects satisfying provisions and obligations resulting from subjects being granted access permissions. Given a set of FAF rules decorated with provisions and obligations, we have given an algorithm that determines the optimal set of provisions and obligations to obtain an access.

## REFERENCES

- Apt, K. R., Blair, H., & Walker, A. (1988). Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann.
- Baral, C., & Subrahmanian, V. S. (1992). Stable and extension class theory for logic programs and default theories. *Journal of Automated Reasoning*, 8, 345-366.
- Bettini, C., Jajodia, S., Wang, S., & Wijesekera, D., (2002a, June). Obligation monitoring in policy management. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks* (pp. 2-12).

- Bettini, C., Jajodia, S., Wang, S., & Wijesekera, D., (2002b). Provisions and obligations in policy rule management and security applications. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China.
- Bruggemann, H. (1992). Rights in an object-oriented environment. In C. Landwehr & S. Jajodia (eds.), *Database Security V: Status and Prospects* (pp. 99-115). North Holland.
- Castano, S., Fugini, M., Samarati, P., & Martella, G. (1995). *Database Security*. Addison-Wesley, 1994.
- Dobson, J., & McDermid, J. (1989, May). A framework for expressing models of security policy. In *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 229-239).
- Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming* (pp. 1070-1080).
- Hagstrom, A., Jajodia, S., Wijesekera, D., & Parisi-Presicce, F. (2001, June). Revocations: A classification. In *Proceedings of the Fourteenth IEEE Computer Security Foundations Workshop*.
- Jajodia, S., Kudo, M., & Subrahmanian, V. S. (2001a). Provisional authorizations. In Anup Gosh (ed.), *E-Commerce Security and Privacy* (pp. 133-159). Boston: Kluwer Academic Press.
- Jajodia, S., Samarati, P., Sapino, M.L., & Subrahmanian, V. S. (2001b, June). Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26 (4), 1-57.
- Rabitti, F., Bertino, E., Kim, W., & Woelk, W. (1991). A model of authorization for next-generation database systems. *ACM Transactions on Database Systems*, 16 (1), 89-131.
- Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 285-309.
- van Gelder, A. (1989). The alternating fixpoint of logic programs with negation. In *Proceedings of the 8th ACM Symposium on Principles of Database Systems* (pp. 1-10).
- Wijesekera, D., Jajodia, S., Parisi-Presicce, F., & Hagstrom, A. (2001). Removing permissions in the authorization framework. *ACM Transactions on Database Systems*. *ACM Transactions on Database Systems*, 28(3), 209-229.

## ENDNOTES

- <sup>1</sup> This work was partially supported by the National Science Foundation under the grant number CCR-0013515.
- <sup>2</sup> Refer to Jajodia (2002b), Section 4.2 for the definitions of *hie*- and *rel*-predicates.

## Chapter V

# Enforcing Privacy on the Semantic Web

Abdelmounaam Rezgui, Virginia Tech, USA

Athman Bouguettaya, Virginia Tech, USA

Zaki Malik, Virginia Tech, USA

### ABSTRACT

*Over the past few years there has been a huge influx of web accessible information. Information access and storage methods have grown considerably. Previously unknown or hard-to-get information is now readily available to us. The World Wide Web has played an important role in this information revolution. Often, sensitive information is exchanged among users, Web services, and software agents. This exchange of information has highlighted the problem of privacy. A large number of strategies employed to preserve people's privacy require users to define their respective privacy requirements and make decisions about the disclosure of their information. Personal judgments are usually made based on the sensitivity of the information and the reputation of the party to which the information is to be disclosed. In the absence of a comprehensive privacy preserving mechanism, no guarantees about information disclosure can be made. The emerging Semantic Web is expected to make the challenge more acute in the*

*sense that it would provide a whole infrastructure for the automation of information processing on the Web. On the privacy front, this means that privacy invasion would net more quality and sensitive personal information. In this chapter, we describe a reputation-based approach to automate privacy enforcement in a Semantic Web environment. We propose a reputation management system that monitors Web services and collects, evaluates, updates, and disseminates information related to their reputation for the purpose of privacy protection.*

## INTRODUCTION

Web technologies are driving unprecedented paradigm shifts in various aspects of life. The technological impact of the Web has transformed the way we perceive things around us. Individuals and groups alike thrive on information in this era, dubbed as the “information-age.” The Web has become an immense information repository with perpetual expansion and ubiquity as two of its intrinsic characteristics. With the recent flurry in Web technologies, the “data-store” Web is steadily evolving to a more “vibrant” environment where *passive* data sources coexist with *active* services that access these data sources and inter-operate with limited or no intervention from humans. The active nature of the Web results from the intense volume of Web *transactions*. The Web has also brought a paradigm shift in the way information is accessed. Traditional systems are by nature closed and deterministic (e.g., enterprise networks) where data sources are accessible only by a few *known* users with a set of predefined privileges. On the contrary, the Web is an open and *non-deterministic* environment where information is potentially accessible by far greater numbers of *a priori unknown* users. Traditional methods of controlling the flow of information across systems are proving to be inadequate. The security community has extensively studied the problem of access control in the context of closed systems. However, the problem of access control in the open Web environment is quite different. In fact, solutions resulting from the research done on closed systems are only of peripheral importance to the problem of protecting information in the Web context. For example, access control models [e.g., RBAC, TBAC, MAC, DAC (Joshi, 2001)] that work for resources shared by a well-known, relatively small set of users are obviously of little help when the resources are information that may be accessed by millions of *random* Web clients. The control exhibited over the collection, accessibility, holding, and dissemination of Web-accessible information is minimal. This is mainly due to the extensive and extendable nature of the Web. Privacy issues come into play when the information is *private*, i.e., related to *personal* aspects, such as the health records, employment history, etc. Accessing personal information through the

Web clearly raises legitimate privacy concerns that call for effective, reliable, and scalable privacy preserving solutions.

The privacy problem is likely to become more challenging with the envisioned *Semantic Web* where machines become much better able to process and understand the data that they merely display at present (Berners-Lee, 2001). To enable this vision, “intelligent” software agents will carry out sophisticated tasks for their users. In that process, these agents will manipulate and exchange extensive amounts of personal information and replace human users in making decisions regarding their personal data. The challenge is then to develop agents that autonomously enforce the privacy of their respective users, i.e., autonomously determine, according to the current context, what information is private.

In this chapter, we propose a reputation-based solution to the problem of preserving privacy in the Semantic Web. The solution is based on two principles. First, the reputation of Web services is quantified such that high reputation is attributed to services that are not the source of any “leakage” of private information. Second, the traditional invocation scheme of Web services (discovery-selection-invocation) is extended into a reputation-based invocation scheme where the reputation of a service is also a parameter in the discovery-selection processes.

The chapter is organized as follows. An understanding about the notion of privacy in the context of the Web is presented in the next section. Then, we provide an overview of the need to preserve privacy in Web-enabled situations followed by an introduction to the key concepts used throughout the chapter. Then, we present a general model for reputation management in a Semantic Web environment. We then describe the architecture for deploying the proposed general model. Some variants of the proposed model are also listed. In the last part of the chapter, we list some research work that has addressed various aspects of trust and reputation in the Web, followed by the conclusion.

## WEB PRIVACY

The need for privacy is almost as old as the human race. However, unlike other *old* concepts, the *concept of privacy* has not yet reached a mature epistemological stage in which it can be perceived as a definite and universally accepted notion. In light of our previous discussion, we define Web privacy as a set of rules that are associated with any information on the Web that dictate implicit and explicit privileges for information manipulation. Information manipulation encompasses rules for access, storage, usage, disclosure, dissemination, changes, etc. These dimensions of privacy will be discussed shortly but first we should devise a methodology to characterize the information on the Web.

## Information Characterization

Information about an individual that is available/accessible through the Web can be characterized according to the *type*, *degree* or *nature* of information.

- **Type.** The information about an individual can either be *personal*, *behavioral* or *communicative*. Personal information refers to the data that helps in identifying the individual wholly or partly. A person's social security number or a passport number is enough to identify an individual while the name, marital status, phone numbers, financial information, etc., are only partly adequate. However, all of the listed (and more) are termed as personal information of an individual. Behavioral information includes activities that an individual performs while accessing the Web. These can be the durations of stay at a particular site, the frequency of visits to a particular site, buying patterns, etc. Communicative type of information refers to information in the form of electronic messages, postings to various sites, online polls, surveys, etc.
- **Degree.** The degree of information defines a level to which the information is related to an individual. For instance, some information may concern a single person (first degree), a family, or a larger group of people (e.g., residents of an area sharing a common zip code). Usually general information is less private while specific information tends to be more private. Therefore it is safe to conclude that the degree level of information and privacy are inversely proportional.
- **Nature.** The information about a user can either be *static* or *dynamic*. Information that is not expected to change substantially over time can be referred to as static information. These include names, residence/email addresses, personal affiliations and beliefs, etc. Dynamic information, on the other hand, is expected to change significantly over time (e.g., digital behavior).

## Dimensions of Web Privacy

There is no generally acceptable method to precisely determine what can or cannot be considered as a violation of privacy in the Web context. Ideally, a Web user must have a *reasonable* degree of control over access, collection and dissemination of information. We thus view privacy as a multi-dimensional concept that encompasses the following dimensions:

- **Access.** A privacy policy must define *access* privileges for individuals. These should include rules that state *who* can access *what*. For example, an online business can have a rule that only employees from the "procurement" department can create purchase orders while all other departments can have the authority to only view them.

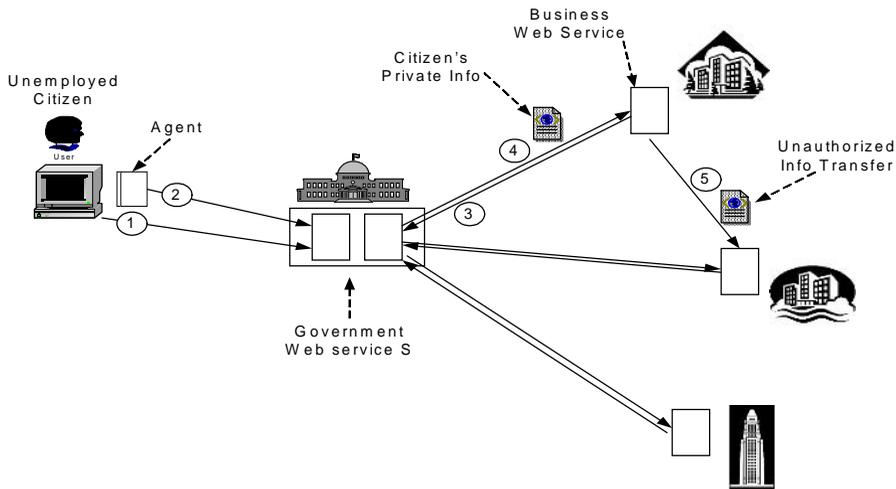
- **Collection.** The information *collection* requirement of privacy demands that no information concerning individuals is collected without their explicit consent. For example, healthcare sites have to explicitly mention the purpose of collecting the information and individuals can then decide whether to share information or not.
- **Usage.** The *usage* aspect of information defines the purpose for which the information is collected. If information is used for a purpose for which the use was not defined, then it is a violation of the usage dimension of privacy.
- **Storage.** The *storage* requirement determines whether and until when the information collected can be stored. For example, consider a citizen using a government Web-based service Medicaid that provides healthcare coverage for low-income citizens. Medicaid may state that the information it collects from citizens will remain stored in the underlying databases one year after they leave the welfare program.
- **Disclosure.** The information *disclosure* component defines the parties to which data can be disseminated. This dimension of privacy is the one that is considered to be the most critical in preserving privacy. If a Web site states that it will not disclose the collected information to unwanted sources, then it should abide by these terms. Disclosure of information without individual consent would be a violation of privacy.

## PRIVACY PRESERVATION REQUIREMENT

Preserving privacy is a key requirement for the Web to reach its full potential. Recent studies and surveys show that Web users' concerns about their privacy are key factors behind their reluctance in using the Web to conduct transactions that require them to provide sensitive personal information. The problem of computer privacy goes back to the early emergence of the burgeoning computer industry (Hoffman, 1969). Despite the recent growing interest in the problem of privacy, little progress appears to have been achieved. The social and cultural reasons include varied degrees for the perception of privacy across various geographical regions of the world. For instance, the concerns about privacy shown by individuals in North America or Europe are different from the way Asians perceive their privacy rights. From a technological standpoint, one reason for the slow progress is the erroneous view that assimilates privacy to security. Although not completely unrelated, the two problems of privacy and security are, essentially, different (Swire, 2002).

Much of the research in securing shared resources views security as an "inward" concept where different users have different access rights to different shared objects. Contrary to the concept of security, we see privacy as an "outward" concept in the sense that information exposes a set of rules (i.e., privacy policies) to users. These rules determine if and how any arbitrary user

Figure 1: An information exchange scenario in the Semantic Web



may access that information. Schoeman (1984) defined privacy along the same lines of control that an individual has over his or her information. In contrast to typical shared information, private information is, generally, related to an individual who normally *owns* the information and has the authority to specify its associated *privacy policy*. Despite important regulatory and technical efforts to address the issue of Web privacy, incidents of privacy violation on the Web continue to be in the headlines. Solutions to the problem of preserving privacy on the Web may not be conceivable or even feasible if not developed for and using tomorrow's Web building blocks: Web services. These are, essentially, applications that expose interfaces through which Web clients may automatically invoke them. A growing number of Web-based applications are being developed using Web services. Examples include health care systems, digital government, and B2B E-commerce. In their pre-Web versions, these applications were typically based on an information flow where users divulge their private information only to known, trusted parties. The recent introduction of Web services as key components in building these applications has enabled new types of transactions where users frequently disclose sensitive information to Web-based entities (e.g., government agencies, businesses) that are unknown and/or whose trustworthiness may not be easily determined.

Consider the example of a Digital Government (DG) infrastructure that offers a Web-based service *S* through which citizens access social and health plans (Figure 1). Part of the service's mission is to assist unemployed citizens in job placement and to deliver other related social and educational benefits. An unemployed citizen subscribes to the service *S* and delegates to his Semantic

Web agent the task of permanently interacting with the service *S* to look for potential job opportunities and any other government-sponsored benefits destined to jobless citizens. As part of this interaction, the citizen's agent must submit personal information to the service *S* (e.g., employment history, family and health information). For the job placement plan to be effective, the government may offer incentives to businesses that commit to hire 20% of their employees from citizens subscribed in the government plan. Businesses interested in these incentives deploy Web services that a government Semantic Web agent *Ag* periodically invokes to learn about any new job opportunities. This agent exploits these job opportunities on behalf of the citizens who qualify for the plan. In this process, it may be necessary for the agent *Ag* to transfer personal data to some of these external services. In some cases, these services are unknown to *Ag*, i.e., *Ag* does not have any history of prior interactions with these services. This obviously raises the problem of *trust* in an environment where Semantic Web agents and Web services interact with no or limited interaction history. The challenge is to, automatically, determine which information may be disclosed to which services. The previous example highlights the need for a solution that enables a privacy preserving interaction amongst Semantic Web agents and Web services. We believe that the linchpin of such a solution is a reliable *reputation* management mechanism that provides an *objective* evaluation of the *trust* that may be put in any given Web service. This mechanism must have functionalities to *collect*, *evaluate*, *update*, and *disseminate* information related to the reputation of Web services. Moreover, this mechanism must meet two requirements:

- **Verifiability.** Web services may expose *arbitrary* privacy policies to their users. Currently, users are left with no means to verify whether or not Web services actually abide by the terms of their privacy policies. Ideally, a privacy-preserving infrastructure must provide a practical means to check whether advertised privacy policies are actually enforced.
- **Automatic Enforcement.** In the envisioned Semantic Web, software agents will have to make *judgments* about whether or not to release their users' private data to other agents and Web services. Thus, a privacy preserving solution must not require an extensive involvement of users.

## REPUTATION MANAGEMENT CONCEPTS

In this section, we set the framework for our discussion. We first clearly define the terms of *Web services* and *Web agents*. We then introduce the two concepts of *attribute ontology* and *information flow difference* on which our reputation management model (described next) is built.

## Web Services and Web Agents Defined

In the envisioned Semantic Web, two types of entities interact: Web services and Web agents. The former are applications that may be invoked through the Web. For instance, in our DG example, a business may deploy a service that provides the list of current job opportunities. Services may have access to private information. Upon invocation, they may also *deliver* private information. In the dynamic Web environment, the number of Web agents and services may not be known *a priori*. In this chapter, we consider a dynamic set  $S$  of Web services that interact with a potential exchange of private information. Web agents are “intelligent” software modules that are responsible of some specific tasks (e.g., search for an appropriate job for a given unemployed citizen in the previous DG example).

## Attribute Ontology

An operation of a Web service may be viewed as a “processing” unit that consumes input parameters and generates output parameters. The invocation of a given operation may potentially result in privacy violation when one or more of the output parameters correspond to private attributes (e.g., Last Name, Address, Phone Number). A requirement for *automating* privacy preservation is to formally capture any possible leakage of sensitive information that may result from service invocation. Our approach is based on a concept called *Information Flow Difference* that provides an estimate of services’ potential to release private information. The definition of this concept is based on a *scaled attribute ontology* that captures two important characteristics of attributes, namely, *synonymy* and *privacy significance order*.

**Synonymy:** Consider two operations that both expect as their input a person’s home phone number and return the same person’s family name. The description of the first operation names the parameters: PhoneNumber and FamilyName while the description of the second operation names these parameters: PhoneNumber and LastName. Clearly, from a privacy perspective, these two operations are *equivalent*. This is due to the *semantic* equivalence of FamilyName and LastName. To capture this equivalence amongst attributes, the proposed attribute ontology defines sets of *synonymous* attributes. The following are examples of sets of synonymous attributes:

$T1 = \{ \text{FamilyName, LastName, Surname, Name} \}$

$T2 = \{ \text{PhoneNumber, HomePhoneNumber, ContactNumber, Telephone, Phone} \}$

$T3 = \{ \text{Address, HomeAddress, Location} \}$

**Privacy Significance Order:** Private attributes do not have the same sensitivity. For example, most people consider their social security number as being more sensitive than their phone number. To capture the difference in

attributes' sensitivity, we define the *Privacy Significance Level* as a function defined over the set of attributes and that, given an attribute  $a$ , associates a number  $PSL(a) \in \mathbb{N}$  that reflects attribute  $a$ 's significance from a privacy perspective. For any two given attributes  $a$  and  $b$ ,  $a$  is said to be of higher privacy significance if its privacy significance level,  $PSL(a)$ , is greater than  $b$ 's privacy significance level,  $PSL(b)$ . This establishes a *privacy significance order* between any pair of attributes. Of course, this order may not be universally valid. For example, two different persons may rank two attributes in different orders. Our approach assumes that, statistically, any two attributes are ranked consistently by a majority of individuals. However, the solution may readily be extended to employ *user-defined* attribute ontologies where *users* specify the sensitivity of the different attributes.

## Information Flow Difference

Let  $s_i$  be a Web service in the set  $S$  and  $Op_j$  an operation of the service  $s_i$  that has  $p$  input attributes and  $q$  output attributes. Let  $Input(Op_j)$  denote the set of input attributes for operation  $Op_j$ , and  $Output(Op_j)$  denote the set of output attributes for operation  $Op_j$ .

**Definition 1:** The *Information Flow Difference IFD* of operation  $Op_j$  is defined by:

$$IFD(Op_j) = \sum_{a \in Input(Op_j)} PSL(a) - \sum_{a \in Output(Op_j)} PSL(a)$$

**Example 1:** Assume that  $Op_j$  has as its input the attribute SSN and as its output the attribute PhoneNumber. The values of the function  $PSL$  for attributes SSN and PhoneNumber are respectively 6 and 4. In this example,  $IFD(Op_j) = 2$ . The meaning of this (positive) value is that an invocation of operation  $Op_j$  must provide information (SSN) that is *more* sensitive than the returned information (PhoneNumber). Intuitively, the *Information Flow Difference* captures the degree of "permeability" of a given operation, i.e., the difference (in the privacy significance level) between what it gets (i.e., input attributes) and what it discloses (i.e., output attributes).

In general, positive values for the function  $IFD$  do not necessarily indicate that invocations of the corresponding operation actually preserve privacy. In the previous example, a service invoking  $Op_j$  may still be unauthorized to access the phone number although it already knows more sensitive information (i.e., the social security number). However, invocations of operations with negative values of the function  $IFD$  necessarily disclose information that is *more* sensitive than their input attributes. They must, therefore, be considered as cases of privacy violation.

**Definition 2:** A service  $s \in S$  is said to have an information flow that violates privacy if and only if it has an operation  $Op$  such that:  $IFD(Op) < 0$ .

**Definition 3:** The *Information Flow Difference* of a Web service  $s$  is the sum of the *IFD*'s of all of its operations.

## REPUTATION MANAGEMENT MODEL

In the previous section, we introduced the concepts of *attribute ontology* and *information flow difference*. We also showed how these concepts are used to capture the sensitivity of the private information flowing to and from Web services. The incentive behind introducing these concepts is to develop mechanisms that *automatically* quantify services' reputation. In this section, we present a general model for a Semantic Web environment where interactions are based on reputation. The objective of the proposed model is to enable Web services and agents to interact in an environment where the decision to disclose private sensitive information becomes an *automatic, reputation-driven* process that does not require the intervention of human users. Our approach mimics the real life business and social environments where (good) *reputation* is a prerequisite (or, sometimes, the reason) for any transaction. The basic idea is to deploy a *reputation management system* that continuously monitors Web services, assesses their reputation and disseminates information about services' reputation to (other) services and agents. To each Web service, we associate a *reputation* that reflects a *common* perception of other Web services towards that service. In practice, different criteria may be important in determining the reputation of a Web service. For example, the reputation of a service that searches for "best" airline fares clearly depends on whether or not it actually delivers the best fares. In this chapter, services' reputation depends only on the effectiveness of their enforcement of the privacy of their users. To simplify our discussion, we will use "services" instead of "services and agents" in any context where "services" is an *active* entity (e.g., "service"  $s$  invokes operation  $Op$ ). We propose five criteria that are the basis in the process of reputation assessment.

### Computing Reputations

To compute the reputations of services in an automatic manner, we identified a set of criteria that: (i) reflect the "conduct" of services with regard to how they protect private information that they collect from users, and (ii) may be automatically and *objectively* assessed.

**Degree of Permeability:** We previously introduced the function IFD that determines services' *Degree of Permeability* (DoP), i.e., their proneness to the disclosure of sensitive information. We also use this function to rank Web services according to their DoP. For example, let  $s_1$  and  $s_2$  be two Web services. If  $IFD(s_1) < IFD(s_2) < 0$ , then  $s_2$  is said to be less permeable than service  $s_1$ .

**Authentication-Based Disclosure of Information:** Web services use different approaches to authenticate the senders of the received requests. The reputation of a service clearly depends on the strength of the mechanism used to authenticate clients. For example, the reputation-based infrastructure may adopt the rule that services using Kerberos-based authentication schemes have better reputation than services that use schemes based on user/password authentication.

**Across-User Information Disclosure:** In some situations, a Web service may properly authenticate its users but has the potential of across-user information disclosure. This characteristic corresponds to the situation where the service discloses private information about *any* valid user to *any* valid user. This flaw in the behavior of Web services must be considered in the process of evaluating services' reputation.

**Use of Encryption Mechanisms:** This criterion captures the efficiency of the encryption mechanisms used by Web services. For example, a service whose messages are encrypted using a 128-bit encryption scheme may be ranked better than another service whose messages are encrypted using a 64-bit encryption scheme.

**Seniority:** This criterion reflects the simple "fact" that, similarly to businesses in the real world, trust in Web services increases with the length of their "lifetime." If the *dates of deployment*,  $d_1$  and  $d_2$ , of two services  $s_1$  and  $s_2$ , are known, then the reputation of  $s_1$  may be considered better than that of  $s_2$  if  $d_1$  precedes  $d_2$ .

## Reputation Definition

We now present a formal definition of the reputation of Web services. Let  $R$  be the set of  $m$  criteria used in the process of reputation assessment ( $m = 5$  in the proposed list of criteria) and  $c_{ji}$  value of criterion  $c_j$  for service  $s_i$ . The values of these  $m$  criteria are normalized such that:

$$\forall s_i \in S, \forall c^j \in R, 0 \leq c^j_i \leq 1$$

In practice, the criteria used in reputation assessment are not equally important or relevant to privacy enforcement. For example, the *seniority* criterion is clearly less important than the *degree of permeability*. To each criterion  $c_j \in R$ , we associate a weight  $w_j$  that is proportional to its relative

importance as compared to the other criteria in  $R$ . A Web service's reputation may then be defined as follows:

**Definition 4:** For a given service  $s_i \in S$ , the reputation is defined by:

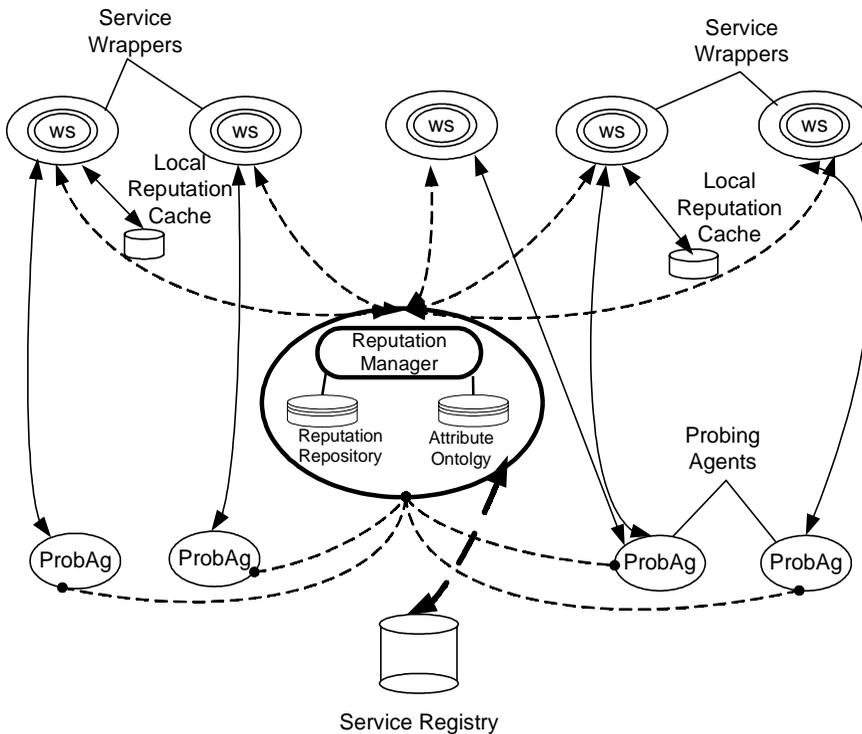
$$Reputation(s_i) = \sum_{k=1}^m w_k \cdot c_i^k \quad (1)$$

The intuitive meaning of formula (1) is that the reputation of service  $s_i$  is the weighted sum of its performances along each of the considered reputation criteria.

## THE ARCHITECTURE

We now describe the architecture (*Figure 2*) supporting the proposed reputation model. The proposed architecture has three main components: the *Reputation Manager*, the *Probing Agents*, and *Service Wrappers*.

*Figure 2: Reputation management system architecture*



## Reputation Manager

The *Reputation Manager* (RM) is the core of the reputation system. It is a *unanimously trusted* party responsible of (i) collecting, (ii) evaluating, (iii) updating, and (iv) disseminating reputation information. To understand the operation of the *Reputation Manager*, consider the following typical scenario enabled by this architecture. A Web service  $s_i$  (or an agent) is about to send a message  $M$  containing private information to a Web service  $s_j$ . Before sending  $M$  to  $s_j$ , the service  $s_i$  submits to the *Reputation Manager* a request asking for the reputation of service  $s_j$ . If a value of  $Reputation(s_j)$  is available and accurate (i.e., reasonably recent), the RM answers  $s_i$ 's request with a message containing that value. Based on the value of the received reputation and its local policy, the service  $s_i$  may or may not decide to send the message  $M$  to  $s_j$ . If the reputation of  $s_j$  is outdated or not available (i.e., has not been previously assessed), the *Reputation Manager* initiates a *reputation assessment process* that involves one of a set of *Probing Agents*. To assess the reputation of a Web service  $s_j$ , the RM collects information that is necessary to evaluate the given criteria for reputation assessment (discussed previously). The evaluation of most of these criteria is based on the (syntactic) description of the service  $s_j$ . For example, to evaluate the *degree of permeability* of  $s_j$ , the RM reads  $s_j$ 's description (by accessing the appropriate service registry), computes  $IFD(s_j)$  ( $s_j$ 's *Information Flow Difference*), and maps that value to the corresponding *degree of permeability*. The RM maintains an *attribute ontology* that is used in computing the degree of permeability of the different Web services. Once the DoP of service  $s_j$  is evaluated, it is stored in the local *Reputation Repository*. The other criteria may be obtained using similar processes. Once all the criteria are evaluated, the RM computes  $s_j$ 's reputation, stores the obtained value in the *Reputation Repository*, and sends it to the service  $s_i$ .

## Anonymous Probing Agents

Services' reputations are not static values. Different types of updates may affect a service's reputation. The *Reputation Manager* must permanently maintain an accurate perception of services' reputations. Two alternatives are possible for a *continuous monitoring* of Web services. In the first, the *Reputation Manager* permanently retrieves services' descriptions and issues requests to services to collect the information necessary to evaluate the different criteria of reputation assessment. This approach is clearly inadequate. First, it leads to a huge traffic at the RM. Second, *malicious* Web services may easily identify requests originating at the RM and reply with messages that do not reflect their actual behavior. To overcome these drawbacks, our solution deploys a set of  $\eta$  *probing agents* (or, *probers*) that collect information necessary to the process of reputation assessment and share it with the *Reputation Manager*.

These agents are responsible of permanently monitoring the services and reporting the collected information to the *Reputation Manager*. These agents are not co-located with the RM and are, *a priori*, anonymous to the Web services being monitored, i.e., services may not distinguish probing requests from ordinary requests.

Services with *low* reputation present a greater potential for unauthorized information disclosure. Therefore, the process of monitoring Web services must be distributed such that services with *low* reputation get probed more aggressively and more frequently. To meet this requirement, services are partitioned into  $\delta$  ( $\delta \in N$ ) clusters  $C_0, C_1, \dots, C_{\delta-1}$  such that services in the same cluster have “comparable” reputations.  $\delta$  is called the *clustering factor*. Formally,

$$\forall C_j, s_j \in C_j \Rightarrow \frac{j}{\delta} < Reputation(s_j) \leq \frac{j+1}{\delta} \quad (2)$$

To enable a variable probing policy, we associated  $\delta_i$  probing agents to each cluster  $C_i$ ,

$$\left( \delta \leq \eta = \sum_{k=1}^{\delta-1} \eta_k \right)$$

A reasonable distribution of the  $\eta$  probers on the  $\delta$  clusters is one in which:

$$\forall i, 0 \leq i < \delta, \dot{R}_i \cdot \eta_i = \alpha$$

where:  $\alpha$  is a constant and  $\dot{R}_i$  is the average reputation of services in cluster  $C_i$ , i.e.,

$$\dot{R}_i = \frac{\sum_{sk \in C_i} Reputation(sk)}{\delta_i}$$

where  $\delta_i$  is the size of cluster  $C_i$ . Probing agents associated with cluster  $C_i$  continuously and randomly invoke services in  $C_i$  to determine the values of the different criteria in the set  $\mathbf{R}$ . In the monitoring process, they may also have to access service registries and retrieve service descriptions.

An advantage of this cluster-based monitoring approach is that it is flexible and may be easily “tuned” to accommodate *loose* and *strict* privacy enforcement. For example, the parameter  $\alpha$  may be set higher (for all clusters) to

achieve stricter privacy control. Also, if a specific cluster or set of clusters (e.g., corresponding to businesses with low reputation) turn out to be more prone to information disclosure than others, only their probing agents may be instructed to switch to a more aggressive monitoring mode.

## Service Wrappers

A significant challenge in deploying the proposed approach is to, *a posteriori*, introduce a privacy preserving mechanism to existing Web services that are already built without a mechanism for privacy enforcement. The solution clearly requires modifying the service invocation scheme to accommodate the added mechanism. Moreover, the solution must not induce a high upgrading cost on “legacy” services. To achieve this transition to *privacy preserving services*, we introduce components called *service wrappers*.

A *service wrapper* associated with a service  $s_i$  is a software module that is co-located with  $s_i$  and that handles all messages received or sent by the service  $s_i$ . To send a privacy sensitive message  $M$  to a service  $s_j$ , the service  $s_i$  first submits the message to its wrapper. If necessary, the wrapper sends a request to the *Reputation Manager* to inquire about  $s_j$ 's reputation. Based on the answer received from the RM,  $s_i$ 's wrapper may then forward the message  $M$  to  $s_j$  or decide to cancel sending  $M$  to  $s_j$ . To avoid an excessive traffic at the *Reputation Manager*, the wrapper may locally maintain a *Reputation Cache* that contains information about the reputation of the most frequently invoked services.

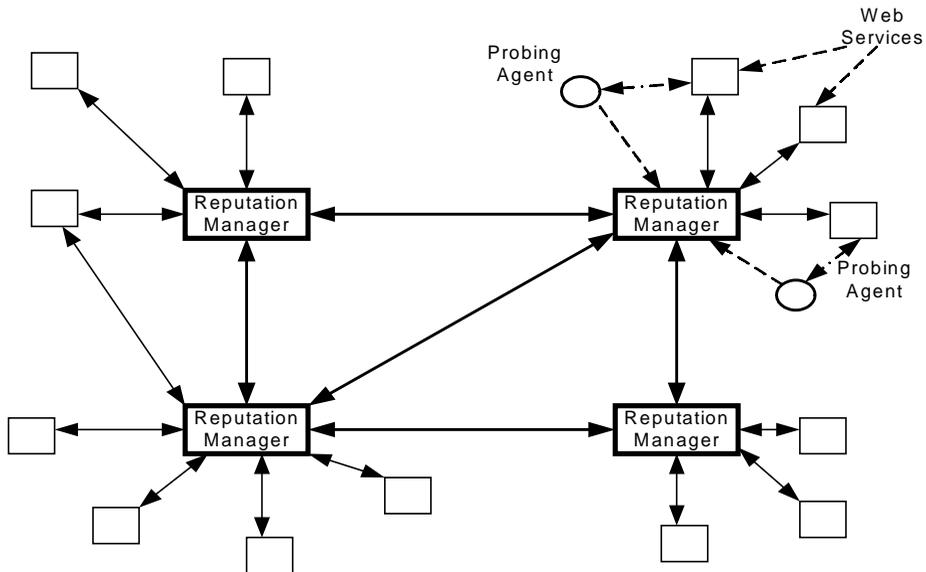
## VARIANTS OF THE GENERAL REPUTATION MODEL

The model that we have studied so far is based on the notion of centrality. A *centralized Reputation Manager* is responsible for collecting, evaluating, and disseminating reputations for various Web services. We now present the decentralized versions of the model that differ, essentially, in the way the *Reputation Manager* is deployed. We will briefly discuss distributed, registry-based and peer-to-peer reputation models as variants to the centralized approach and outline their main characteristics:

### Distributed Reputation Model

Evident from its name, a *distributed reputation model* is one in which multiple Reputation Managers cooperate with each other to manage the reputations of Web services. Each reputation manager works in approximately the same way that we have described previously. However, the scope and responsibility for reputation management is reduced. It is localized to a particular set of

Figure 3: Distributed reputation model

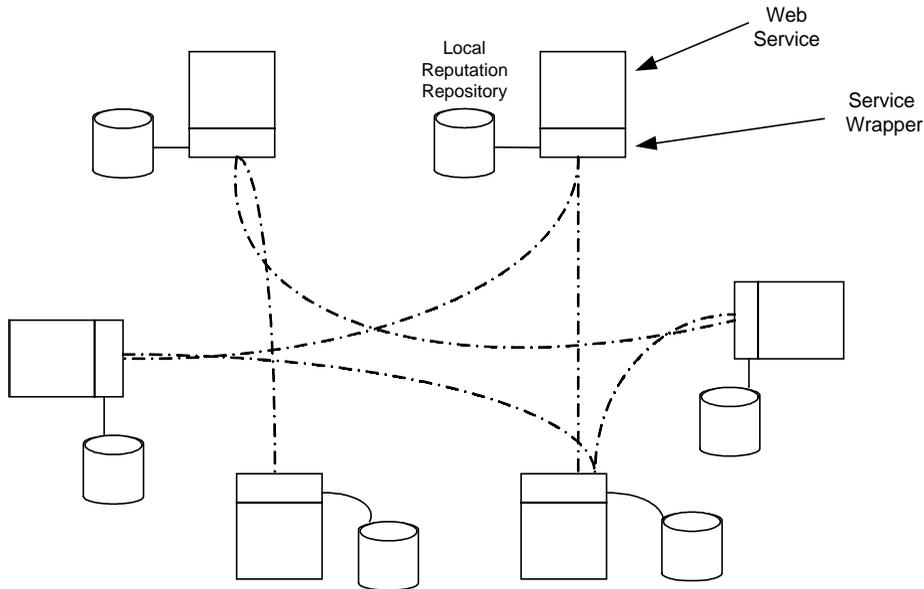


Web services and the reputation managers have to interact with each other periodically to ensure the accuracy of reputation reports. How a particular Web service gets “assigned” to a particular Reputation Manager is out of the scope of our current discussion. The model can be thought of as having various *local* reputation views, communicated periodically to achieve a *global* view of reputations. For example, a server that detects a change (falling below a minimum threshold) in a Web service’s reputation may decide to broadcast a notifying message to all other reputation managers, informing them of this event. Figure 3 shows a distributed reputation model, which is designed as a *decentralized* system.

## Registry-Based Reputation Model

The *registry-based reputation model* views a Web services’ reputation as part of its description. The idea is to base reputations on previous knowledge of the Web services. Service registries are extended with a reputation management layer that carries out the task of a *virtual reputation manager* (VRM). Reputations are reported by the participating Web services to the VRM after each transaction. Using the VRM service registries, Web services will be able to invoke services not only based on their description but, also, on their reputation. Ideally, the registry must be able to rank its services according to their reputation. Clients may then access registries to discover services that meet a minimum

Figure 4: Peer-to-peer reputation model



*reputation threshold*. An incentive-based reputation reporting mechanism also needs to be in place, which encourages *true* reputation reporting. Meaning, that false reports about a Web service's actions are not submitted to the VRM.

## Peer-to-Peer Reputation Model

This model (Figure 4) does not employ any entity that is exclusively dedicated to managing reputations. Each Web service has its own *Reputation Manager*. In this model, a Web service does not have a common "reputation" value, but rather, establishes its own perception of services' reputation, i.e., each service  $s_i$  has its own definition of the function *Reputation* (noted  $Reputation_i$ ) that is specified only for services *known* to  $s_i$ . A service  $s_i$  also adopts its own reputation threshold  $p_i$  and invokes a service  $s_j$  only if  $Reputation_i(s_j) \geq p_i$ .

It should also be noted that reputations about other *unknown* services can be gathered from *known trusted* Web services. This follows the general social mode of reputation reporting that individuals adopt in a real society. For instance, assume that  $X$  knows (can report reputation information of)  $Y$  and  $Z$ . Also  $Y$  and  $Z$  are *unknown* to each other. Now if  $Z$  wants to conduct some transaction with  $Y$ ,  $Z$  would ask  $X$  about  $Y$ 's reputation. This could happen over many links ( $X$  asking any other party), or  $Z$  may decide to ask several parties, and judge  $Y$  on some aggregate value.

## RELATED WORK

The concept of reputation has been extensively studied in E-commerce systems. Online businesses have deployed reputation systems that improve customers' trust and, consequently, stimulate sales (Resnick, 2000). Examples include: *Ebay*, *Bizrate*, *Amazon*, and *Epinions*. Several studies have investigated and, generally, confirmed the potential positive impact of reputation systems on online shoppers' decisions to engage in business transactions (e.g., the study reported in Lucking-Reily (2000) that targeted *Ebay*'s reputation system).

Current reputation systems are based on the simple idea of evaluating the reputation of a service or a product using feedbacks collected from customers that, in the past, have purchased similar or comparable services or products. Customers may then make "informed shopping" in light of past experiences. Different variations derived from this general model have been proposed. For example, in Zacharia (1999), the authors presented two models based on the *global* and *personalized* notions of reputation. The global model, *Sporas*, is similar to the reputation model used at *Ebay* while the personalized model, *Histos*, takes the identity of the inquirer and the environment in which the inquiry is carried out into account.

In most E-commerce systems, a typical transaction involves two parties (e.g., a seller and a buyer). In some cases, one of the parties (the seller) is also the provider of the reputation service (i.e., assesses and/or disseminates the reputation). These reputation systems inherently lack the *objectivity* that generally characterizes systems based on a *neutral* third party. An example of such a system was proposed in Atif (2002). It is based on a trust model that uses *intermediaries* to bolster the buyer's and seller's confidence in online transactions. An intermediary agent, known as a *trust service provider* (TSP), is responsible for making sure that the end-parties (buyer and seller) behave as expected (e.g., the buyer pays and the seller delivers). The whole process is invisible to the transacting parties as the TSP carries out the transaction in an automatic fashion.

Another approach to introduce more objectivity in reputation systems was proposed in Abdulrahman (2000). It treats trust as a non-transitive concept and takes into account the credibility of the source providing a recommendation. The proposed approach assumes that the recommender may lie or state contradictory recommendations.

Other solutions have also been proposed for other types of Web applications. In a previous work (Rezgui, 2002; Medjahed, 2003), we addressed the problem in the context of Digital Government applications. The solution was developed to enforce privacy in Web-accessible (government) databases. Its principle was to combine *data filters* and *mobile privacy preserving agents*.

The former were used to protect privacy at the server side (i.e., the databases) and the latter were used to preserve privacy at the client side (i.e., Web clients requesting private information).

The reputation system proposed in this chapter differs from the previously mentioned (and other) systems in three aspects. First, contrary to most existing systems that target *Web sites* or *Web databases*, our system targets a Semantic Web environment hosting Web services and software agents. Second, reputation in our system is not directly based on business criteria (e.g., quality of a service or a product) but, rather, it reflects the “quality of conduct” of Web services with regard to the preservation of the privacy of (personal) information that they exchange with other services and agents. Finally, the proposed reputation management system is fully automated. It does not use (or necessitate) potentially subjective human recommendations.

## CONCLUSION

In this chapter, we presented a reputation management system for the Semantic Web. The proposed system aims at automating the process of privacy enforcement in Web services. The solution is based on a general model for assessing reputation where a reputation manager permanently probes Web services to evaluate their reputation. We also presented alternatives to the proposed centralized model. The decentralized approach includes distributed, registry-based and peer-to-peer reputation models.

## ACKNOWLEDGMENT

This research is supported by the National Science Foundation under grant 9983249-EIA and by a grant from the Commonwealth Information Security Center (CISC). The authors would also like to thank Mourad Ouzzani and Brahim Medjahed for their valuable comments on earlier drafts of this chapter.

## REFERENCES

- Abdulrahman, A., & Hailes, S. (2000). Supporting Trust in Virtual Communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*.
- Atif, Y. (2002). Building trust in e-commerce. *IEEE Internet Computing*, 6 (1), 18-24.
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5), 34-43.

- Hoffman, L.J. (1969). Computers and privacy: A survey. *ACM Computing Surveys*, 1 (2), 85-103.
- Joshi, J., Ghafoor, A., Aref, A.G., & Spafford, E.H. (2001). Digital government security infrastructure design challenges. *Computer*, 34 (2), 66-72.
- Lucking-Reily, D., Bryan, D., Prasad, N., & Reeves, D. (2000). *Pennies From eBay: The Determinants of Price in Online Auctions*. Retrieved from the World Wide Web: <http://www.vanderbilt.edu/econ/reiley/papers/PenniesFromEBay.pdf>.
- Medjahed, B., Rezgui, A., Bouguettaya, A., & Ouzzani, M. (2003). Infrastructure for e-government web services. *IEEE Internet Computing*, 7 (1).
- Resnick, P., Zeckhauser, R., Friedman, E., & Kuwabara, K. (2000). Reputation systems. *Communications of the ACM*, 43 (12), 45-48.
- Rezgui, A., Ouzzani, M., Bouguettaya, A., & Medjahed, B. (2002). Preserving privacy in web services. In *Proceedings of the 4th ACM Workshop on Information and Data Management* (pp. 56-62).
- Schoeman, F.D. (1984). *Philosophical dimensions of privacy*. Cambridge: Cambridge University Press.
- Swire, P., & Steinfeld, L. (2002, June). Security and privacy after September 11: The health care example. *Minnesota Law Review*, 86 (6).
- Zacharia, G., Moukas, A., & Maes, P. (1999). Collaborative reputation mechanisms in electronic marketplaces. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*.

## **SECTION III:**

# **DATA DISTRIBUTION AND DISSEMINATION ON THE NET**

## Chapter VI

# Secure Data Dissemination

Elisa Bertino, Università degli Studi di Milano, Italy

Barbara Carminati, Università degli Studi di Milano, Italy

Elena Ferrari, Università degli Studi dell'Insubria, Italy

### ABSTRACT

*In this chapter, we present the main security issues related to the selective dissemination of information (SDI system). More precisely, after provided an overview of the work carried out in this field, we have focused on the security properties that a secure SDI system (SSDI system) must satisfy and on some of the strategies and mechanisms that can be used to ensure them. Indeed, since XML is the today emerging standard for data exchange over the Web, we have casted our attention on Secure and Selective XML data dissemination (SSXD). As a result, we have presented a SSXD system providing a comprehensive solution to XML documents. In the proposed chapter, we also consider innovative architecture for the data dissemination, by suggesting a SSXD system exploiting the third-party architecture, since this architecture is receiving growing attention as a new paradigm for data dissemination over the web. In a third-party architecture, there is a distinction between the Owner and the Publisher of information. The Owner is the producer of the information, whereas Publishers are responsible for managing (a portion of) the Owner information and for answering user queries. A relevant issue in this architecture is how the Owner can ensure a secure dissemination of its data, even if the data are managed by a third-party. Such scenario requires a redefinition of dissemination mechanisms*

*developed for the traditional SSXD system. Indeed, the traditional techniques cannot be exploited in a third party scenario. For instance, let us consider the traditional digital signature techniques, used to ensure data integrity and authenticity. In a third party scenario, that is, a scenario where a third party may prune some of the nodes of the original document based on user queries, the traditional digital signature is not applicable, since its correctness is based on the requirement that the signing and verification process are performed on exactly the same bits.*

## INTRODUCTION

Companies and organizations are today massively using Internet as the main information distribution means both at internal and external levels. Such a widespread use of the web has sped up the development of a new class of information-centred applications focused on the selective dissemination of information (hereafter called SDI). The obvious purpose of these applications is the delivery of data to a possible large user community. The term selective in this context means that each user should not receive all the data but he/she must receive only specific portions of them. Such portions can be determined according to several factors, such as user interests and needs, or the access control policies that the data source has in place. The chapter focuses on security issues for selective data dissemination services, since such issues represent one of the most novel and promising research directions in the field. A *Secure and Selective Dissemination of Information – SSDI* service– is an SDI service that ensures a set of security properties to the data it manages. In particular, we focus on four of the most important security properties: *authenticity*, *integrity*, *confidentiality*, and *completeness*. Since it is often the case that data managed by an SDI service are highly strategic and sensitive, the scope of SSDI applications is wide and heterogeneous. For instance, a first relevant scenario for such kinds of applications is related to the electronic commerce of information. This is, for instance, the case of digital libraries or electronic news (e.g., stock price, sport news, etc.). In such a case, users subscribe to a source and they can access information on the basis of the fee they have paid. Thus, in a digital library scenario, it is necessary to develop a mechanism ensuring that a user receives all and only those portions of the library he/she is entitled to access, according to the fee he/she has paid and only for the subscription period. Additionally, the service must ensure that these contents are not eavesdropped during their transmission from the library to the intended receiver. Another important scenario for SSDI applications is data dissemination within an organization or community, where the delivery is controlled by security rules defined by Security Administrator(s) (SAs). Consider, for instance, documents containing sensitive information about industrial projects. In such a case, personal data of the enrolled

staff should be available only to the authorized secretaries, whereas technical details should be accessible only to the technical staff.

Ensuring security properties is particularly crucial when a *push* dissemination mode is adopted for distributing documents to users. Indeed, under a push dissemination mode, the SSDI service periodically (or whenever some relevant event arises) broadcasts data to the whole (or to selected portions) of the user community. The traditional techniques for data access used in conventional DBMSs are not suitable to enforce information push, since they are based on the conventional on-user demand paradigm (referred also as *information pull*). In fact, since different users may have privileges to see different, selected portions of the same document, supporting the push dissemination mode with traditional techniques may entail generating different physical views of the same document and sending them to the proper users. The number of such views may become rather large and thus such an approach cannot be practically applied. Thus, in the chapter we illustrate some innovative solutions for efficiently supporting information push, based on the use of cryptographic techniques.

In explaining such techniques, and the architectures on which SSDI services are based, we cast our discussion in the framework of XML documents (Bray, 1998). The reason is that XML represents today a standard for data exchange over the Web. However, the approaches we present are general enough to be applied to web documents expressed according to other languages as well. More precisely, in the second part of the chapter we present few innovative solutions to efficiently implement secure and selective XML data dissemination services. We also discuss techniques that can be used to improve scalability for SSXD services, by focusing on the use of third-party architectures.

The remainder of the chapter is organized as follows. First we provide an overview of the works carried out in SDI field. Then we deal with the security issues related the SDI, and then we focus our attention to XML data, introducing SSXD services exploiting the access control paradigm, and the subscription paradigm. Finally, we propose a new architecture for SSXD service, which improves the scalability.

## **BACKGROUND: APPROACHES FOR SDI SERVICES**

Since the concept of SDI was introduced by H.P. Luhn (Luhn, 1958; Luhn, 1961), different approaches for implementing SDI services have been proposed. Moreover, due to the increasing improvements of hardware, communication bandwidth and ubiquity capability, the evolution and proliferation of SDI systems have been amazing in these last years, thus making the task of reviewing the related work difficult. Just to have an idea of the SDI evolution, at the beginning, that is, around the mid-1970s, SDI systems were implemented only for university

libraries [see (Housman, 1973) for an overview], with the aim to deliver updates of bibliographic information on technical journals to users. In these systems, managing only textual data, user interests were modeled by Boolean expressions, which state the keywords and the conditions that documents should have or satisfy in order to be interesting for the user. By contrast, today SDI systems are able to manage several data types: structured, unstructured, semi-structured (Belkin, 1987; Salton, 1983), and multimedia data (Alert, 2003). Furthermore, the use of the Web as means for data exchange has sped up the evolution of SDI systems. Typical and widespread examples of SDI systems exploiting the Web are the personal portal pages, that is, the possibility that nowadays the major portal sites (i.e., like Excite, Yahoo, etc.) allow users to customize their own portal pages by simply selecting topics (financial news, horoscope, society news, etc.) they are interested in to launch when their browsers open the portal site web pages.

Thus, instead of describing the main features of some commercial or academic SDI systems, we prefer to give in this section an overview of the most important issues related to SDI. We start by recalling that the common goal of all SDI systems is to collect new data items from several data sources, filter them according to some criteria (i.e., user profile, subscription, or access control policies) and then deliver them to interested users. This means that a key issue is that the SDI service is able to effectively and efficiently filter the data to deliver on the basis of user profile<sup>1</sup> — that is, to send all and only the information interesting for user, avoiding the delivery of unrequested data or the loss of information. Thus, in such a context, there exist at least two main issues to be faced according to which SDI systems can be classified. The first issue is the representation of dynamic user profile, whereas the second is the filtering of the information according to user profiles. Indeed, the two problems are closely related in that the user profiles representation greatly influences the filtering algorithm. In general, we can say that an SDI system converts the user profiles in queries expressed in a language understood by the system, so that the matching of them on the documents gives the information to deliver to the users. Thus, in the following we mainly focus on the first issue.

In particular, the research groups that have mostly investigated these issues are the Information Filtering (Loeb, 1972) and the Information Retrieval communities (Salton, 1983). Indeed, the main goal of these groups is to develop systems that select among large collections of information all and only the ones that are of interest for a requesting user. The results of the efforts carried out by these research communities are three different models, namely, the Boolean model, the Vector Space model, and the Probabilistic model, which represent the retrieval model most widely adopted by SDI systems.

The Boolean model represents user profile by means of set of words, and Boolean predicates applied to them. These predicates are matched on the documents to determine the information to deliver to users. Several SDI services

exploiting the Boolean model exist, such as, for instance, the IBM's Gryphon (Strom, 1998), and SIENA (Carzaniga, 2000). These SDI systems, basically, differ on the basis of the index structure adopted to implement efficient and effective information filtering [see (Yan, 1994a) for an overview of the index structures suitable for the Boolean model].

The major problem with the Boolean model is that it does not provide the possibility of ranking documents by their relevance for the submitted query. Indeed, according to this model all the documents satisfying the boolean expression specified in the query are delivered to the users, without any other kind of refining. Best-match retrieval models have been devised to cope with this limitation. In particular, the Vector Space model (Salton, 1983; Salton, 1989; Yan, 1999), which is widely known, represents the texts and the queries (i.e., the user profile) as weighted vectors in a multidimensional space. Then, the information filtering is based on a comparison between the text vector and the query vector. The closer the two vectors the more relevant will be the text query. The assumption, which is the base of the Vector Space model, is that the more similar the vector representing a text is to a vector representing the user description, the more likely is that the text is relevant to that user. Similarly to the Boolean model, the SDI systems exploiting a Vector Space model may differ on the adopted index structure [for an overview of the index structures for SDI adopting a Vector Space model see (Yan, 1994b)].

Finally, the probabilistic model is based on the Probability Ranking Principle (Robertson, 1977), which later became known as the binary independence retrieval (BIR). The fundamental idea of the probabilistic model is based on the concept of idea answer set. This set represents the set of documents consisting of all and only those documents that are relevant for a predefined user query. Thus, given a user query if the description of the ideal answer set corresponding to is known, we would not have problems in retrieving its document. According to this basic concept, the main issue in the probabilistic model is how to describe the ideal answer set, which implies associating a set of properties to it. In order to devise these properties, the probabilistic model implies a first guessing phase, where a preliminary probabilistic description of the ideal answer set is provided. According to this description is retrieved an initial set of documents. In order to improve the probabilistic description of the ideal answer set, the probabilistic model implies iterations with the user, with the goal of refining the initial set of retrieved documents. Thus, the probabilistic model implies the reiterating of the iteration phase, in order to make the description much closer as possible to the ideal answer set.

It is important to note the nature of the data being filtered strongly affects the filtering model. Thus, since the IR and IF communities are mainly related to textual data, the database community has investigated the data filtering algorithms for structured data. More precisely, the most significant research efforts have been done in the area of Continual Queries. A continual query is a standing

query that monitors updates of the source, thus to return the interested information to users as soon as it is acquired by the source. Early work on CQ for relational databases was done by Terry et al. (Terry, 1992). More recently, OpenCQ (Liu, 1999) and NiagaraCQ (Chen, 2000) have been proposed for information delivery over the Internet.

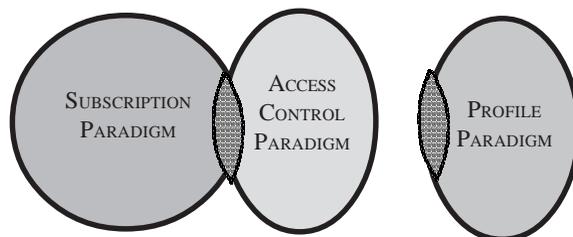
Moreover, due the rapid diffusion of XML as a standard for data exchange over Internet, in the last couple of years several researchers have investigated efficient filtering strategies for XML documents on the basis of user preferences (Altinel, 2000; Diao, 2003; Pereira, 2001). In the proposed approaches, user preferences are specified using some XML pattern specification language (e.g., XPath, 1999). The goals of these research activities are to define algorithms and data structures to implement an effective, efficient and scalable filtering of XML data.

## **SECURITY ISSUES IN DATA DISSEMINATION SERVICES**

In the previous section we have summarized the main research efforts carried out in the area of information dissemination, and we have given an overview of some of the proposed approaches. From this overview it easy to note that security issues have not been so far widely investigated. However, today companies and organizations are massively adopting the information dissemination paradigm for theirs businesses, and this makes security issues very relevant and highly strategic. The aim of this section is thus to introduce some security properties that a secure and selective dissemination of information system must ensure, by sketching some of the possible solutions, which will be carefully presented in the rest of this chapter. However, since these security properties and the mechanisms to ensure them are strictly correlated to the data dissemination paradigms adopted by the SSDI system, we need first to introduce them. Indeed, in designing an SSDI system, it is necessary to take into consideration that there does not exist a unique paradigm to filter the data before its delivery. Indeed, it is possible to devise at least three different paradigms, namely the access control paradigm, the profile-based paradigm, and the subscription-based paradigm, which are briefly introduced in the following.

The access control paradigm implies the existence of an access control model whereby a SA specifies authorization rules stating which portions of the source can be accessible by which users. This is the case, for instance, of a source containing confidential data (e.g., high strategic industrial information), which must be accessible in a selective manner. In this case, the selective delivery of data is regulated by the specified access control policies. This means that each user receives only the data portions for which he/she has an authorization according to the specified policies.

*Figure 1: A taxonomy of data filtering paradigms*



A different approach for data filtering exploits the profile-based paradigm. According to such a paradigm, the service releases data to users on the basis of their profiles (i.e., user interests, and needs). More precisely, a user profile contains information that could be directly specified by the users (i.e., during an subscription phase), or could be collected indirectly by the SSDI service (for instance, by analysing previous data requests made by the same user), and which are used by the SSDI system to better customize the service for their users. Unlike the access control paradigm that is mainly driven by security issues, in a profile-based scenario the main goal of data filtering is to avoid the delivering of superfluous information to users.

The last data filtering paradigm is the subscription-based paradigm, according to which users have to register to the data source by a mandatory subscription phase. More precisely, during this subscription, a user specifies directly to the SSDI service the portions of the source that he/she is interested in receiving. In this case, the selective delivery of data is regulated by this selection, in that users will receive all and only the subscribed portions. It is interesting to note that in this scenario the user profiles are not taken into account, in that the data filtering mechanism does not care about the user interests and needs, but it simply sends to a user the subscribed portions. Due this reason, this paradigm is particularly tailored for pay-per-view services, which are characterized by the fact that SDI services release to users all and only those portions for which the users have paid a subscription fee.

As reported in *Figure 1*, the three paradigms are not exclusive. This means that there may exist SDI services exploiting, for instance, both the subscription and the access control paradigm. Starting from the left, the first circle represents SSDI systems exploiting the subscription paradigm, the second circle represents those systems exploiting the access control paradigm, whereas the last circle represents the SSDI systems adopting the profile paradigm. As depicted in *Figure 1*, an SSDI service could exploit both the access control and the subscription paradigms, as well as both the access control and the profile paradigms. The decision to adopt the combination of two paradigms depends on the scenario in which the service operates. Consider for instance a newspaper

company exploiting an SSDI system to securely disseminate articles and news. In this scenario, it is possible to adopt both the subscription and the profile-based paradigm. However, the newspaper company may want to ensure additional filtering criteria, for instance to avoid sending articles containing offensive content to underage people. In such a case, the SSDI system should adopt also the access control paradigm in order to satisfy these additional secure requirements.

We are now ready to start our discussion by focusing on three of the most important security properties: authenticity, integrity, and confidentiality.

Ensuring document authenticity means that the user receiving a document is assured that the document contents come from the source they claim to be from. Ensuring document integrity means ensuring that the contents of the documents are not altered during their transmission from the source to the intended recipient. Finally, ensuring document confidentiality means that the document contents can only be disclosed to users, authorized according to the specified access control rules stated on the source.

In order to ensure these properties it is necessary to integrate the SDI architecture with an additional mechanism whose goal is to securely disseminate the information. The mechanism must take into consideration the delivery mode adopted by the service. There are two main data delivery modes that an SDI service can adopt: pull mode and push mode. According to the information pull mode, when a user submits an access request, the SSDI mechanism checks which authorizations the user has on the requested document. Based on the information contained in the profile, the user may be returned a *view* of the requested document that contains all and only those portions that match the profile. Confidentiality requirements are thus guaranteed since the profile also contains information on the access control policies satisfied by the user. To ensure also the answer integrity and the authenticity, the SSDI mechanism can adopt standard cryptographic techniques (i.e., digital signature, message authentication code, symmetric or asymmetric encryption, etc.). Besides the traditional information pull mode, a push mode also can be adopted. According to the push mode the SSDI system periodically, or when some pre-defined events happen (i.e., an update on the source), sends (portions of) its documents to interested users, without the need of an explicit access request by the users. Supporting push information in SSDI system is much more complicated, because standard techniques used for the pull mode are not efficient when applied to the push mode. In fact, since different users may be returned different, selected portions of the same document, supporting a push dissemination approach may entail generating different physical views of the same document and sending them to the proper users. The number of such views may become rather large and, thus, such an approach cannot be practically applied.

To avoid this situation a solution is that of exploiting broadcast encryption techniques (Fiat, 1994) to efficiently support information push. Broadcast encryption implies the encryption of different portions of the same document

with different encryption keys based on the specified access rules. Thus, the same encrypted copy of the document is then broadcasted to all users, whereas each user only receives the key(s) of the portion(s) he/she is enabled to access.

It is important to note that the definition of the mechanisms ensuring the above introduced security requirements depends also by the architecture adopted by the SSDI system. Indeed, the mechanisms become more cumbersome if a third-party architecture is adopted. In fact, the basic idea of third-party architectures is the distinction between the Owner of the information, and a third-party entity managing this information. Such a scenario requires a redefinition of dissemination mechanisms developed for the traditional SDI system. Indeed, the traditional techniques cannot be exploited in a third-party scenario. For instance, let us consider the traditional digital signature techniques, used to ensure data integrity and authenticity. In a third-party scenario, that is, a scenario where a third-party may prune some of the nodes of the original document based on user queries, the traditional digital signature is not applicable, since its correctness is based on the requirement that the signing and verification process are performed on exactly the same bits.

Moreover, in a third-party scenario, in addition to the traditional security requirements (i.e., integrity, authenticity, and confidentiality), a further important requirement is the completeness of the answer. Ensuring completeness implies that a user receiving an answer to an access request must be able to verify that he/she receives all the document(s) (or portion(s) of document(s)) that he/she is entitled to access, according to the stated access control policies and the submitted request. To cope with these security requirements, later in the chapter we propose an approach ensuring the completeness and the authenticity of the answer in a third-party architectures.

## **A COMPREHENSIVE FRAMEWORK FOR AN SSXD SYSTEM**

In this section, we present the overall architecture of the proposed SSXD, whereas the descriptions of its internal architecture are presented in later sections. We cast our discussion in the framework of XML documents. For this, reason, before presenting the details of our architecture, we need to briefly introduce the basic notions of XML.

### **Basic Concepts of XML**

The eXtensible Markup Language (XML) (Bray, 1998) is currently the most relevant standardization effort in the area of document representation through markup languages and it is rapidly becoming a standard for data representation and exchange over the Web. The motivation pushing the large development

Figure 2: An example of XML document

```

<?xml version="1.0" encoding="UTF-8" ?>
<Newspaper Title="MyNewspaper" Date="...">
  <Frontpage>
    <Leading_Article Author="..."Title="...">
      .....
    </Leading_Article>
    <Paragraphs>
      <Paragraph Author="..." Title="...">
        .....
      </Paragraph>
      .....
    </Paragraphs>
  </Frontpage>
  <Politic_page>
    <Politic topic="USA" Author="..." Title="...">
      .....
    </Politic>
    <Politic topic="EUROPE" Author="..." Title="...">
      .....
    </Politic>
    .....
  </Politic_page>
  <Literary_page>
    <Article topic="Books" Author="..." Title="...">
      .....
    </Article>
    <Article topic="Movies" Author="..." Title="...">
      .....
    </Article>
    .....
  </Literary_page >
  <Sport_page>
    <News topic="Soccer" Author="..." Title="...">
      .....
    </News>
    <News topic="Basket" Author="..." Title="...">
      .....
    </News>
    .....
  </Sport_page>
</Newspaper>

```

efforts concerning XML is the need to introduce also for Web documents the usual separation between the *structure* and *contents* of documents and their *presentation*, and to describe the semantics content of the various document portions. By separating document structure and contents from presentation, the same source document can be visualized according to different modes. Therefore, a document coded according to XML can be displayed at various devices, even at devices not foreseen at document preparation time. Therefore, even though XML has been initially developed for the Web, it can be used in any application or environment where one needs to organize and describe data, independently from the storage formats and transmission means.

The basic building blocks of an XML document are tagged elements. Elements can be nested at any depth and can contain other elements (*subelements*) in turn originating a hierarchical structure. An element contains a portion of the document delimited by two *tags*: the *start tag*, at the beginning of the element, with the form <tag-name>, and the *end tag*, at the end of the element, with the form </tag-name>, where *tag-name* indicates the type of the element (*markup*). Additionally, an element can contain attributes of different types allowing one to specify element identifiers, additional information about the element, or links to other elements of the document (attribute of type IDREF(s)/URI(s)). An important characteristic of XML is the possibility of attaching to a document an intentional description of its structure, i.e., the *Document Type Definition* (DTD) or an *XMLschema*.

An example of XML document, modeling a newspaper, is given in *Figure 2*. In particular, the newspaper consists of a Frontpage (modeled through the Frontpage element) containing a leading article (the Leading\_Article element) and one or more additional short articles (the Paragraph elements). As traditional paper-based newspaper, the newspaper contains also additional pages related to several topics. Each page is modeled by a different XML element (i.e., Literary\_page, Sport\_page and Politic\_page elements).

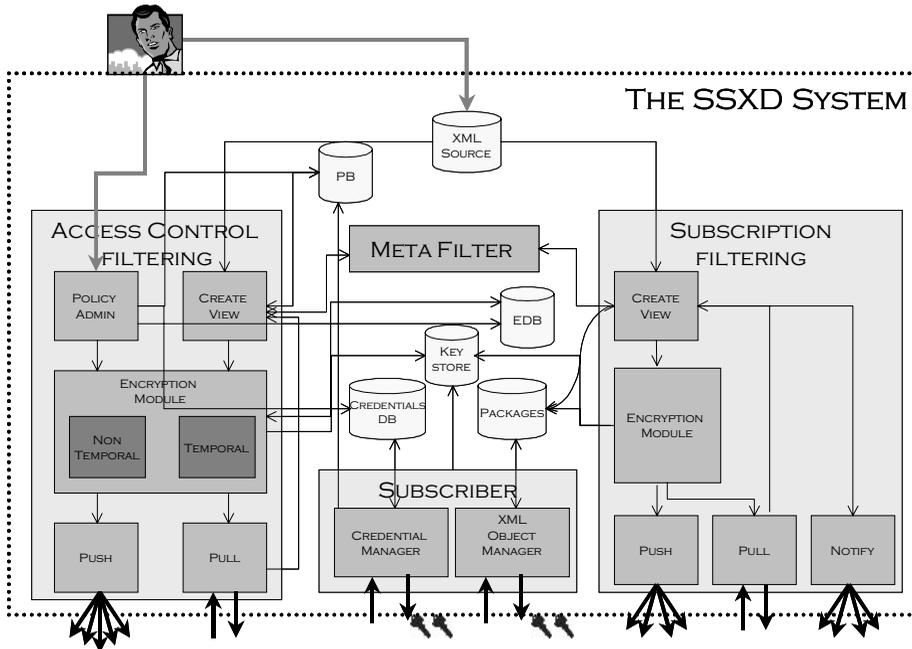
## THE CORE SSXD ARCHITECTURE

According to the taxonomy previously introduced, we have designed an SSXD system supporting both the access control and the subscription paradigm. Our SSXD system, whose overall architecture is represented in *Figure 3*, is built on top of two systems previously developed by us, that is, the AuthorX (Bertino, 2001b) system, for a selective dissemination of XML documents according to a discretionary access control model, and the XPublisher (Bertino, 2001a) system, implementing a SSXD adopting a subscription-paradigm.

As depicted in *Figure 3*, the architecture includes two separate modules (i.e., the *Access Control Filtering* module, and the *Subscription Filtering module*, respectively) to implement these two data dissemination paradigms. Moreover, these modules have their own encryption scheme for push distribution and their own strategy for the view generation. However, it is possible to integrate the access control and subscription paradigms by means of the “Meta Filter” module. In particular, this module starts up each time a modification occurs on predefined data, that is, data on which both the access control and the subscription paradigms are defined.

Moreover, in addition to the modules implementing the access control and subscription dissemination mechanisms, the proposed SSXD architecture includes also a module managing the subscription phase, i.e., the *Subscriber* module. The aim of this module is to manage the registration of new users into

Figure 3. The SSXD core architecture



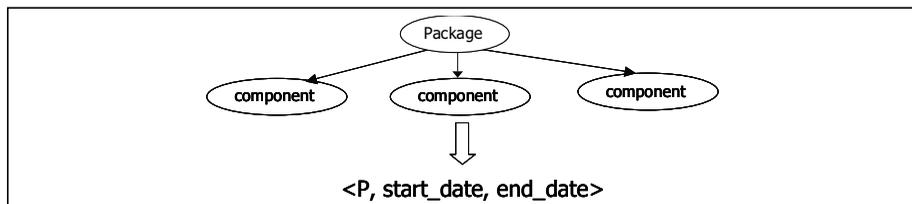
the SSXD system. Since our SSXD system combines both a subscription and an access control mechanism, the Subscriber must be able to manage the subscription phases of both of them. More precisely, as it will be explained in next sections, since the access control model adopted in our framework exploits the notion of credentials to identify the users, the Subscriber module needs to be able to manage credentials. By contrast, the user subscriptions are modelled as in the continual query paradigm, that is, they are defined as a set of queries (i.e., XPath expressions) (XPath, 1999) on the document source, which specify the information the user is interested in receiving. To cope with these requirements, the Subscriber module consists of two components: the *Credential Manager* and the *XML Object Manager*.

Since the Access Control Filtering and the Subscription Filtering modules are the core components of our SSXD system, in the following sections we will describe them in detail.

## Subscription Filtering Module

The Subscription Filtering Module has been mainly conceived for a digital library-like scenario, where the most important requirement to be satisfied is the complete flexibility in the user subscription. According to this approach, our SSXDI system allows users to customize their subscriptions in terms of data they are interested in receiving, in terms of the subscription period during which the

Figure 4: Package structure



data must be sent, and in terms of the distribution mode under which the data are delivered. User subscriptions are defined according to the continual query paradigm, that is, they are defined as a set of queries (i.e., XPath expressions) (XPath, 1999) on the document source, which specify the information the user is interested in receiving. In such a context, the SSXD service has to ensure that information is accessible only during the subscription periods. In order to satisfy this requirement the user subscription contains also information on the subscription periods. More precisely, our approach is based on the concept of *package*. As depicted in Figure 4, a package is defined as a collection of *components*, where each component consists of one or more XPath expressions (denoted as P in Figure 4) specifying portions of information, together with a subscription period (i.e., the begin and the end date of the subscription period).

A further feature is that it supports several dissemination modes, by allowing the user to select, for each package, the distribution mode to apply to that package according to his/her needs. More precisely, a user can select, during the subscription phase, if the content of the package (i.e., the content of the portions specified in the package), must be delivered according to a *pull* or a *push* distribution. The push mode is further specialized into *push<sub>on</sub>* and *push<sub>off</sub>*. If the *push<sub>on</sub>* mode is selected, the SSXD system, upon any modification of the source content, sends the modified portions to the user subscribed to a package to which the modified portions belong. By contrast, if the *push<sub>off</sub>* mode is selected, the system periodically (for instance once a day) checks which portions of the source have been modified and sends the updated portions to all the users subscribed to a package to which the portions belong. We introduce these two kinds of push-based distribution modes, making the SSXD system more adaptable to different user characteristics. Moreover, in order to further improve the flexibility service, a further distribution mode has been introduced in addition to the pull and push distribution modes, that is, the *notify* mode. If the notify mode is selected, the user only receives a notification (for instance an e-mail or an SMS) informing him/her that a modification in one of his/her package occurs, and he/she has to explicitly request the portion(s) he/she is interested in receiving.

Thus, the Subscription Filtering module gives the user complete freedom in defining the information he/she is interested in, in that a user can subscribe to

different packages and each package may contain portions with different subscription periods, according to the user needs.

*Example 1*

Consider four users,  $U_1$ ,  $U_2$ ,  $U_3$ , and  $U_4$ , and the XML document in *Figure 2*. Suppose that user  $U_1$  is interested in receiving the MyNewspaper Frontpage in the period from 1/01/03 to 12/31/03, and all the newspaper articles whose title contains the phrase “XML security” in the period from 01/01/03 to 06/30/03. Moreover, suppose that  $U_1$  specifies, during the subscription phase, the  $push_{on}$  mode for all these portions. Suppose that user  $U_2$  is interested in receiving the Frontpage of all the newspapers in the period from 03/01/03 to 12/31/03. Moreover, he wants to receive in the period from 03/01/03 to 12/31/03 all the articles written by Tom Red. Furthermore,  $U_2$  specifies the Pull mode for all those portions. Suppose that user  $U_3$  is interested in receiving the MyNewspaper Frontpage in the period from 01/01/03 to 06/30/03, and that for this portion he specifies the  $push_{on}$  mode. Finally, suppose that user  $U_4$  is interested in receiving the MyNewspaper Frontpage in the period from 01/01/03 to 02/28/03, and that for this portion he specifies the  $Push_{on}$  mode. Thus, the XPath expressions generated during the subscription phase are the following:

- $P_1$ : the Frontpage element of the instances of the newspaper DTD;
- $P_2$ : the Frontpage element of the XML document in *Figure 2*;
- $P_3$ : the Article elements of all the instances of the newspaper DTD, where the attribute Title contains the phrase “XML security”;
- $P_4$ : the Article element of all the instances of the newspaper DTD, where the attribute Author has value “Tom Red”.

As a consequence of the different subscription periods selected by the users and of the different subscription modes, the packages the publishing service creates are the following:

$$\begin{aligned}
 PA_1 &= \langle U_1, \langle C_2, C_3 \rangle, Push_{on} \rangle \\
 PA_2 &= \langle U_2, \langle C_1, C_4 \rangle, Pull \rangle \\
 PA_3 &= \langle U_3, \langle C_5 \rangle, Push_{on} \rangle \\
 PA_4 &= \langle U_4, \langle C_6 \rangle, Push_{on} \rangle
 \end{aligned}$$

where the components are:

$$\begin{aligned}
 C_1 &= \langle \langle \text{“//Newspaper/Frontpage//node()”} \rangle, 03/01/01, 12/31/01 \rangle \\
 C_2 &= \langle \langle \text{“//Newspaper[@Title=“Times”]/Frontpage/node()”} \rangle, 01/01/01, 12/31/01 \rangle
 \end{aligned}$$

$C_3 = \langle \langle \text{"//Newspaper//Article[@Title="XML security"]/node()"} \rangle, 01/01/01, 06/30/01 \rangle$

$C_4 = \langle \langle \text{"//Newspaper//Article[@Author="Tom Red"]/node()"} \rangle, 03/01/01, 06/30/01 \rangle$

$C_5 = \langle \langle \text{"//Newspaper[@Title="Times"]/Frontpage/node()"} \rangle, 01/01/01, 06/30/01 \rangle$

$C_6 = \langle \langle \text{"//Newspaper[@Title="Times"]/Frontpage/node()"} \rangle, 01/01/01, 02/28/01 \rangle.$

Security properties are enforced by adopting suitable encryption schemes. In particular, in this scenario, it is necessary to ensure that a user can access the portions of information to which he/she has subscribed for the duration of his/her subscription and that such information is no longer accessible by the user when the subscription expires. To fulfill these requirements, in Bertino (2001a) we have proposed different encryption schemes for the different distribution modes we support. More precisely, we exploit two different symmetric encryption schemes, namely the *Pull Package* encryption scheme, and the *Push Package* encryption scheme, which are briefly described in the following.

#### *Pull Package Encryption Scheme*

If a user subscribes to a package in the notify mode, each time a portion of a non-expired component of such package is modified the user receives a notification that advises him/her of the modification. Then, the user can decide if he/she is interested in receiving the modified portion(s) or not. If the user is interested in receiving the modified portion(s), the user has to explicitly request them to the SSXD system as in the case of pull mode packages. Thus, we can uniformly treat the distribution of packages with notify and pull mode, by using the Pull package encryption scheme. More precisely, in case of pull and notify packages the subscriber receives, upon the completion of the subscription phase, a symmetric key (Stallings, 2000), which is then used by the SSXD system to encrypt the portions of the package returned to the user as answer to an access request. In such a way, only the entitled user is able to decrypt the access request answer because he/she is the only one sharing that encryption key with the SSXD system. Indeed, the basic idea of this scheme is to associate a different symmetric key, called *package key*, with each different package defined with the pull or notify distribution mode. Package keys are returned by the SSXD system to a user as a result of the subscription phase. More precisely, during the subscription phase a user defines her/his packages on the basis of her/his needs. Then, as last step of the subscription, he/she specifies the distribution mode. If the user selects the pull or notify distribution mode, the SSXD system checks if the chosen package is an existing one (i.e., the same portions with the same subscription period and under the pull or notify mode). If this is the case, the

SSXD system sends the package key associated with such an existing package to the new-subscribed user, otherwise it generates a new one. Thus, different users many have the same package key only if they are subscribed to the same portions with the same subscription period and under the same mode (i.e., pull or notify).

Then, each time a user requests a pull or notify package, the SSXD service first checks which package portions have been modified from the previous request by the same user. Then it selects only those portions belonging to a non-expired component. Finally, it encrypts all those portions with the package key associated with the requested package and sends them to the requesting user.

### *Push Package Encryption Scheme*

As for the notify and pull mode, also the push<sub>on</sub> and push<sub>off</sub> modes are uniformly treated. Indeed, the difference between push<sub>on</sub> and push<sub>off</sub> mode is only on the event that triggers the delivery of a package or of some of its portions. In both cases, the system checks whether the portions that have been modified are contained in a non-expired component. The time of this check depends on the distribution mode. In case of push<sub>on</sub>, this check is executed upon each source update, whereas for push<sub>off</sub> this check is periodically executed.

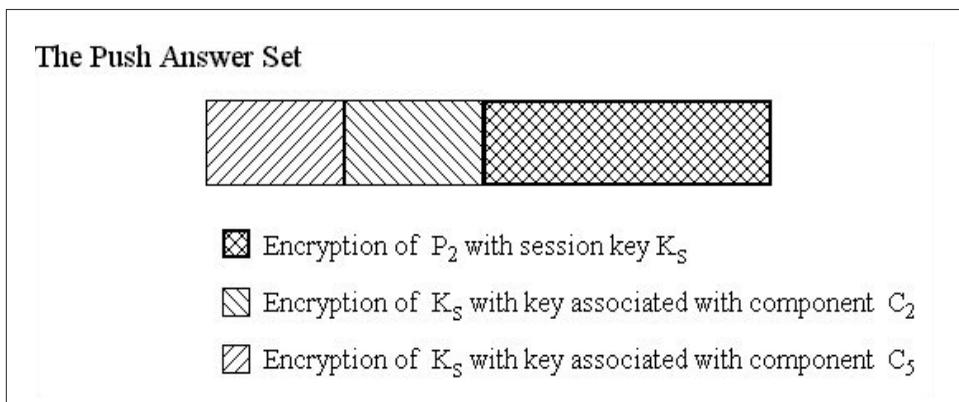
The push package encryption scheme aims at minimizing the number of keys that need to be generated by guaranteeing, at the same time, the security of information delivery. Indeed, the pull package encryption scheme is not efficient in this context, because the SSXD system has to broadcast the same portion of information, which may belong to several components or packages, to all the users entitled to access it. Thus, with the pull package encryption scheme it should encrypt the same content with a different key (i.e., the appropriate package key), and then send it to each different user. By contrast, to limit the number of keys that need to be generated, we use a technique based on session keys (Wool, 1998). In particular, the subscription Filtering module generates a different key for each component defined over the source. Once a user finishes the subscription phase, the SSXD system sends the user the keys associated with the components belonging to his/her packages. Then, when the SSXD system needs to send a modified portion, it encrypts it with a session key and broadcasts it to all the users subscribed to a package that contains a component defined over that portion and whose subscription period has not yet expired. Moreover, the SSXD system encrypts also the session key with the key associated with the component, and sends it together with the encrypted portions, generating thus a unique package, called *Push Answer Set*. Such an approach ensures that the portion can be accessed only by subscribers whose subscription period has not yet expired. Indeed, once a component is expired, the subscription period is over, the encryption key is no longer used. Moreover, such an approach ensures that only a limited number of keys need to be generated, since the set of components that can be defined over a source of information is limited.

### Example 2

Consider the scenario of Example 1. Suppose that on 03/30/03 a new issue of MyNewspaper is made available at the source, and that this issue contains an article written by Tom Red. Moreover, suppose that this issue does not contain any article on XML Security. Therefore, the portions defined in Example 1 that are influenced by this update are:  $P_1$ ,  $P_2$ , and  $P_4$ . Let us consider  $P_1$ . This portion denotes all the Frontpage elements of the instances of the DTD corresponding to the XML document in *Figure 2*. Thus, portion  $P_1$  contains the Frontpage of the new issue of MyNewspaper. When a portion has been modified, the subscription filtering module checks whether the modified portion belongs to a non-expired component, contained into a package having a  $push_{on}$  mode, and then creates the *Push Answer Set*. Considering the scenario of Example 1, portion  $P_1$  belongs only to a non-expired component (i.e.,  $C_1$ ), but this component belongs to package  $PA_2$ , which has a pull distribution mode. Therefore, the module does not generate the Push Answer Set for  $P_1$ . By contrast, when the SSXD system checks portion  $P_2$ , it verifies that  $P_2$  belongs to components  $C_2$ ,  $C_5$ , and  $C_6$ . All the packages containing these components, that is, packages  $PA_1$ ,  $PA_3$ , and  $PA_4$ , have a  $push_{on}$  distribution mode. Thus, the SSXD system checks the subscription period of each of these components. It thus verifies that component  $C_6$  is expired (on 02/28/03), whereas components  $C_2$ , and  $C_5$  are not. Thus, the Push Answer Set consists of the encryption of  $P_2$  with a session key  $K_s$ , the encryption of  $K_s$  with  $K_{C_2}$ , that is, the key associated with component  $C_2$ , and the encryption of  $K_s$  with  $K_{C_5}$ , that is, the key associated with component  $C_5$ . The Push Answer Set for portion  $P_2$  is illustrated in *Figure 5*.

Then, the system sends the Push Answer Set to the appropriate users, that for portion  $P_2$  are users  $U_1$  and  $U_3$ . Note that if the Push Answer is eavesdropped by other user, say  $U_4$ , he/she does not have the keys necessary to decrypt the

Figure 5: The Push Answer Set for portion  $P_2$



session key (indeed,  $U_4$  has only key  $K_{C_6}$  associated with component  $C_6$ ). Thus,  $U_4$  cannot decrypt portion  $P_2$ , since his/her subscription period to  $P_2$  is expired.

Finally, when the SSXD system checks portion  $P_4$ , it verifies that it belongs to a non-expired component (i.e.,  $C_4$ ), and that  $C_4$  is contained into package  $PA_2$ , whose distribution mode is Pull. Therefore, the Subscription Filtering module does not create the Push Answer Set for  $P_4$ . Let us suppose that after a week, user  $U_2$  requests package  $PA_2$ . In this case, the SSXD system first identifies the portion of  $PA_2$  belonging to non-expired components, that is,  $C_1$  and  $C_4$ . Then, it verifies for each portions of those components (i.e.,  $P_1$  and  $P_4$ ) if it has been modified from the last request by  $U_2$ . Suppose that the last request has been done on 03/29/03, therefore all portion contents are new for user  $U_2$ . Thus, the SSXD system creates the Pull Answer Set for  $U_2$ , by encrypting portions  $P_1$  and  $P_4$  with the key associated with package  $PA_2$ . Then, the SSXD system sends this Pull Answer Set to  $U_2$ .

## Access Control Filtering Module

The access control module has been designed in the framework of the AuthorX project (Bertino, 2001b) and it provides discretionary access control for XML documents. The access control model (Bertino, 2002b), on which the Access Control Filtering module is based, takes into account for policy specification XML document characteristics, the presence of DTDs/XML schemas describing the structure of documents at a schema level, and the types of actions that can be executed on XML documents (i.e., navigation, browsing, and update). The access control model provides a fine-grained access control, in that it is possible to specify policy that apply to collection of XML documents (for instance, all the documents instance of a DTD/XML Schema), or selected portions within a document (e.g., an element or attribute). Additionally, access control policies are characterized by a temporal dimension, in that it is possible to express policies that hold only for specific periods of time (such as for instance a particular day of the week). This is a relevant feature because very often users must have the right to access a document or a document portion only in specific periods of time.

Furthermore, access control model supports the specification of *user credentials* as a way to enforce access control based on user qualifications and profiles.

In the following sections we briefly explain the pull and push distribution, by mainly focusing on the information push mechanisms since it represents the most novel approach to data dissemination. We refer the interested reader to Bertino (2001b) for a detailed description of the information pull mechanism implemented in AuthorX, as well as to Bertino (2002b) for an exhaustive explanation of the access control model on which the Access Control Filtering module relies.

### *Pull Distribution*

Access control usually takes place in conventional DBMSs according to this traditional mode. If the pull mode is adopted, the users explicitly require the XML documents (or portions of documents) when needed. Upon a document request, Access Control Filtering module first verifies whether the requesting user is entitled to access the requested document, according to the specified access control policies. Based on these authorizations, the user is returned a *view* of the requested document(s), consisting of all and only those portions for which he/she has a corresponding authorization. When no authorizations are found, the access is denied.

### *Push Distribution*

The push distribution mode is particularly suitable for XML documents that must be released to a large community of users and which show a regular behavior with respect to their release. Also in this case, different users may have privileges to see different, selected portions of the same document based on the specified access control policies. To respect these different privileges, the mechanism enforcing information push must ensure that the correct view is delivered to each different user. Obviously the naïve solution to generate different physical views of the same document for each group of users to which the same access control policies apply is impracticable, since it could imply, in the worst case, the generation of a different view for each different user. Moreover, after the generation of the views, the Access Control Filtering module should properly deliver all these views to the interested users. In this scenario, we need to take in consideration that due to the possibly high number of users accessing an XML source, and the wide range of access granularities provided by the underlying access control model, the number of these views might become considerably large and, thus, such an approach cannot be practically applied.

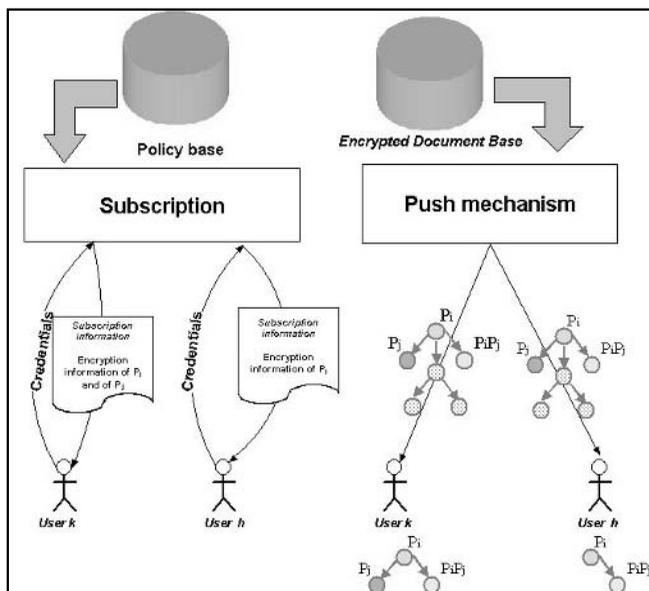
For these reasons, to efficiently support information push we adopt an approach based on the use of broadcast encryption techniques. According to this technique, the first step is the development of a strategy to generate a correct document encryption, that is, a document encryption ensuring that all the users are able to decrypt, and thus to access, all and only the authorized portions. In order to obtain a correct encryption, Access Control Filtering module encrypts all the portions of an XML document to which the same *policy configuration* applies with the same secret key, where with the term *policy configuration* we denote a set of access control policies. We refer to these encryptions as well-formed encryptions. The encrypted copy of the document is then placed into the Encrypted Document Base (EDB), which stores the encrypted copies of all the documents to be released under the push mode. We refer the interested reader to Bertino (2002b) for a detailed description of the algorithms computing the well-formed encryption. More precisely, the generation of the document encryption

is performed by two main algorithms. First, the XML document undergoes a *marking phase* in which each document node is marked with the access control policies that apply to it. Then, the second algorithm groups all the nodes with the same policy configuration and encrypts them with the same key. The same encrypted copy of the document is then distributed to all users, whereas each user only receives the key(s) for the portion(s) he/she is authorized to access.

The main issue in the generation of a well-formed encryption is that it could imply the management of a high number of secret keys. For instance, by using a traditional symmetric scheme, in the worst case the well-formed encryption implies the generation of  $2^{N_p}$  different secret keys, where  $N_p$  is the number of access control policies defined for the XML source, that is, one for each different policy configuration that could be generated from  $N_p$  policies. Moreover, the problem of key generation and management is further complicated by the temporal dimension associated with access control policies. To cope with such a high number of secret keys, in Bertino (2002a) it has been proposed a flexible key assignment scheme [adapted from (Tzeng, 2001)], which greatly reduces the number of keys that need to be managed.

More precisely, the adopted key assignment scheme relies on the framework depicted in *Figure 6*. Each user is required to register to the SSXD system, during a mandatory *subscription phase*. During the subscription phase, a user can be assigned one or more credentials, which are stored at the service site. As a result of the subscription phase, SSXD returns the user some information, called *subscription information*. In particular, the subscription information

Figure 6: Push mechanism enforcement



allows the user to decrypt all the nodes that he/she is entitled to access according to the specified access control policies and only for the set of time instants representing the validity period of the policies. More precisely, the subscription information consists of a set of parameters and additional information, one for each policy the user satisfies, called *encryption information*.

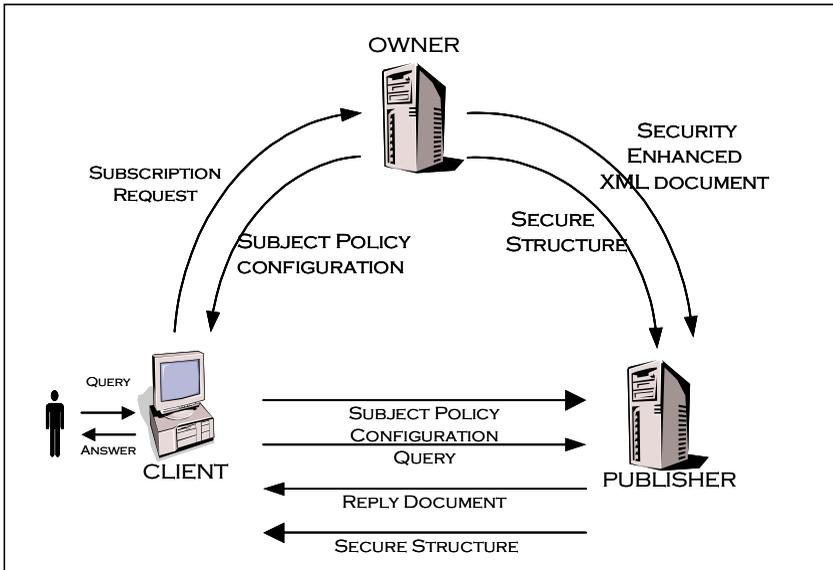
The problem of key generation and management is further complicated by the temporal dimension associated with our access control policies. To manage the temporal dimension of access control policies, the encryption key is obtained by combining the encryption information associated with a policy configuration with generic temporal information, denoted as  $w_t$ . Thus, at each time granule  $t$ , the system verifies whether a modification occurs in the XML source. In this case, the system groups the modified source portions according to their policy configurations. Then, for each group it generates a secret key by which it encrypts the modified portions. The secret key is the result of hashing the temporal information  $w_t$  together with the encryption information associated with the policy configuration marking the interested portions. Thus, in order to decrypt these portions, a user needs to have both the temporal information  $w_t$  and the encryption information associated with the policy configuration.

The most relevant aspects of the proposed key assignment scheme is that it is defined in such a way that, from the encryption information associated with a policy  $acp_i$ , it is possible to derive all and only the encryption information associated with policy configurations containing  $acp_i$ . Thus, using this management scheme the system has to manage only  $N_p$  different encryption information, whereas the others associated with the remaining policy configurations can be derived only when needed. Moreover, by using only the encryption information received during the subscription phase, a user is able to generate all the keys associated with the policy configurations he/she satisfies, and to generate all the  $w_t$ s corresponding to time granules belonging to the interval of validity of the policies he/she satisfies.

## SCALABILITY ISSUES IN SECURE SELECTIVE DATA DISSEMINATION

Given the possibly high number of subjects accessing a SSDI system, one of the most important issues is the scalability of the service. In the last few years, to overcome scalability problems, a new paradigm for data dissemination over the Web is receiving growing attention: the third-party architecture. A third-party architecture relies on the distinction between the *Owner* of the information and one or more *Publishers* that are entitled to publish the information (or portions of it) of the Owner and to answer queries submitted by subjects. A relevant security issue in this architecture is how the Owner can ensure a secure

Figure 7: A third-party architecture



dissemination of its data, even if the data are managed by a third-party that can be untrusted with the considered properties. In Bertino (2002c), we have proposed a third-party architecture for SSDI of XML data, by which a user is able to verify the authenticity, integrity and completeness of the answer received by an untrusted Publisher. In the following, we briefly summarize the overall architecture by explaining the role of each party in the architecture. Then, in next sections we give more details on authentication and completeness verification.

### Owner

In our approach, the Owner specifies access control policies over the XML source, and it sends them to the Publishers together with the documents they are entitled to manage. For each document, the Owner sends the Publisher also some security information needed by the Publisher to correctly answer subject's queries. More precisely, the Owner supplies the Publisher with information on which subjects can access which portions of the document, according to the access control policies it has specified. Access control policies are specified according to the access control model presented in Bertino (2002b). In this scenario, in order to ensure authenticity it is not enough that the Owner signs each document it sends to the Publisher, since the Publisher may return to a subject only selected portions of a document, depending on the query the subject submits and on the access control policies in place. For this reason, we propose an alternative solution which requires that the Owner sends the Publisher, in

addition to the documents it is entitled to manage, a summary signature (called *Merkle Signature*) for each managed document, generated using a technique based on Merkle hash trees (Merkle, 1989). The idea is that, when a subject submits a query to a Publisher, the Publisher sends him/her, besides the query result, also the signatures of the documents on which the query is performed. In this way, the subject can locally recompute the same bottom-up hash value signed by the Owner, and by comparing the two values he/she can verify whether the Publisher has altered the content of the query answer and can be sure of its authenticity.

All this additional information is encoded in XML and attached to the original document. The original document complemented with this additional information is called the *security enhanced XML document* (called SE-XML, hereafter). An example of SE-XML document is presented in *Figure 8*. The information contained into the SE-XML document allows a subject to verify the authenticity of the information returned by the Publisher, but it is not sufficient to make a subject able to verify the completeness of a query result. For this reason, the Publisher receives from the Owner some additional information about the structure of the original XML document, which must be sent in turn to interested subjects. This information is encoded into an XML document, called *secure structure* (called ST-XML, hereafter), which contains an obfuscated version of the structure of the XML document. Moreover, the secure structure is complemented with its Merkle Signature, to prevent Publisher alterations. An example of ST-XML document is presented in *Figure 11*. Further details on the ST-XML generation will be provided in section devoted to the completeness property.

### *Subjects*

As depicted in *Figure 7*, subjects are required to register with the Owner, through a mandatory subscription phase. As a result of the subscription process, the Owner returns the subject a data object, called the *subject policy configuration*, which stores information on the access authorizations that apply to the subject, according to the policies stated by the Owner. Since in a third-party architecture the subject policy configuration plays the role of a certificate, it must be signed with the private key of the Owner, to prevent the subject from altering its content.

### *Publisher*

Once the subscription phase has been completed, a subject can submit queries to a Publisher. As reported in *Figure 7*, when a subject submits a query, it also sends the Publisher its subject policy configuration to enable the Publisher to determine which access control policies apply to the subject. On the basis of the subject policy configuration and the submitted query, the Publisher computes a *view* of the requested document(s), which contains all and only those portions

of the requested document(s) for which the subject has an authorization according to the access control policies in place at the Owner site. In order to verify the authenticity of the answer the subject must be able to locally recompute the same bottom-up hash value signed by the Owner (i.e., the Merkle signature), and to compare it with the Merkle signature generated by the Owner and inserted by the Publisher into the answer. Since the view computed by the Publisher may not contain all the nodes of the requested documents, the subject may not be able to compute the bottom-hash value over the whole document by considering only the nodes in the view. Thus, the Publisher complements the view with additional information (e.g., hash values computed over the document portions not contained in the view). Both the view and the additional information are locally computed by the Publisher by considering only the SE-XML version(s) of the requested document(s). The result is an XML document, called *reply document*. In addition to the reply document, the Publisher sends the subject also the *secure structure* associated with the document on which the query applies. Upon receiving the reply document, the subject can verify, by using only the information in the reply document itself, the authenticity of the answer. Additionally, the subject can make some verification on the completeness of the query result by using the information contained in the *secure structure* received by the Publisher.

In the next sections we describe in more in detail how a subject can verify the authenticity and the completeness of a query answer.

## Authenticity Property

In this section we focus on authenticity verification. More precisely, we consider each single party of the architecture by pointing out its role in the authenticity verification process.

### *The Owner: Generation of the SE-XML Document*

To make the Publisher able to answer subject queries and the subjects able to verify the authenticity of the query answers, the Owner inserts into the XML documents sent to the Publishers two distinct pieces of information: the *Merkle signature*, and a set of *Policy Information*. In the following we briefly explain both of them.

### *Merkle Signature*

Traditional digital signature techniques are not suitable to ensure authenticity in a third-party architecture. Indeed, since the Publisher may return a subject only selected portions of an XML document, depending on the query the subject submits and on the access control policies in place, the subject could not be able to validate the Owner signature, which is computed over the whole XML document. We need, thus, a mechanism allowing the Publisher to prune some

nodes and, at the same time, allow the subject to verify the Owner signature having only the returned view and some additional information.

In our approach, we propose a different way to sign an XML document, which is based on an alternative mode to compute the *digest*. The function we use to compute the digest value is the Merkle function (denoted in the following as  $MhX()$ ), which exploits the Merkle tree authentication mechanism proposed in Merkle (1989). By this function, it is possible to univocally associate a hash value with a whole XML document through a recursive computation. As depicted in *Figure 9*, the basic idea is to associate a hash value with each node in the graph representation of an XML document. The hash value associated with an attribute is obtained by applying a hash function over the concatenation of the attribute value and the attribute name. By contrast, the hash value associated with an element is the result of the same hash function computed over the concatenation of the element content, the element tag name, and the hash values associated with its children nodes, both attributes and elements. Once the digest of the XML document is computed (i.e., the Merkle hash value of the root of the document), it is then encrypted with the private key of the Owner, generating what we call the *Merkle Signature*.

By using Merkle signatures the Owner is able to apply a unique digital signature on an XML document by ensuring at the same time the authenticity and integrity of both the whole document, as well as of any portion of it. The Merkle

*Figure 8: An example of SE-XML document*

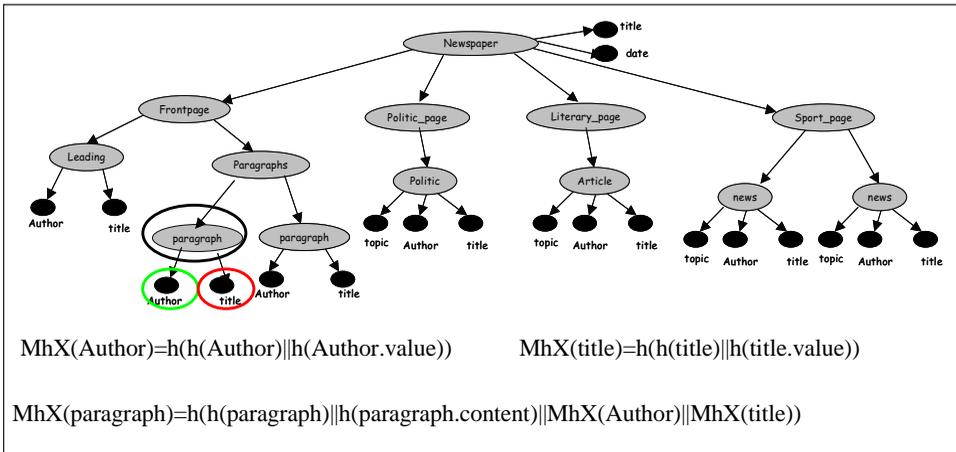
```

<?xml version="1.0" encoding="UTF-8" ?>
<Annual-report Year="2002" name="University of Milano" Sign="CG4g3D8/mPVV/t+T201kZRFhdio=">
  <Policy>1,2,3,4,5,6,7</Policy>
  <Assets>
    <Asset Dept="DICO">
      <Expenses Tot="..." PC_ATTR="08" />
      <Funds>
        <Fund Funding-Date="15/09/2002" Type="CNR" Amount="..." PC_ATTR="080808" />
      </Funds>
    </Asset>
    <Asset Dept="EED">
      <Expenses Tot="..." PC_ATTR="02" />
      <Funds>
        <Fund Funding-Date="01/20/2002" Type="CNR" Amount="..." PC_ATTR="060602" />
        <Fund Funding-Date="06/01/2002" Type="MURST" Amount="..." PC_ATTR="060602" />
      </Funds>
    </Asset>
  </Assets>
  <Patents>
    <Patent date="02/26/2002" Id-Pat="..." Dept="DICO" PC_ATTR="808080">
      <Short-descr PC="90">.....</Short-descr>
      <Tech-details PC="80">.....</Tech-details>
      <Authors PC="90">.....</Authors>
    </Patent>
    <Patent date="05/12/2002" Id-Pat="..." Dept="EED" PC_ATTR="202020">
      <Short-descr PC="60">.....</Short-descr>
      <Tech-details PC="20">.....</Tech-details>
      <Authors PC="60">.....</Authors>
    </Patent> </Patents>

</Annual-report>

```

Figure 9: The Merkle function



signature of an XML document is inserted by the Owner into the SE-XML document, by means of the Sign attribute (see Figure 8).

### Policy Information

In addition to the Merkle signature, the Owner inserts in the SE-XML document also information about the access control policies applied on the corresponding XML document. More precisely, the Owner inserts into each element, to which at least one policy applies, information about the policy configuration applied to such an element. Policy information is specified at the element level, since different access control policies can apply to different portions (i.e., elements and/or attributes) of the same document. The idea is to encode information about the set of policies that apply to a specific element into a string of hexadecimal values, called *policy configuration*, and to store this string as an additional attribute of the corresponding element within the SE-XML document (i.e., PC attribute). Whereas, to store information about policies that apply to attributes, we propose a slightly different approach. The idea is to store into a unique attribute associate with an element, called PC\_ATTR, the concatenation of the policy configurations of the element attributes.

More precisely, the policy configuration is the hexadecimal encoding of a binary string where each bit represents the state of a policy (1, if the policy is applied to the node, 0 otherwise). This means that the *i*-th bit of a policy configuration is set to 1 if and only if the policy whose identifier is *i* is applied to that element. We have to note that, since the number of policies defined by the Owner can be very large, this approach implies very large identifiers, and, as a consequence, a large string as policy configuration. For this reason, we have proposed to represent a *local policy configuration*, that is, a binary string built

by considering only the policies applied to the document. Thus, we need to insert in addition to the policy configuration also information about the set of access control policies applied to a document. This information is contained into the Policy element (see *Figure 8*).

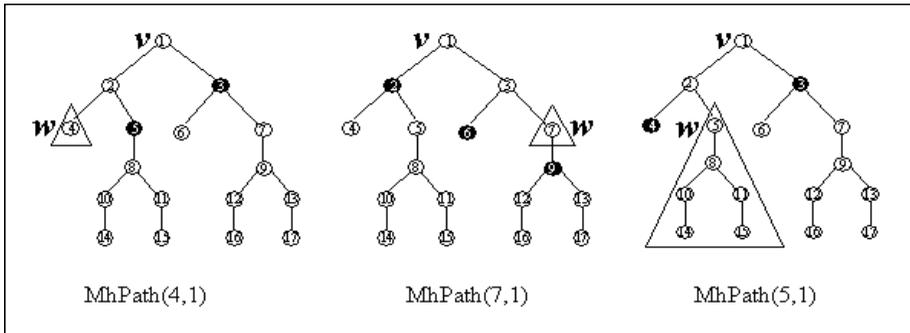
### **The Publisher: Generation of the Reply Document**

When a subject submits a query to a Publisher, together with the query, he/she also sends information on the policies he/she satisfies (i.e., the subject policy configuration). On the basis of the subject policy configuration and the submitted query, the Publisher computes a view of the requested document, which contains all and only those portions of the requested document for which the subject has an authorization. Then, besides the query result, the Publisher returns the requesting subject also the Merkle signatures of the documents on which the query is performed. In this way, the subject can locally recompute the same bottom-up hash value signed by the Owner, and by comparing the two values he/she can verify whether the Publisher has altered the content of the query answer and can be sure of the source authenticity. The problem with this approach is that, since the subject may be returned only selected portions of a document, he/she may not be able to recompute the Merkle signature, which is based on the whole document. For this reason, the Publisher sends the subject a set of additional hash values, referring to the missing portions. This additional information is called *Merkle Hash Paths*. Both the view and the Merkle Hash Paths are locally computed by the Publisher by considering only the security enhanced version(s) of the requested document(s). The result is an XML document, called *Reply document*. Upon receiving the reply document, the subject can verify, by using only its content (i.e., the Merkle signature, the Merkle hash paths, and the query answer), the authenticity of the answer, without interacting with the Owner. The idea is that by performing a computation on the Merkle hash Paths and the nodes of the query answer and by comparing the result with the Merkle signature of the XML document, the subject is able to verify the authenticity of the answer. The next section deals with Merkle hash paths computation.

#### *Merkle Hash Paths*

Merkle Hash paths can be intuitively defined as the set of the Merkle Hash values of those nodes pruned during query evaluation. In general, given two nodes  $v$ ,  $w$  such that  $v$  belongs to  $\text{Path}(w)$  (where  $\text{Path}(w)$  denotes the set of nodes connecting a node  $w$  to the root of the corresponding document), the Merkle Hash path between  $w$  and  $v$ , denoted as  $MhPath(w,v)$ , is the set of Merkle Hash values needed to compute the Merkle Hash value of  $v$  having the Merkle Hash value of  $w$ . More precisely, the Merkle Hash path between  $w$  and  $v$  consists of all the Merkle Hash values of  $w$ 's siblings, together with the Merkle Hash values of all the siblings of the nodes belonging to the path connecting  $w$  to  $v$ .

Figure 10: Examples of Merkle Hash paths



To better clarify how the proposed approach works, let us consider *Figure 10*, which depicts three different examples of Merkle hash paths. In the graph representation adopted in *Figure 10* we do not distinguish elements from attributes, by treating them as generic nodes. In the figure, triangles denote the view returned to the subject, whereas black circles represent the nodes whose Merkle hash values are returned together with the view, that is, the Merkle hash paths. In the first example, the view consists of a leaf node  $w$ . In such a case, the Merkle hash path between nodes 4 and 1 consists of the Merkle hash values of nodes 5 and 3. Indeed, by using node  $w$  (i.e., 4) and the Merkle hash value of node 5 it is possible to compute the Merkle hash value of node 2. Then, by using the Merkle hash values of 2 and 3, it is possible to compute the Merkle hash value of 1. In the second tree depicted in *Figure 10*, the view consists of a non-leaf node. In such a case  $\text{MhPath}(7,1)$  also contains the Merkle hash value of the child of node 7, that is, node 9. Thus, by using the Merkle hash value of node 9 and node 7, it is possible to compute the Merkle hash value of 7. Then, by using this value and the Merkle hash value of node 6, it is possible to generate the Merkle hash value of node 3. Finally, by using the Merkle hash values of nodes 3 and 2 it is possible to generate the Merkle hash value of node 1. By contrast, in the third example, the view consists of the whole sub-tree rooted at node 5. In such a case,  $\text{MhPath}(5,1)$  does not contain the hash values of the children of node 5. Indeed, having the whole subtree rooted at 5, it is possible to compute the Merkle hash value of node 5 without the need of any further information.

## Completeness Property

The Merkle signature allows a subject to detect an alteration on the content of an XML document (or portions of it). More precisely, the subject is able to verify that the content/value and the tagname/name of an element/attribute have not been modified by an intruder. However, by using only the Merkle signature, the subject is not able to verify if the Publisher sends him/her the correct view,

that is, a view containing all the portions of the XML document answering the submitted query and satisfying access control policies.

To make the subject able to verify the completeness of the view, we have introduced the *secure structure* of an XML document, which basically gives information about the structure of the documents on which the query is submitted. Indeed, by secure structure of an XML document we mean the XML document without its element contents, thus, containing only the names of the tags and all the attributes of the XML document (that is, the values and names of the attributes). Moreover, to avoid allowing the subject to see tagname and attribute values of portions he/she may not be authorized to access, we impose that each tag and attribute name and value has to be hashed with a standard hash function, before being inserted into the secure structure (see *Figure 11*).

This information makes the subject able to locally perform on the secure structure all queries whose conditions are against the document structure of the original document or on the attribute values. Indeed, these queries specify their conditions by using only the tag/attribute names, and the attribute value, which are information all contained as hash values in the secure structure. The proposed solution for completeness verification implies the translation of the submitted query  $q$ , by substituting the tag/attribute name and attribute values with the corresponding hash values. Afterwards, the translated query can be evaluated on the secure structure. In such a way, under the assumption of a collision-resistant hash function, the node-set resulting by the evaluation of the query on the secure structure corresponds to all and only the nodes of document  $d$ , answering query  $q$ . We have to note that since the Publisher generates the view according to the access control policies stated by the Owner, during the completeness verification the subject must consider also the access control policies specified on the document. For this reason, we have inserted in the

*Figure 11: An example of secure structure*

```
<?xml version="1.0" encoding="UTF-8"?>
<x-82592553 x-4639474="rVR5DQ" x-67303774="QTQXS" Sign="OD2mc9aVV/tP4g3TG+1kr4sFhdio=">
  <Policy>1,2,3,4,5,6,7 </Policy>
  <x-1915689488>
    <x-785490824 x-40276037="PlcZUo">
      <x-590292021 x-57205665="..." PC_ATTR="08"/>
      <x-13947931>
        <x-1037159472 x-122584813="fNhtL" x-0260379="hgKID" x-93640287="..." PC_ATTR="080808"/>
      </x-13947931>
    </x-785490824>
    <x-785490824 x-40276037="pKGEs">
      <x-590292021 x-57205665="..." PC_ATTR="02"/>
      <x-13947931>
        <x-1037159472 x-122584813="gPd39" x-0260379="hgKID" x-93640287="..." PC_ATTR="060602"/>
      </x-13947931>
    </x-785490824>
  </x-1915689488>
</x-82592553>
```

secure structure also the Policy element, PC, and PC\_ATTR attributes, that is, the attributes containing the policy information of the elements and attributes.

Additionally, in order to prevent alterations by the Publisher, the Owner computes the Merkle Signature of the secure structure, and, similarly to the SE-XML document, it sends this signature to Publishers together with the corresponding secure structure (i.e., the Sign attribute).

## CONCLUSION

The chapter has focused on SSDI systems. We have first provided an overview of the work carried out in the field then we have focused on the security properties that an SSDI system must satisfy and on some of the strategies and mechanisms that can be used to ensure them. More precisely, since XML is the emerging standard today for data exchange over the Web, we have cast our attention on SSDI systems for XML documents (SSXD). As a result, we have presented a SSXD system providing a comprehensive solution to XML documents. Finally, since the third-party architecture is receiving growing attention as a new paradigm for data dissemination over the Web, we have discussed a new architecture for SSXDI system, which has the benefit of improving the scalability.

## REFERENCES

- Alert. (2003). *ALERT project web page*. Available on the World Wide Web at: <http://alert.uni-duisburg.de/>.
- Altinel, M., & Franklin, M.J. (2000, September). Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference* (pp. 53-64), Cairo, Egypt.
- Belkin, N.J., & Croft, B.W. (1987). Retrieval techniques. *Annual Review of Information Science and Technology* (109-145). Elsevier.
- Bertino, E., & Ferrari E. (2002b). Secure and selective dissemination of XML documents. *ACM Transactions on Information and System Security (TISSEC)*, 5 (3), 290-331.
- Bertino, E., Carminati, B., & Ferrari, E. (2001a). A secure publishing service for digital libraries of XML documents. In *Information Security Conference (ISC01)* (pp. 347-362), LNCS 2200, Malaga, Spain. Springer-Verlag.
- Bertino, E., Castano, S., & Ferrari E. (2001b, May/June). AuthorX: A comprehensive system for securing XML documents. *IEEE Internet Computing*, 5 (3), 21-31.
- Bertino, E., Carminati, B., & Ferrari, E. (2002a, November). A temporal key management scheme for broadcasting XML documents. In *Proceedings*.

- of the 9th ACM Conference on Computer and Communications Security (CCS'02), Washington. ACM Press.
- Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., & Gupta, A. (2002c). *Selective and authentic third-party distribution of XML document*. Accepted for publication with IEEE Transactions on Knowledge and Data Engineering (TKDE), February 2002. Available on the World Wide Web: [http://ebusiness.mit.edu/research/papers/187\\_Gupta\\_XML.pdf](http://ebusiness.mit.edu/research/papers/187_Gupta_XML.pdf).
- Bray, T., & Paoli, J., & Sperberg-McQueen, C.M. (1998, February). *Extensible Markup Language (XML) 1.0*. W3C Recommendation.
- Carzaniga, A., Rosenblum, D.S., & Wolf, A.L. (2000). Achieving scalability and expressiveness in an Internet-scale event notification service. In *Symposium on Principles of Distributed Computing*, (pp. 219-227).
- Chen, J., DeWitt, D., Tian, F., & Wang, Y. (2000, May). NiagaraCQ: A scalable continuous query system for Internet databases. In *Proceedings of the ACM SIGMOD Conference*, Dallas, Texas.
- Diao, Y., & Franklin, M.J. (2003). High-performance XML filtering: An overview of YFilter. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*.
- Fiat, A., & Noar, M. (1994). Broadcast encryption. In *Advances in Cryptology (Crypto 93)*, (pp. 480-491). LNCS 773. New York: Springer-Verlag.
- Housman, E.M. (1973). Selective dissemination of information. *Annual Review of Information Science and Technology*, 8, 221-241.
- Liu, L., Pu, C., & Tang, W. (1999, January). Continual queries for Internet scale event-driven information delivery. Special Issue on Web Technologies, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (Special Issue on Web Technologies).
- Loeb, S., & Terry, D. (1992). Information filtering. *Communications of the ACM*, 12 (35), 26-28.
- Luhn, H.P. (1958). A business intelligent system. *IBM Journal of Research and Development*, 4 (2), 314-319.
- Luhn, H.P. (1961). Selective dissemination of new scientific information with the aid of electronic processing equipment. *American Documentation*, 12, 131-138.
- Merkle, R.C. (1989). A certified digital signature. *Advances in Cryptology-Crypto '89*.
- Pereira, J., Fabret, F., Llibat, F., Jacobsen, H.A., & Shasha, D. (2001). *WebFilter: A High-throughput XML-Based Publish and Subscribe System*.
- Robertson, S.E. (1977). The probability ranking principle in IR. *J. Doc.*, 33 (4), 294-304.
- Salton, G. (1989). *Automatic text processing*. Addison Wesley, 1989.

- Salton, G., & McGill, M.J. (1983). *Introduction to modern information retrieval*. McGraw-Hill.
- Stallings, W. (2000). *Network security essentials: Applications and standards*. Prentice Hall.
- Strom, R., Banavar, G., Chandra, T., Kaplan, M., Miller, K., Mukherjee, B., Sturman, D., & Ward, M. (1998). *Gryphon: An information flow based approach to message brokering*.
- Terry, D. B., Goldberg, D., Nichols, D. A., & Oki, B. M. (1992, June). Continuous queries over append-only databases. In *Proceedings of ACM SIGMOD Conferences* (pp. 321-330).
- Tzeng, Wen-Guey. (2002). A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14 (1), 182-188.
- Wool, A. (1998, November). Key Management for Encrypted Broadcast. In *Proceedings of the 5th ACM Conference Computer and Communication Security* (pp. 7-16), San Francisco, CA.
- Xpath. (1999). *World Wide Web Consortium. XML Path Language (Xpath), 1.0*. W3C Recommendation. Available on the World Wide Web: <http://www.w3.org/TR/xpath>.
- Yan, T.W., & Garcia-Molina, H. (1994a). Index structures for selective dissemination of information under the Boolean model. *ACM Transactions on Database Systems (TODS)*, 2 (19), 332-334.
- Yan, T.W., & Garcia-Molina, H. (1994b). Index structures for information filtering under the vector space model. In *Proceedings of International Conference on Data Engineering* (pp. 337-347). IEEE Computer Society Press.
- Yan, T.W., & Garcia-Molina, H. (1999). The SIFT information dissemination system. *ACM Transactions on Database Systems*, 4 (24), 529-565. <Article topic="Movies" Author="..." Title="...">

## ENDNOTES

- <sup>1</sup> In this section, the term 'user profile' means the set of information associated to a user, relevant for the selective dissemination (e.g., the user preference, subscription information or the access control policies applied to the user).

## Chapter VII

# Multimedia Security and Digital Rights Management Technology

Eduardo Fernandez-Medina, Universidad de Castilla-La Mancha, Spain

Sabrina De Capitani di Vimercati, Università di Milano, Italy

Ernesto Damiani, Università di Milano, Italy

Mario Piattini, Universidad de Castilla-La Mancha, Spain

Pierangela Samarati, Università di Milano, Italy

## ABSTRACT

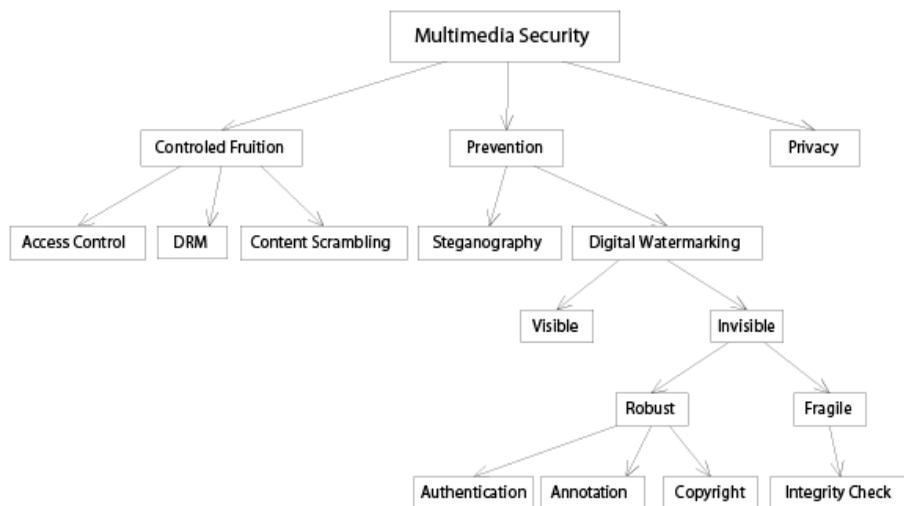
*Multimedia content delivery applications are becoming widespread thanks to increasingly cheaper access to high bandwidth networks. Also, the pervasiveness of XML as a data interchange format has given origin to a number of standard formats for multimedia, such as SMIL for multimedia presentations, SVG for vector graphics, VoiceXML for dialog, and MPEG-21 and MPEG-7 for video. Innovative programming paradigms (such as the one of web services) rely on the availability of XML-based markup and metadata in the multimedia flow in order to customize and add value to multimedia content distributed via the Net. In such a context, a number of*

*security issues around multimedia data management need to be addressed. First of all, it is important to identify the parties allowed to use the multimedia resources, the rights available to the parties, and the terms and conditions under which those rights may be executed: this is fulfilled by the Digital Rights Management (DRM) technology. Secondly, a new generation of security and privacy models and languages is needed, capable of expressing complex filtering conditions on a wide range of properties of multimedia data. In this chapter, we analyze the general problem of multimedia security. We summarize the most important XML-based formats for representing multimedia data, and we present languages for expressing access control policies. Finally, we introduce the most important concepts of the DRM technology.*

## INTRODUCTION

Multimedia information (such as free text, audio, video, images, and animations) is of paramount importance for human-computer interaction in a number of application fields, such as entertainment, distance learning, pay-per-view/listen business, and collaboration among others. Multimedia security problems are very similar to traditional security problems, but there are some important aspects that make more complex the solutions: traditional multimedia documents usually are monolithic, without a clearly defined internal structure, they are conceived to be discretionally distributed, they are easy to clone, easy to modify, to remove, to manipulate, and so on. Recently, the importance of complementing binary multimedia data with metadata in the form of XML tagging has been fully realized. XML-based standards formats for multimedia, such as SVG (SVG, 2001), SMIL (SMIL, 2001), and VoiceXML (VoiceXML, 2002) make the internal structure of multimedia flows available to consumer applications and devices. These languages describe the internal properties of the multimedia documents, making possible to refer to structural or semantic components, such as keywords, and graphical/audio/audiovisual properties.

The wealth of semantics- and structure-related information carried by these new XML-based multimedia formats suggests that the time has come to develop novel approaches for controlling access and fruition of multimedia content. In particular, a number of security issues around multimedia data management need to be addressed. *Figure 1* illustrates a *Multimedia Security Taxonomy* according to which multimedia security encompass techniques to determine who can use multimedia content and at what conditions (*controlled fruition*); to prevent multimedia content from being illegally copied (*prevention*); and to protect multimedia content while it is being transmitted or stored (*privacy*). In the following, we describe these problems in more detail.

*Figure 1: Multimedia Security Taxonomy*

## Multimedia Fruition Control

Fruition control can be seen as a modern view of access control, where unauthorized accesses are prevented, complex conditions for authorized users can be specified (e.g., timing delay, quality of the rendering), fine-grained encryption can be enforced, and policies and infrastructures are managed. In a nutshell, fruition control techniques try to define the usage conditions of digital products from the creation to the consumption.

They deal mainly with two concepts: digital right management (DRM) and access control policies. Basically, DRM includes a set of techniques to specify rights and conditions associated with the use and protection of digital contents and services. Access control (AC) policies specify restrictions of individuals or application programs to obtain data from, or to place data into, a storage device. The main difference between DRM and AC is that DRM represents fruition policies specified on the resources, regardless of the system used to deliver them, while AC represents fruition policies that take into consideration the delivery service. Access control and DRM are discussed in later sections.

## Multimedia Prevention

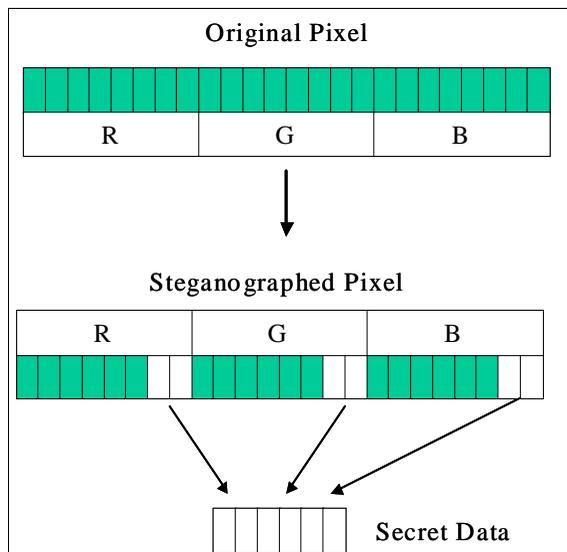
With the proliferation of multimedia content and of the concerns of privacy on the Internet, research on information hiding has become even more pressing. Information is collected by different organizations and the nature of multimedia content allows for the exact duplication of material with no notification that the material has been copied. Systems to analyze techniques for uncovering hidden information and recovering destroyed information are thus of great importance

to many parties (e.g., law enforcement authorities in computer forensics and digital traffic analysis). In such a context, we briefly explore two important labeling techniques: *steganography* and *watermarking*.

Steganography is the art of hiding information inside other information (in our case, multimedia documents). The hidden information can be, for example, a trademark or the serial number of a product, and can be used to detect copyright violations and to prosecute them. Steganography dates back to ancient Greece, and the initial purpose was to hide messages inside other messages, for war, espionage, and many other reasons. Some curious historical steganographic methods are invisible ink, pictographs, null cipher, positional codes, and deliberate misprints.

According to the type of multimedia document, different properties can be exploited to hide information (Johnson et al., 2000). For instance, for text, it is possible to code messages by changing the space between text lines (*line-shift coding*) or between words (*word-shift coding*), or by altering certain text features (*feature coding*) such as vertical endlines of letters. For images, the most common techniques include the *least significant bit insertion (LSB)*, *redundant pattern encoding*, and *spread spectrum method*. The idea behind the LSB algorithm is to insert the bits of the hidden message into the least significant bits of the pixels. *Figure 2* illustrates an example with a 24-bit pixel. Here, the secret data are inserted in the last two bits of each byte. The *redundant pattern encoding* consists in painting a small message over an image

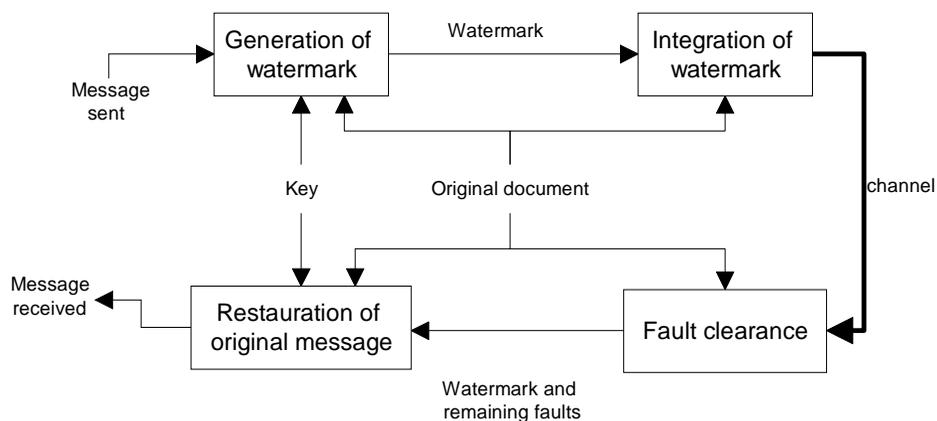
*Figure 2: Example of integrating hidden data in a pixel representation*



many times. The *spread spectrum method* scatters an encrypted message throughout an image (not just the least significant bit). For audio, the most common techniques modify bits of the audio files or some audio properties. For instance, the *low-bit encoding* replaces the least significant bit of each sampling point with a coded binary string. The *phase coding* method works by substituting the phase of an initial audio segment with a reference phase that represents the data. All these techniques introduce changes in documents that humans are not able to identify, but a computer can identify, thus obtaining the hidden message. Unfortunately, many steganographic techniques are not robust against simple compression. Some compression mechanisms [such as JPEG (JPEG, 2003)] lose the least significant bits to be more efficient, thus also losing the hidden information embedded in them. Some proposals try to overcome this drawback [see, for example, Currie & Irvine (1996) and Koch & Zhao (1995)]. A complete overview of steganography techniques can be found in Johnson et al. (2000) and Sellars (1999).

Watermarking describes techniques used to include hidden information by embedding the information into some innocent-looking cover data (Loo, 2002). Watermarking and steganography are closely related in that both employ mechanisms for hiding information. However, in steganography, the object of communication is the hidden message and the main goal is to keep the message (or the communication of) from being detected. Watermarks, on the other hand, can be considered as attributes of the object in which they are inserted. The existence of an embedded watermark may be known or unknown. Also, steganography can incorporate encryption, while watermarking is noncryptographic. *Figure 3* shows the scheme that represents how digital watermarking is managed. Let  $m$  be a message to be sent. A process generates

*Figure 3: Scheme of watermarking flows*



the watermark  $w$  by using a secret key  $k$ . Next, this watermark is integrated into the message, thus obtaining a new message  $m'$ . When message  $m'$  is received, the watermark  $w$  is extracted by using the key  $k$  and the original message is recovered. Once the watermark has been extracted, it is possible to compare it with the initial watermark to detect whether the message has been modified.

Digital watermarks can fall into two different classes, namely, *visible* and *invisible*. Visible watermarks are visual patterns that are overlaid on digital content. It is not possible to remove visible watermarks without destroying the digital content. Digital watermarks can also be classified as *robust* or *fragile*. Robust watermarks show strong resistance against accidental and malicious attacks such as content alteration, compression, and filtering. Fragile watermarks have just the opposite characteristics in that they drastically change in case of any alteration of the digital content. Watermarking is useful for many applications (Loo, 2002):

- *Copyright protection.* With embedded data, copyright holders can verify the ownership of copyrighted contents with watermark in case of intentional or unauthorized distribution. New emerging metadata-based access control techniques and DRM languages provide additional information that is separated from the multimedia document. This information could be integrated as a watermark in the multimedia document. Access control and DRM policies could then be directly enforced at the client site by extracting and interpreting the watermark. Note however that this kind of watermarking information does not prevent people from copying the digital contents.
- *Copy protection.* A copy protection mechanism prevents users from making unauthorized copies of digital data.
- *Fingerprint for pirate tracing.* Watermarks are used in fingerprinting applications to identify the legal recipient of the digital contents and typically are used together with copyright protection watermarks.
- *Authentication.* Authentication watermarking is a technology that prevents forgery of digital contents. If the original digital content is modified, the watermark is destroyed. For instance, *Figure 4* shows two photos: on the left there is the original photo and on the right the modified one. Authentication watermarking allows us to detect that the photo on the right has been modified.

All watermarking techniques have three major requirements. Watermark should be *robust* against accidental or malicious attacks, *imperceptible*, and carry the *required number of bits*. These three requirements conflict with each other. For instance, increasing the number of embedded bits increases the capacity but decreases the robustness.

*Figure 4: Example of authentication problem*

## Multimedia Privacy

While the privacy of “traditional data” is a well-known problem, the variety of types and usages of multimedia information makes the multimedia privacy problem fuzzier. In fact, the relationship between multimedia data invasion and privacy has not yet been clearly described. Probably we are not able to perceive privacy problems in music, or in movies, but multimedia is much more than that. For instance, textual information can denote the way things are presented; audio documents can indicate tone of voice, accent or dialect; and video can show dress, look of user, and so on (Adams, 2000). There is not much work in multimedia privacy. However, some interesting work about the relationship between multimedia privacy and users’ perceptions can be found in Adams & Sasse (1999a) and Adams & Sasse (1999b).

## XML METADATA FOR MULTIMEDIA

The importance of complementing binary multimedia data with descriptive metadata in the form of XML tagging has been fully realized only recently. The pervasiveness of XML as a data interchange format has given rise to a number of standard formats for multimedia representation such as SMIL (SMIL, 2001) for multimedia presentations, SVG (SVG, 2001) for vector graphics, VoiceXML (VoiceXML, 2002) for dialog, and MPEG-21 (MPEG-21, 2002) and MPEG-7 (MPEG-7, 2002) for video. Innovative programming paradigms (such as the one of web services) rely on the availability of XML-based markup and metadata in the multimedia flow in order to customize and add value to multimedia content distributed via the Net. SVG and VoiceXML make the internal structure of multimedia flows available to a variety of consumer applications and devices. More ambitious efforts, such as MPEG-21 and MPEG-7, are aimed at achieving the same results for video. Intelligent applications like search indexes, topic maps

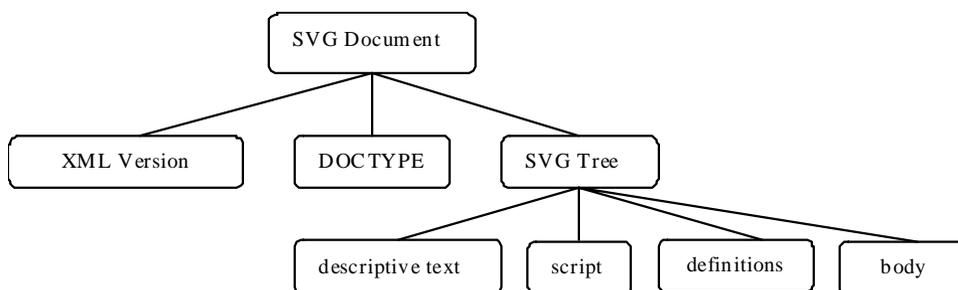
or browsable directories can use XML tagging and auxiliary metadata to identify the structural or semantics-related components of multimedia flows, such as keywords, key frames, audiovisual summaries, semantic concepts, color histograms, and shapes, as well as recognize speech. XML formats are paving the way to a new generation of applications even in more traditional fields such as image processing. For instance, raster graphical formats (e.g., *GIF* or *JPEG*) have severe limitations, in as much they do not carry any information that can be queried, re-organized, or searched through. By contrast the *Scalable Vector Graphics* (*SVG*), an XML-based language for describing two-dimensional vector and mixed vector/raster graphics, works well across platforms, across output resolutions and color spaces. Also, *SVG* clearly specifies the image structure, allowing applications to process data at a finer level of granularity. The wealth of semantics- and structure-related information carried by new XML-based multimedia formats suggests that the time has come to enrich traditional, coarse-grained access control models with a number of new concepts that are specific to the nature and meaning of multimedia data. An interesting consequence of using XML for representing multimedia metadata is that fine-grained access policies can be specified. So, XML elements used to refer to multimedia components inside policies can be mapped to XML tags and metadata in the data flow. Policy-to-data mapping can be customized to the particular XML-based multimedia format under discussion, achieving fast and effective enforcement via either data filtering or encryption.

In the following subsections we shortly describe these XML-based multimedia standard formats.

## Scalable Vector Graphics

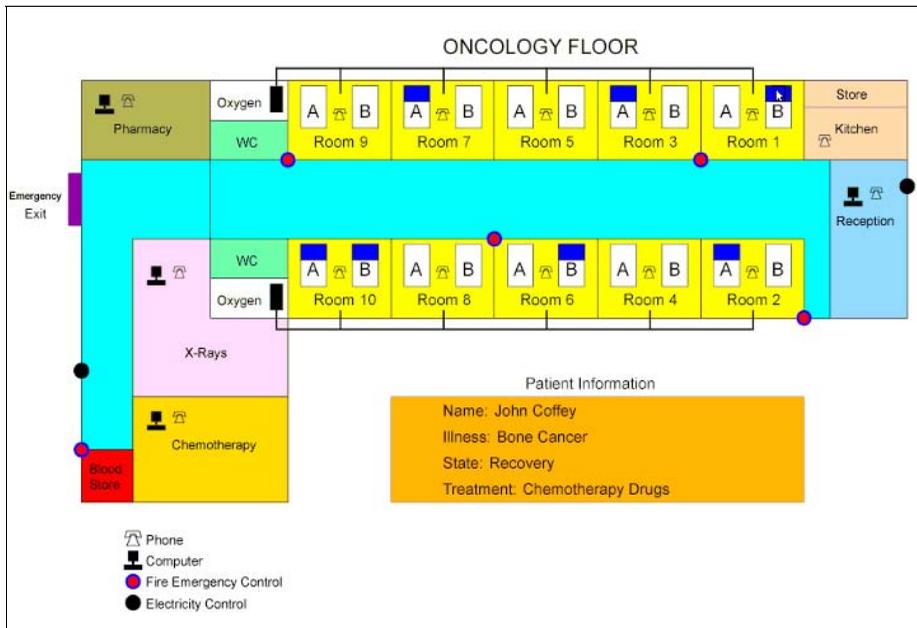
*SVG* (*SVG*, 2001) is a language for describing two-dimensional vector and mixed vector/raster graphics in XML. An *SVG* document has a flexible structure, composed of several optional elements placed in the document in an arbitrary order. *Figure 5* shows the general structure of a *SVG* document.

*Figure 5: General structure of an SVG document*



Nodes *XML Version* and *DOCTYPE* are common for any XML-based document and specify the XML version used in the document and information about the type of the document (the public identifier and the system identifier for SVG 1.0), respectively. Node *SVG tree* contains all the elements specific to SVG documents and is composed of four parts: *descriptive text*, *script*, *definitions*, and *body*. The *descriptive text* includes textual information not rendered as part of the graphic and is represented by two elements: *title*, usually appearing only once, and *desc*, appearing several times to describe the content of each SVG fragment. The *script* portion contains function definitions. Each function is associated with an action that can be executed on SVG objects in the document. Functions have a global scope across the entire document. The *definition* portion contains global patterns and templates of graphical elements or graphical properties that can be reused in the body of the SVG document. Each definition is characterized by a name, which is used in the body of the document to reference the definition, and by a set of properties. The graphical elements to be rendered are listed after the *<defs>* node, according to the order of rendering. Each element can belong to any of the basic SVG graphics elements, such as *path*, *text*, *rect*, *circle*, *ellipse*, *line*, *polyline*, *polygon*, and *image*, whose names are self-explanatory. The *body* of an SVG document contains any number of container and graphics elements. A container element can have graphical elements and other container elements as child elements. Container *g* is used for *grouping together* related graphics elements. A graphics element

Figure 6: Example of an SVG document



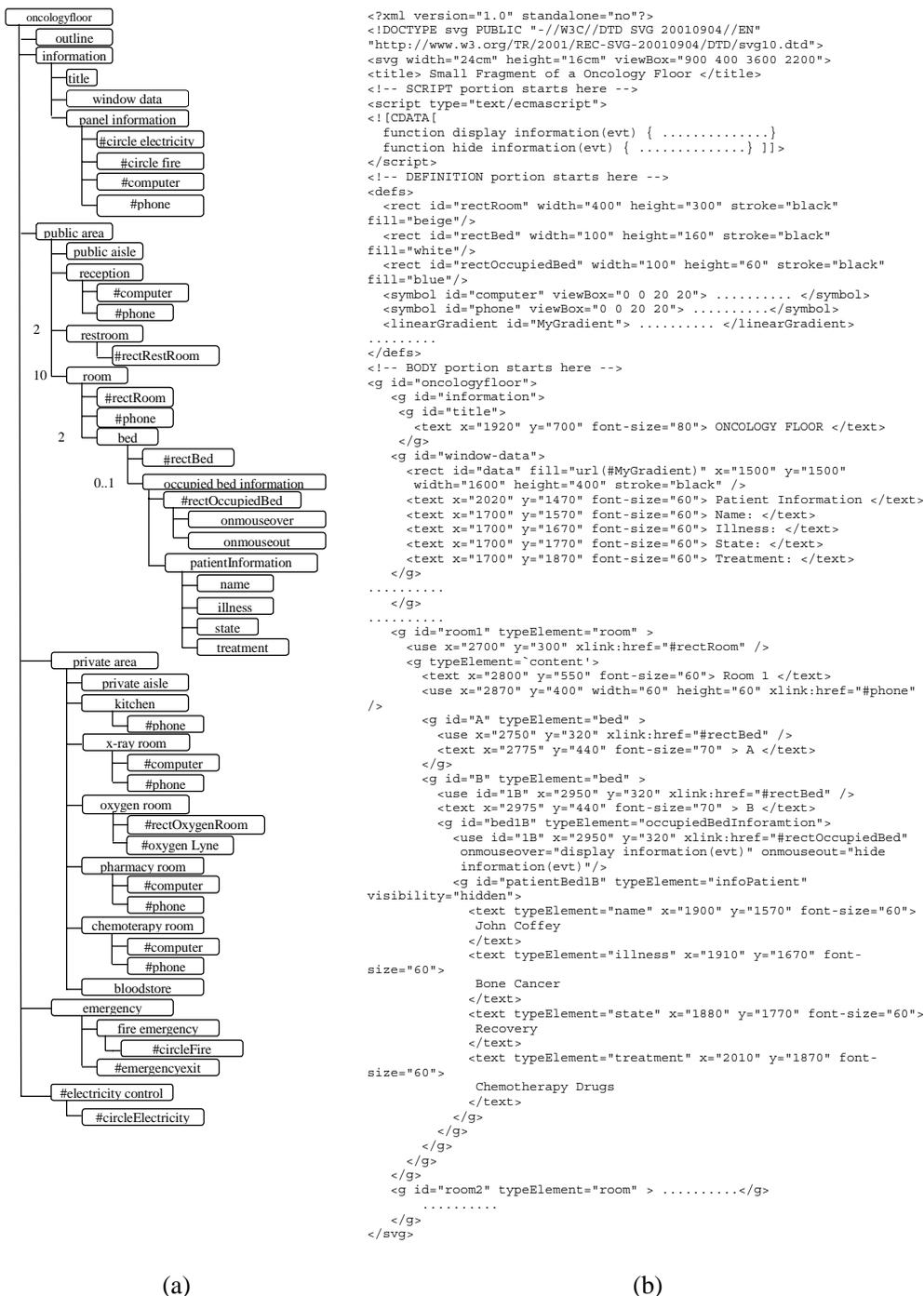
can cause graphics to be drawn. For instance, the *use* graphics element references another element (usually a definition) and indicates that the graphical contents of that element must be drawn at that specific point in the document. Each SVG element may have its own properties, modeled by XML attributes. All elements in the document can be uniquely identified including the special attribute *id='identifier'*. It is also possible to include user-defined properties, which can be useful for SVG data processing.

*Figure 6* illustrates the rendering of a sample SVG document, showing the oncology floor of a hospital. The document, integrated in a website, allows the hospital staff to know both details of the floor (e.g., rooms and equipments location) and recovered patient information. In particular, the rectangular appearing at the bottom with the text provides the information of the patient of bed *1B* on which the mouse is currently positioned (moving the mouse on other beds the corresponding patient will be returned).

*Figure 7(a)* shows a tree-based representation of the document rendered in *Figure 6*, reporting the types associated with the group elements composing its body. In particular, the body is a group element with *oncologyfloor* as identifier and with sub-elements of type *outline*, *information*, *public area*, *private area*, *emergency* and *electricity control* (the document defines one group for each of them). Group *public area* includes *public aisle*, *reception*, two *restroom* instances, and ten *room* instances. Each room, in turn, is composed of a graphical representation (*rectRoom* definition), a name, and two beds. Each bed is composed of a graphical representation (*rectBed* definition) and a bed name. Occupied beds further include a group with a new graphic element (*rectOccupiedBed* definition) and information on the occupying patient. The graphical representation of an occupied bed has two procedural attributes, namely *onmouseover='display information(evt)'* and *onmouseout='hide information(evt)'*, which show and hide respectively the patient information as the mouse pointer is positioned over the bed or moved out.

*Figure 7(b)* gives a portion of the SVG document rendered in *Figure 6* reporting its XML version, DOCTYPE, and part of its SVG tree with portions of its definitions, scripts, and body. The definition element (*<defs>*) includes the definition of several *abstract objects* (symbols, in the SVG terminology) like *computer* and *phone*, and different *auxiliary objects* like *rectRoom* and *rectBed*, which will be used as graphical interface for the objects of type 'room' and 'bed,' respectively. Element *script* includes the definition of functions *display information* and *hide information*, which are triggered by the *onmouseover* or *onmouseout* events to show and hide information on a patient occupying a given bed. The body includes the definition of all the groups composing it [as illustrated in *Figure 7(a)*]. The chunk reported in *Figure 7(b)* illustrates the definition of the *information* element and of room *room1*.

Figure 7: Tree-based graphical representation (a) of an SVG document (b)



## VoiceXML

VoiceXML (VoiceXML, 2002) is an XML format that has been designed for creating voice-user interfaces, particularly for the telephone. It uses speech recognition and DTMF (Dual Tone Multi Frequency) for input, and pre-recorded audio and text-to-speech synthesis (TTS) for output. A VoiceXML document is a tree whose root element *vxml* contains all the elements that compose the dialogue. VoiceXML supports two types of dialogs: (1) a *form* defines an interaction that collects values for each of the fields in the form, (2) a *menu* offers different alternatives of how to continue a dialog. A VoiceXML document represents a conversational finite automata that represents all the possible states in which the user can be in a particular moment.

The VoiceXML standard defines a very rich set of elements for managing all the voice related concepts, such as *audio*, *choice*, *form*, *grammar*, *menu*, *metadata*, *option*, *record*, *subdialog*, and so on. *Figure 8* shows a simple example of VoiceXML document where some metadata elements (*author* and *theme*) and a form are defined. This VoiceXML document represents a piece of a voice application that asks the user what kind of phone call she wants to do. Here, grammar *TelephoneCall.grxml* defines the allowable inputs for field TelephoneCall. This information, once collected, it is submitted to a web service (<http://www.TelephoneCall.example.com/tcall.asp>).

## SMIL

SMIL (SMIL, 2001) is an XML-based language for describing multimedia presentations. SMIL makes it possible to integrate and synchronize multimedia components such as audio, video, text, and images to form a multimedia audiovisual presentation. SMIL defines syntactic constructs for timing and

*Figure 8: An example of VoiceXML document*

```
<?xml version=" 1.0" encoding=" UTF-8"?>
<vxml xmlns=" http://www.w3.org/2001/vxml"
  xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=" http://www.w3.org/2001/vxml
    http://www.w3.org/TR/voicexml20/vxml.xsd"version=" 2.0">
  <meta name=" author" content=" John Coffey"/>
  <meta name=" theme" content=" VoiceXML straightforward example"/>
  <form>
    <field name=" TelephoneCall">
      <prompt> National or International call?</prompt>
      <grammar src=" TelephoneCall.grxml" type=" application/srgs+xml"/>
    </field>
    <block>
      <submit next="
        http://www.TelephoneCall.example.com/tcall.asp"/>
    </block>
  </form>
</vxml>
```

Figure 9: An example of SMIL document

```

<smil xmlns=" http://www.w3.org/2001/SMIL20/">
  <head>
    <layout>
      <root-layout width=" 400" height=" 300" background-color=" white"/>
      <region id=" r1" left=" 75" top=" 25" width=" 300" height=" 100" fit=" fill"/>
      <region id=" r2" left=" 150" top=" 100" width=" 150" height=" 100" />
      <region id=" r3" left=" 20" top=" 110" width=" 272" height=" 60" z-index=" 1"/>
    </layout>
  </head>
  <body>
    <par>
      <audio src=" bell.wav" type=" audio/wav" repeat=" 5" />
      <seq>
        <text src=" example.txt" type=" text/html" region=" r1" dur=" 10s"/>
        <animation src=" clock.swf" region=" r1" dur=" 3s" >
          <par>
            
            <video src=" changing.avi" type=" video/msvideo" region=" r3" />
          </par>
        </par>
        
      </seq>
    </par>
  </body>
</smil>

```

synchronization of media streams that allow fine-grained synchronization. The most important time containers provided by SMIL are the synchronization constructs *sec*, *excl*, and *par*. The *seq* element plays the child elements one after another in sequential order. The *excl* element plays one child at a time, but does not impose any order. The *par* element plays child elements as a group (allowing parallel playback). SMIL provides a rich set of multimedia constructs to specify multimedia properties, such as duration of an audio/video document, numbers of times that an element has to be played, dimensions of graphics, and many others.

SMIL improves bandwidth efficiency because it is possible to divide multimedia content into separate files and streams, send them to the user, and have them displayed together as if they were a single multimedia stream. The ability of separating text and images makes the multimedia content much smaller, thus reducing the download time.

Figure 9 shows a simple SMIL presentation that illustrates some sounds, video, text and images, synchronized with *seq* and *par* elements, and laid out in three overlapping regions.

## MPEG-7

MPEG-7 (MPEG-7, 2002) is an ISO/IEC standard formally named “Multimedia Content Description Interface”. It is a standard for describing the multimedia content data that supports some degree of interpretation of the information’s meaning, which can be passed onto, or accessed by, a device or a computer code. MPEG-7 provides a rich set of audiovisual descriptions that are based on catalogue (e.g., title, creator, date, rights), audiovisual semantic (e.g.,

who appears, what happens in a specific moment, where something happens, where information about object appears) and structural (e.g., the color histogram, the duration of acts) features of the audiovisual content. MPEG-7 uses XML schema as the language for content description, thus ensuring interoperability. Multimedia content description that is defined with MPEG-7 can be used to search, browse, and retrieve that content efficiently and effectively, but it does not standardize the extraction of audiovisual features.

The main elements of the MPEG-7 are:

- *Descriptors*. A descriptor (D) defines the syntax and the semantic of each feature (metadata element).
- *Description schemes*. A description scheme (DS) specifies the structure and semantics of the relationship between their components that may be both Ds and DSs.
- *Description Definition Language (DDL)*. It defines the syntax of the MPEG-7 descriptors and allows the creation, extension, and modification of DSs and Ds.

Figure 10 shows the relationship among the different MPEG-7 elements. The DDL allows the definition of MPEG-7 description tools, both D and DS, providing the means for structuring the Ds into DSs. The DDL also allows the

Figure 10: MPEG-7 main elements (MPEG-7, 2002)

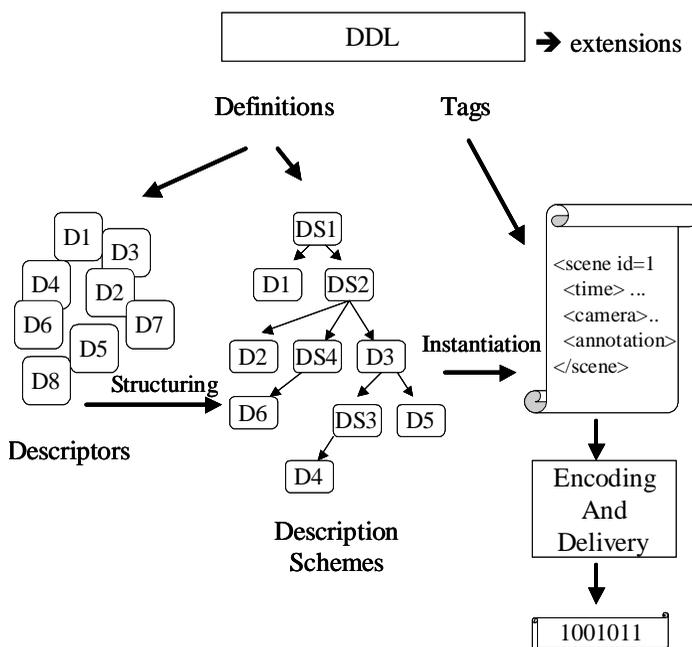
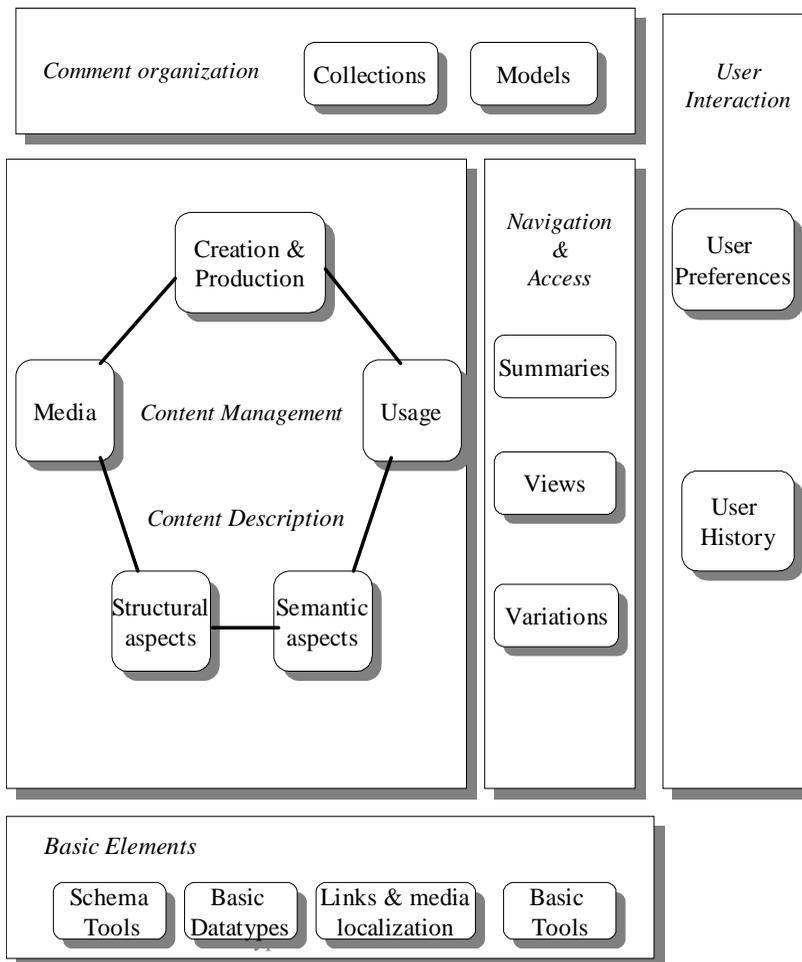


Figure 11: MPEG-7 general schema (MPEG-7, 2002)



extension of specific DSs for specific applications. Thanks to the DDL, the description tools are instantiated as descriptions in textual format (XML).

Figure 11 shows an overview of the organization of MPEG-7 Multimedia DSs into the following areas: *basic elements*, *content description*, *content management*, *content organization*, *navigation and access*, and *user interaction*. Basic elements include tools to manage annotation of MPEG-7 descriptions. The content management section provides tools to describe the document’s creation and production, media coding, storage and file formats, and content usage. The content description section provides tools to describe the structure of the audiovisual content in terms of video segments, frames, and audio segments. It also provides semantic tools to describe the objects, events, and

Figure 12: An example of MPEG-7 annotation

```

<VideoSegment id=" S1">
  <TextAnnotation>
    <FreeTextAnnotation> Ronaldo scores a goal.</FreeTextAnnotation>
  </TextAnnotation>
  <CreationMetaInformation>
    <Creation>
      <Creator> John, Coffey.</Creator>
    </Creation>
  </CreationMetaInformation>
  <MediaTime>
    <MediaTimePoint> 00:27:13;7</MediaTimePoint>
  </MediaTime>
  <MediaDuration> 00:00:25;3</MediaDuration>
</VideoSegment>

```

notions from the real world that are captured by the audiovisual content. The navigation and access section provide DSs for facilitating browsing and retrieval of audiovisual content by defining summaries, partitions and decompositions, and variations of the audiovisual material. The content organization section includes DSs for organizing and modeling collections of audiovisual content and descriptions. Finally, the user interaction section describes user preferences and usage history pertaining to the consumption of the multimedia material. Details about this organization can be found in MPEG-7 (2002).

MPEG-7 provides also a very rich standard to manage semantic metadata describing audiovisual information. It allows the analysis of an audiovisual document, and the manual or automatic creation of annotations containing the semantics metadata. As an example, *Figure 12* shows a portion of an MPEG-7 annotation document related to a football match. This document contains information about an event in the football match: a goal of Ronaldo, including the time where it happened and the duration. The complexity and variety of events and elements that can happen in an audiovisual document suggest a systematic annotation approach (Tsinaraki et al., 2003).

## MULTIMEDIA ACCESS CONTROL POLICIES

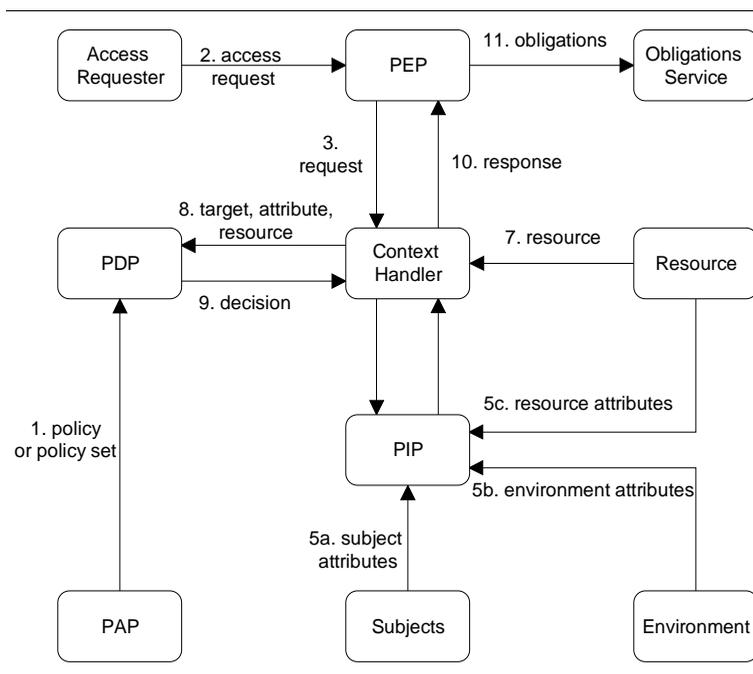
XML-based multimedia formats allow the definition of fined-grained access control policies that in turn can be expressed by an XML-based language. In this section, we present an overview of the eXtensible Access Control Markup Language (XACML) (XACML, 2003a), and illustrate a proposal of access control for SVG documents, one of the four multimedia languages described in the previous section.

## Extensible Access Control Markup Language

The eXtensible Access Control Markup Language (XACML) (XACML, 2003a) is an Oasis standard that provides a means for standardizing access control decisions for XML documents. XACML is used to define whether to permit requested access to a resource. XACML includes both an access control policy language and a request/response language. The policy language is used to specify access control policies. The request/response language expresses queries about whether a particular access should be allowed and describes answers to those queries that can be: *permit*, the request is allowed; *deny*, the request is denied; *indeterminate*, an error occurred or some required value was missing, so a decision cannot be made; and *not applicable*, the request cannot be answered by this service.

XACML defines three top-level policy elements: *Rule*, *Policy*, and *PolicySet*. The *Rule* element contains a boolean expression that can be evaluated in isolation. The *Policy* element contains a set of *Rule* elements and a specified procedure for combining the results of their evaluation. The *PolicySet* element contains a set of *Policy* or other *PolicySet* elements and a specified procedure for combining the results of their evaluation. This is the standard means for combining and reusing separate policies into a single combined one.

Figure 13: Dataflow diagram (XACML, 2003a)



*Figure 13* shows the main actors in the XACML domain and the information flow between them. The standard gives a definition of these concepts (actors, information flows, elements) that we summarize as follows:

- *PAP (Policy Administration Point)*. The system entity that defines the security policies.
- *PIP (Policy Information Point)*. The system entity that acts as a source of attribute values (of subjects, resources, actions, or environments).
- *PEP (Policy Enforcement Point)*. The system entity that performs access control.
- *PDP (Policy Decision Point)*. The system entity that evaluates applicable policies and renders an authorization decision.
- *Context Handler*. The system entity that converts decision request in the native request format to the XACML canonical form, and converts authorization decisions in the XACML canonical form to the native responded format.
- *Environment*. The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource, or action.
- *Target*. The set of decision requests, identified by definitions for resource, subject and action, that a rule, policy or policy set is intended to evaluate.

As *Figure 13* shows, in a typical XACML usage scenario, a subject wants to take some action on a particular resource. The PAP entity has previously specified security policies and makes them available to the PDP. The subject submits its query to the PEP. The PEP sends the request for access to the context handler in its native request format. The context handler constructs an XACML request based on the subjects, action, resources and other relevant information that are provided by the PIP. The PEP then sends this request to a PDP, which examines the request, retrieves policies (written in the XACML policy language) applicable to this request, and determines whether the access should be granted according to the XACML rules for evaluating policies. That answer (expressed in the XACML response language) is returned to the PEP, which can then allow or deny the access to the requester (Kay, 2003).

This access control language standard has many benefits (XACML, 2003b):

- It allows the unification of access control languages;
- Policies do not have to be rewritten in different languages;
- Developers do not have to invent new policy languages and write code to support them;
- It encourages reusability;
- Multi-application tools for managing and writing access control policies will be unified;

- It allows extensions to the access control language to accommodate other access control policies;
- It allows one policy to contain or refer to another.

Figures 14, 15 and 16 show an example extracted from XACML (2003a), and the XACML code corresponding to a policy, a request and a response, respectively. Suppose that there is a corporation named Medi Corp that defines a high level policy as follows: *Any user with an email name in the ‘medico.com’ namespace is allowed to perform any action on any resource.*

Figure 14 shows the policy that describes the high level policy of Medi Corp. It is composed of policy headers, an optional policy description, and the decision request to which this policy applies. If the subject, resource or action in a decision request do not match the values specified in the target, then the remainder of the policy does not need to be evaluated. The rule is then defined, including its effect (permit), an optional description and the specification of the

Figure 14: An example of an XACML policy

```
<?xml version=" 1.0" encoding=" UTF-8"?>
<Policy xmlns=" urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=" urn:oasis:names:tc:xacml:1.0:policy
  http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-policy-01.xsd"
  PolicyId=" identifier:example:SimplePolicy1"
  RuleCombiningAlgId=" identifier:rule-combining-algorithm:deny-overrides">
  <Description> Medi Corp access control policy </Description>
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources> <AnyResource/> </Resources>
    <Actions> <AnyAction/> </Actions>
  </Target>
  <Rule RuleId=" urn:oasis:names:tc:xacml:1.0:example:SimpleRule1"
    Effect=" Permit">
    <Description>
      Any subject with an e-mail name in the medico.com domain
      can perform any action on any resource.
    </Description>
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch
            MatchId=" urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
            <SubjectAttributeDesignator
              AttributeId=" urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType=" urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
            <AttributeValue
              DataType=" urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"> medico.com
            </AttributeValue>
            </SubjectMatch>
          </Subject>
        </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
  </Rule>
</xacml:Policy>
```

rule target that describes the decision requests to which this rule applies. In this case we can see the use of a function to identify users with an email in the ‘medico.com’ namespace.

As an example, suppose that John Coffey, with an email “jc@greenmile.com,” wants to read his medical record at Medi Corp. In XACML, information in the decision request is formatted into a request context statement such as that illustrated in *Figure 15*.

The statement has the request headers, and then the specification of the subject, resource and action, specifying for each attribute that identify the properties of the request. For the subject of our example an attribute “subject-id” is defined whose value is “jc@greenmile.com.” The PDP compares the policy target and the request information (subjects, resources and actions). If the policy target matches (i.e., there is a matching of all elements), then the policy matches this context. If the policy matches the request, the next step is to check the matching between the target rule and the request. In this case, the policy matches this context, but the rule does not, because there is not match between “\*@medico.com” and “jc@greenmile.com.” As a result, there is no rule in this policy that returns a positive answer and therefore response *NotApplicable* should be returned. *Figure 16* shows the corresponding XACML response.

XACML defines the syntax and semantic of a rich set of policies and context elements, also functional requirements for different entities and elements, various extensibility attributes, and so on. More details can be found in XACML (2003a).

*Figure 15: An example of an XACML request*

```

<?xml version=" 1.0" encoding=" UTF-8"?>
<Request xmlns=" urn:oasis:names:tc:xacml:1.0:context"
  xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=" urn:oasis:names:tc:xacml:1.0:context
  http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
  <Subject>
    <Attribute AttributeId=" urn:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType=" urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
      <AttributeValue> jc@greenmile.com</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId=" urn:oasis:names:tc:xacml:1.0:resource:ufs-path"
      DataType=" http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue> /medico/record/patient/JohnCoffey</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId=" urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType=" http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue> read</AttributeValue>
    </Attribute>
  </Action>
</Request>

```

*Figure 16: An example of an XACML response*

```

<?xml version=" 1.0" encoding=" UTF-8"?>
<Response xmlns=" urn:oasis:names:tc:xacml:1.0:context"
  xsi:schemaLocation=" urn:oasis:names:tc:xacml:1.0:context
  http://www.oasis-open.org/tc/xacml/1.0/cs-xacml-schema-context-01.xsd">
  <Result>
    <Decision> NotApplicable</Decision>
  </Result>
</Response>

```

## Access Control System for SVG Documents

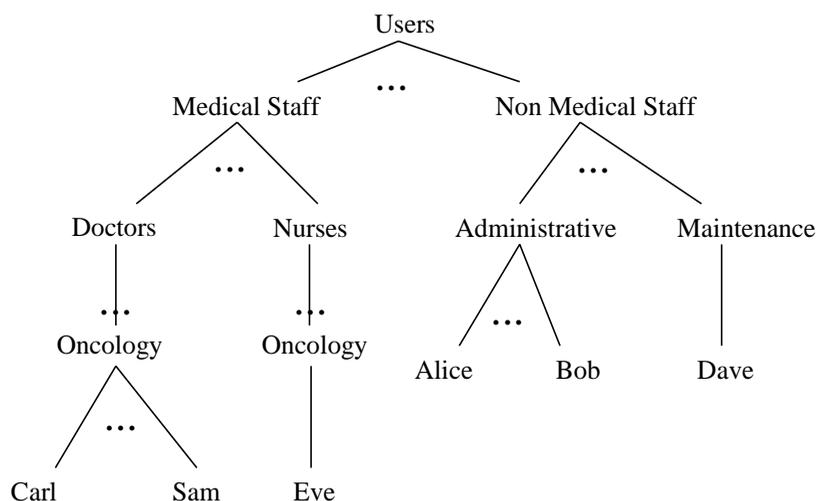
We now present an approach to fine-grained feature protection of Scalable Vector Graphics (SVG) data, one of the multimedia languages we have summarized in the previous section. This approach, proposed in Damiani et al. (2002a), is based on the use of authorization rules that are themselves expressed via an XML-based language. This approach exploits the peculiar characteristics of SVG documents and similar approaches could be defined for the other multimedia languages described in the previous sections [see, for example, the access control model and encryption mechanism for SMIL documents described in Kodali & Wijesekera (2002)].

An authorization rule states what actions performed on multimedia data are to be allowed (or denied). Basically, each authorization rule has four components: the *subject* to which the rule applies, the *object* to which the authorization refers, the *action* to which the rule refers, and the *sign* describing whether the rule states a permission (sign = '+') or a denial (sign = '-') for the access. Here, for the sake of simplicity, we assume action to be the request to render the document (intuitively the *read* operation). This is not limiting, as reference to specific actions defined on the document (e.g., *rotate*) can be regulated by allowing (or not allowing) access to the corresponding element. We are now ready to describe subjects and objects of these rules.

## Authorization Subjects

The specification of subjects in access control rules has often two apparently contrasting requirements (Samarati & De Capitani di Vimercati, 2001). On the one side, subject reference must be simple, to allow for efficient access control and for exploiting possible relationships between subjects in resolving conflicts between the authorizations (e.g., most specific relationships between groups and sub-groups). On the other side, one would like to see more expressiveness than the simple reference to user identities and groups, providing support of profile-dependent authorizations whose validity depend on properties associated with users (e.g., age, citizenship, or field-of-specialization) (Bonatti

Figure 17: An example of user/group hierarchy



et al., 2001a; Bonatti et al., 2002b). This approach encounters both requirements by supporting both user groups and user profiles.

As usual, *groups* are sets of users hierarchically organized: groups can be nested and need not be disjoint (Jajodia et al., 2001). *Figure 17* reports an example of a user-group hierarchy. In addition, for each user, a profile may be maintained specifying the values of properties such as name, address, and nationality that the model can exploit to characterize them. The profile is modeled as a semi-structured document and can then be referenced by means of XPath expressions (XPath, 2001). *Figure 18* illustrates three XML documents defining the profiles for three users (all documents will be instances of an XML schema where all the used properties have been defined as optional). A *path expression* is a sequence of element names or predefined functions separated by character / (slash):  $l_1/l_2/ \dots /l_n$ , and is used to identify the elements and attributes within a document. For instance, XPath expression `user profile//./citizenship[@value = 'EU'] AND [./job[@value='doctor']]` returns element user profile of profiles of EU citizens who work as doctors. In particular, such an expression, evaluated on the profiles in *Figure 18* would return the profile of user Sam.

Therefore, the subject component of this authorization rules includes two parts:

- an *identity*, whose value can be a user or a group identifier;
- a *subject expression* which is an XPath expression on users' profiles.

Figure 18: An example of XML user profiles

<pre> &lt;user_profile id='Carl'&gt;   &lt;name value='Carl' /&gt;   &lt;address value='California Ave.' /&gt;   &lt;citizenship value='US' /&gt;   &lt;job value='doctor' /&gt;   &lt;specialization value='oncology' /&gt; &lt;/user_profile&gt; </pre>	<pre> &lt;user_profile id='Dave'&gt;   &lt;name value='Dave' /&gt;   &lt;address value='Forest Ave.' /&gt;   &lt;job value='maintenance worker' /&gt;   &lt;level value='senior' /&gt;   &lt;building value='ADM125' /&gt;   &lt;office value='33' /&gt; &lt;/user_profile&gt; </pre>	<pre> &lt;user_profile id='Sam'&gt;   &lt;name value='Sam' /&gt;   &lt;address value='Manchester Rd' /&gt;   &lt;citizenship value='EU' /&gt;   &lt;job value='doctor' /&gt;   &lt;specialization value='oncology' /&gt; &lt;/user_profile&gt; </pre>
---	---	---

Intuitively, the authorization rule should apply only if the profile of the requestor satisfies the constraints expressed in the Xpath expression.

Authorization subjects are then defined as XML-elements of the form:

```

<subject>
  <id value='user/group-id' />
  <subj-expr>xpath-expr</subj-expr>
</subject>

```

For instance, subject element:

```

<subject>
  <id value='MedicalStaff' />
  <subj-expr>user profile//[./citizenship[@value='EU']]</subj-expr>
</subject>

```

denotes European users belonging to group *MedicalStaff*.

Again, with reference to the group hierarchy in *Figure 17* and the profiles in *Figure 18*, the subject expression will evaluate to true for *Sam* and therefore an authorization rule using this subject will be considered applicable to him. By contrast, the authorization rule will not be applicable to *Dave* (who does not belong to *MedicalStaff*), or to *Carl* (who does not satisfy the constraint on citizenship).

## Authorization Objects

According to the description in the section on “Scalable Vector Graphics,” three kinds of protection objects can be identified: *definitions (defs)*, *groups*

(*g*), and *SVG elements*. SVG elements can be graphical or textual, such as *rect* or *circle*, or can reference the definitions [e.g., element use in *Figure 7(b)*]. The authorization model supports fine-graining by allowing the association of authorizations with any of such specific elements within an SVG document. As SVG is XML-based, generic XPath expressions on the SVG document can be used to specify the elements to which an authorization applies (Damiani et al., 2002b). For instance, with respect to the document in *Figure 7(a)*, the path expression *defs/rect[position() = 1]* identifies the first *rect* child of the *defs* element (corresponding to the definition of the perimeter of a room). Such an expression can then be used in an authorization rule to grant or deny access to that specific element. Although generic XPath expressions are sufficient to provide fine-grained authorization specification, their only support results limiting from the point of view of the authorization administration. While verbose, these path expressions refer to the syntax and are detached from the semantics of the elements in the SVG document. As a result, the translation of high-level protection requirements into corresponding path expressions on the document is far from being trivial. It is therefore important to provide a higher-level support for the specification of authorization objects. Providing high-level support for the definition of authorization objects translates into solving two problems:

- *object identification*: how to identify the portion (element) of the SVG document to which an authorization refers;
- *condition support*: how to specify conditions that the identified element(s) have to satisfy.

### *Object Identification*

To provide an expressive authorization language, the tree format of SVG documents is exploited by assuming that *semantics aware tags* and good design techniques can be defined and exploited. In particular, as illustrated in the section on “Scalable Vector Graphics,” each SVG element can have an identifier (attribute *id*). Identifiers provide useful as they permit explicit reference to specific elements (e.g., *room1*) based on their name. However, identifiers are not sufficient as listing explicit elements may, in some situation, result inconvenient (e.g., to protect all rooms of the floor we will have to specify one authorization for each room identifier). Also, support for distinguishing the shape of an object from its content seems to be needed (e.g., to support cases where a user can see the existence of a room — and then its shape — but cannot see what is inside the room). These two requirements are addressed in Damiani et al. (2002a) as follows. First, in addition to the identifier, each element can have an attribute *typeElement* that defines the *conceptual type* of the element (e.g., room, bed, telephone, computer). The type element can be exploited to reference all objects of a given type (e.g., all rooms) in a single expression. Second, if the

*shape* of an element is conceptually meaningful, the model assumes a good design of the element where the shape (i.e., the *drawing instructions*) appears at the first level and the content appear in a nested element group. The predefined function *perimeter()* identifies the shape (i.e., the drawing instructions) of an element (referenced via its identifier or type).

Summarizing, an object can then be referenced via any of the following:

- a *path expression* resolving in the object;
- its *object identifier* (value of its attribute *id*);
- its *type* (value of its attribute *typeElement*);
- the application of function *perimeter* to any of the above.

To distinguish which of the above means is used in the specification of an authorization object, a dot notation is used prefixing the object with either “*id.*,” “*type.*,” or “*path.*” For instance, value “*type.room*” indicates objects with *typeElement* equal to *room*, while value “*id.room1*” denotes the room with *id*’s value *room1*. Analogously, *perimeter(type.room)* identifies the perimeter of all the rooms, and *perimeter(id.room1)* identifies the perimeter of room *room1*.

### Condition Support

To provide a way for referencing all elements satisfying specific semantically rich conditions, the model allows the specification of *object conditions* that identify a set of objects satisfying specific properties. For instance, we may need to define an access rule stating that “a doctor can see computers only if they are in the same room as (i.e., *together with*) diagnostic machines.” In this model, conditions are boolean expressions that can make use of the following *predicates*:

- *inside(obj)*. It returns the object in the authorization rule if it is inside an element whose identifier, type, or name is *obj*.
- *together\_with(obj)*. It returns the object in the authorization rule if it is a child of an element together with an object whose identifier, type, or name is *obj*.
- *number\_of(obj, n)*. It returns the object in the authorization rule if there are *n* instances of the object whose identifier, type, or name is *obj*.

Authorization objects in the model are then defined as:

```
<object>
  <refer value='object-id' />
  <cond> pred-expr </cond>
</object>
```

where element refer provides the object identification and element cond specifies additional conditions.

Some examples of object expressions are as follows:

- `<refer value='type.phone' /> <cond>together with(type.computer)</cond>`

denotes all “phones” that are in the same room as (together with) a “computer.”

With respect to the example in *Figure 6*, it denotes the phones in the “Pharmacy,” “X-Rays,” “Chemotherapy,” “Kitchen,” and “Reception: rooms.

- `<refer value='perimeter(type.room)' /><cond>inside(type.oncologyfloor)</cond>`

denotes all the graphical elements (the *use* elements referencing the *rectRoom* definition) that draw the perimeter of the rooms of the oncology floor.

Summarizing the authorization rules [whose schema can be analyzed in Damiani et al. (2002a)], allow the specification that specific users or groups thereof satisfying specific properties can (if sign=‘+’) or cannot (if sign=‘-’) access given objects (referred by means of their, identity, type, or path expressions as well as of conditions that they satisfy). With this expressive way of referring to subjects and objects, it is worth noticing how the combined use of positive and negative authorizations results convenient for the specification of different constraints, providing an additive or subtractive way of defining authorized views. In particular, one could start from the empty map and add positive authorizations specifying the objects that may be released, or specifying a global positive authorization and further (more specific) negative authorizations for the objects that cannot be released.

Of course, the two approaches can be combined as they best fit the application.

## Example of Authorization Rules

In this subsection we show some examples of protection requirements and corresponding authorizations to regulate access to the graphic introduced in the section on “Scalable Vector Graphics.” The user groups used and the properties used in the authorizations refer to the user group hierarchy in *Figure 17* and the users’ profiles in *Figure 18*.

**Rule 1** Everybody can see the emergency exits  
`<subject><id value='Users' /></subject>`  
`<object><refer ='type.emergencyexit' /></object>`  
`<sign value='+' />`

**Rule 2** Everybody can see the content of any room in the public area

```
<subject><id value='Users'/></subject>
<object><refer='id.PublicArea'/></object>
<sign value='+'/>
```

**Rule 3** Everybody can see the perimeter of any room in the private area

```
<subject><id value='Users'/></subject>
<object>
<refer='Perimeter(g[@id='oncologyfloor']//g)/>
  <cond>inside(id.PrivateArea)</cond>
</object>
<sign value='+'/>
```

**Rule 4** Only members of the NonMedicalStaff whose job is “maintenance worker” can see the fire emergency and electricity controls

```
<subject> <id value='NonMedicalStaff'/><cond>job[@value='maintenance
worker']</cond>
</subject>
<object><refer='type.electricitycontrol'/></object>
<sign value='+'/>
<subject>
  <id value='NonMedicalStaff'/><cond>job[@value='maintenance
worker']</cond>
</subject>
<object><refer='type.fire emergency'/></object>
<sign value='+'/>
```

**Rule 5** Medical staff can see the content of any room in the private area

```
<subject><id value='MedicalStaff'/></subject>
<object><refer='id.PrivateArea'/></object>
<sign value='+'/>
```

**Rule 6** Doctors with specialty “oncology” can read patient information; everybody else is explicitly forbidden

```
<subject>
  <id value='Doctors'/><cond>specialty[@value='oncology']</cond>
</subject>
<object><refer='type.patientinformation'/></object>
<sign value='+'/>
<subject><id value='Users'/>
</subject>
<object><refer='type.patientinformation'/></object>
<sign value='-'/>
```

Figure 19: An example of two views on the SVG document in Figure 6

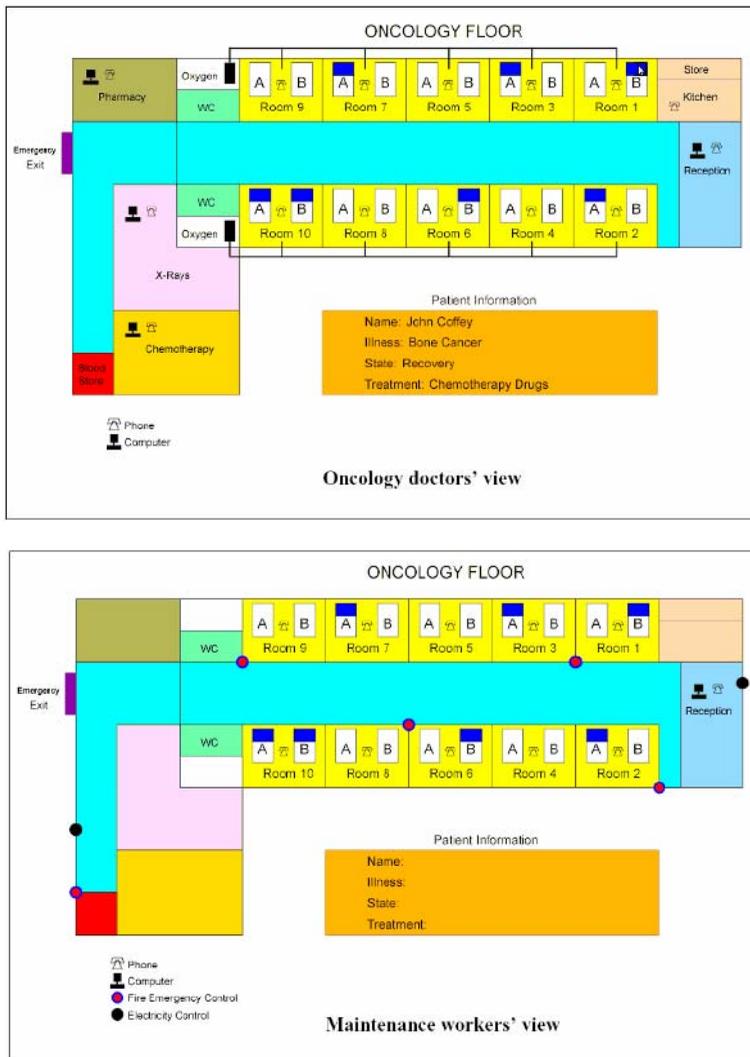


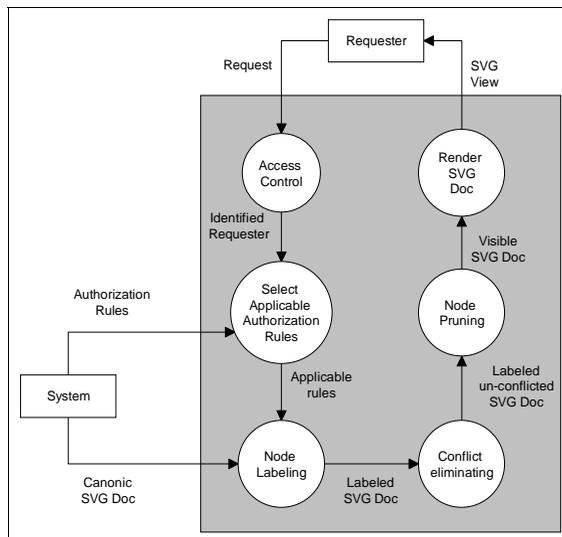
Figure 19 illustrates the views that will be returned to oncology doctors and maintenance workers respectively based on the specified authorizations. Notice in particular that the doctors' view does not include fire emergency and electricity control information (as the authorizations in Rule 4 are not applicable to them). The maintenance workers' view, while containing fire emergency and electricity control information is clearly missing many details, such as hospital machines and patient details, whose access is restricted to physicians. The next section illustrates the process for obtaining such views.

## Policy Enforcement

Figure 20 shows the dataflow diagram that represents the main steps of an algorithm to enforce SVG access control policies. This schema, based on the proposal in Damiani et al. (2002a) would be similar for any kind of multimedia documents that have been summarized in the second section. The enforcement algorithm consists of two main phases: *node labeling* and *tree transformation*. Node labeling takes as input a user request and the DOM tree of the SVG document. Then, it evaluates the authorization rules to be enforced and selectively assigns a plus or minus sign to the nodes of the DOM tree. Subsequently, the tree transformation phase takes the labeled DOM tree as input and transforms it into another valid DOM tree. The result is a view of the SVG document containing only the elements that the requestor is entitled to access. The following are the main steps of the algorithm.

1. Determine the set *Applicable authorizations* of authorizations applicable to the requestor. These are all the authorizations for which the requestor is a member (possibly proper) of the subject identity (*id*) and for which requestor's profile satisfies the subject expression ( $\langle subj\text{-}expr \rangle$ ).
2. Evaluate the object expressions in every authorization in *Applicable authorizations* (e.g., resolving them into suitable XPath queries), and label the corresponding SVG elements with the authorization subject identity (*id*) and the sign of the authorization.

Figure 20: Enforcement algorithm



3. If an element has more than one label, eliminate all labels whose subject identity is a super-group of the subject identity in another label (most-specific take precedence).
4. If an element remains with more than one label, then: (i) if all labels are of the same sign assume that sign for the element; (ii) if labels are of different sign assume '-' (denials take precedence on remaining conflicts).
5. Starting from the root, propagate each label on the DOM tree as follows:
  - (a) one step upward to the father node, provided the father node is a <g> element while the current node is not.
  - (b) downward to descendants. Unlabeled descendants take the label being propagated and propagate it to their children, while unlabeled ones discard the label being propagated and propagate their own to their children (most specific take precedence).
6. Discard from the document all subtrees rooted at a node with a negative label.
7. Discard from the document all subtrees whose nodes are all unlabeled nodes.
8. Render the resulting document.

While steps 1-3 solve the problem of determining labels to be given to nodes according to applicable authorizations, step 4 of the enforcement algorithm deals with completing the labeling phase by propagating initial labels on the SVG document's DOM tree; such a step is specific to the SVG data model and deserves some comments.

## Multimedia DRM

Traditionally, rights management (RM) of physical products exploited the physical nature of goods (e.g., books, records, pictures) as a barrier against unauthorized exploitation of content. Nowadays, the increasing success of digital products such as MP3 files and DVD movies has fostered breaches of copyright law because of the relative ease with which digital files can be copied and transmitted. For this reason, Digital Rights Management (DRM) is an important research topic and many commercial solutions aimed at protecting digital content are being proposed. Initially, DRM research focused on encryption as a means of tackling unauthorized copying of digital content. Piracy prevention was the main motivation underlying first-generation DRM techniques. More recently, second-generation approaches to DRM address a much wider area, including the management of the description, identification, commercialization, protection, monitoring of online (web, digital tv, digital radio, 3rd mobile phone generation, etc.) and off-line (CD, DVD, etc.) multimedia documents relating to the rights and usage.

It also affects the relations with the holders of those rights. In this wider perspective, Digital Rights Management (DRM) has been defined as “the science, art and business of controlling digital goods so that all of the participants in the digital goods chain win” (DRM, 2003).

Such claim may be substantiated as follows:

- Consumers win by getting a good, perhaps novel, product or service at a reasonable price.
- Distribution and infrastructure providers win by getting paid to facilitate the distribution of goods, and perhaps by additional related interactions with their customers.
- Content owners win by getting fairly paid for their efforts, and by having new, innovative distribution channels available to them.

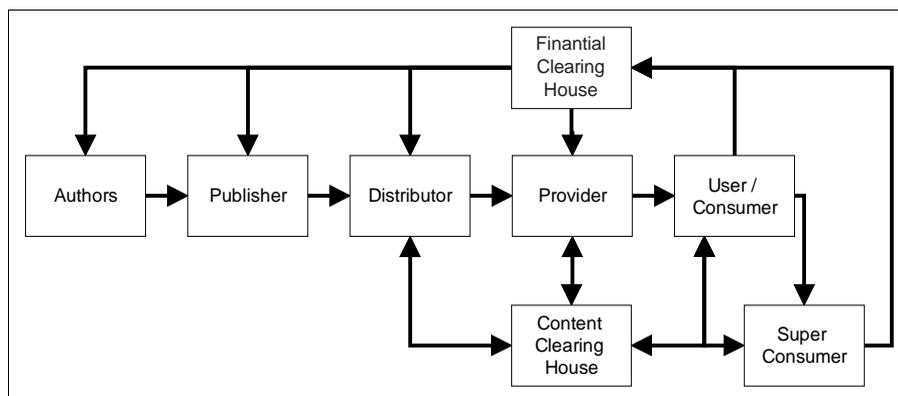
Nevertheless, DRM has also a high cost (of different nature) for all these participants. Consumers are the least interested in DRM because it will make more difficult to obtain illegally and free multimedia products; distribution and infrastructure providers will have to integrate mechanisms to guarantee that DRM requirements are fulfilled in the distribution and usage process; content owners will have to integrate mechanisms to define and manage rights in the content.

The basic idea underlying recent approaches to DRM is defining an infrastructure that allows specifying the rights on the multimedia documents (usage conditions, access control restrictions, etc.), and providing enforcement mechanisms that ensure the fulfilling of these rights. *Figure 21* shows all the roles that are involved with DRM. We can appreciate three types of flows related to the content, financial process and rights management. The roles that are involved with the content are as follows: *author* (creates the content), *publisher* (packages the content), *distributor* (retains protected the content), *provider* (sells the content), and finally *user* (sees, plays, listens, etc.). *Financial clearinghouse* is in charge of managing the payments and billing, and the *content clearing house* authorizes the adequate content usage to the user.

Implementing such a DRM-aware infrastructure can prove a complex task because of the impact of the global network infrastructure. Barriers to publishing are disappearing (there are billions of pages on the World Wide Web); redistribution of content has become extremely easy; concepts of territoriality are meaningless on the network (but still very significant for business in the real world); in a physical world, access created a point of scarcity in the value chain, while in the digital world, access is no longer scarce; stealing normal materialized goods takes away the possession of the thing to someone else, but copying digital content just produces another instance.

Moreover, new business imperatives increase the desirability of having new agile rights management functionalities in enterprise content systems (Rosenblatt

Figure 21: DRM roles



& Dykstra, 2003). This has an impact over the entire life-cycle of media information:

- *Control access during workflow:* Controlling allowed uses of digital content is a critical function of DRM technology. By predetermining and controlling the exact uses and conditions for content, the DRM technology extends and enhances the traditional access control techniques.
- *Outsourcing:* The outsourcing of content production processes increases considerably the requirements for control of authority and authentication. This is an important problem since outsourcing is growing fast.
- *Downstream use:* Companies need to deliver controlled access downstream so that content can be licensed, deployed and repurposed by business partners in accordance with the terms or agreements.
- *Protection throughout content life-cycles:* Piracy costs billions of dollars each year. But business models rely on an assurance that copyrights, and usage rights, are protected and extended beyond content production and distribution systems.
- *Modification of rights over time:* Digital content is dynamic, since it can be transformed, reused, repurposed and renegotiated. Companies look for ways to mold their content as business needs dictate, and rights, licenses, and relationships allow.
- *Regulatory and business standards:* Integrity, authentication, security, privacy and accountability need new legislative and regulatory standards that integrate to fulfill a shared goal.

DRM has many benefits. In particular, it improves the content rights management, maximizing the investment and reducing costs. Also, it makes it

possible to define realistic budgets, and improve the control of sells. DRM has an added value, because it enables the developing of some business models that would have many problems without it: paid downloads such as movies, music trials, statistics, etc. need analog efficiency as physical media commerce; subscriptions such as journals, cable TV, websites, databases and repositories, music, etc. need an adequate rights management; pay-per-view/listen for television, music, etc.; usage metering; peer-to-peer/superdistribution; rights licensing; etc.

In the next section we analyze the DRM reference architecture and the DRM standards, paying special attention to the eXtensible rights Markup Language (XrML).

## DRM Architecture

An overview of a generalized, comprehensive DRM system in which users are granted specific *use rights* to information under the originator's control is shown in *Figure 22*. This model assumes the availability of standardized or proprietary infrastructures for identification, metadata, authentication and cryptography. The process flow depicted in *Figure 22* can be outlined as follows (Erickson, 2002a):

1. The user obtains content packages: The user might receive content through different channels.
2. The user attempts to use the content: The DRM Controller determines that the requested use requires authorization.
3. DRM Client makes Rights Request: If the license package containing the necessary authorization credentials is not directly available, attributes of the user's request, including the usage context, are sent to a License Server.
4. The License Server verifies the submitted client identification against an identity database.
5. The License Server looks up the rights specifications for this content item.
6. A financial transaction is executed, if none has been recorded and the rules require it.
7. The contents of the license package are assembled: the rights specification, identifiers, revocation information, cryptographic keys to the content — all specific to the content and context of use.
8. The license is securely packaged and transferred to DRM Client.
9. The DRM Client uses the license to open the content for the particular requested.
10. The content is rendered, viewed, or listened as requested.

This architecture requires standard interfaces between three architectural levels of abstraction: *rights expression languages*, *rights messaging protocols* and mechanisms for *policy enforcement and compliance*.

Rights expression languages provide the basis for expressing rights information and usage control policies. In this sense, XrML (Erickson, 2002b) can be seen as a starting point trying to cover all these points.

Right messaging protocols could provide the means for inquiring about and disseminating digital rights information and policies. Mechanisms such as SAML (SAML, 2003) and XACML (see the section on “Extensible Access Control Markup Language”), transported over RPC-like mechanisms like SOAP (SOAP, 2000) provide a strong, standard foundation upon which build a framework of interoperating rights management services.

As far as policy enforcement and compliance mechanism are concerned, they should provide a set of open Application Program Interfaces (APIs) that enable a competitive market in the provision of enforcement methods.

## DRM Standards

The market of DRM solutions has been slow to grow, and commercial products providing DRM functionalities are not yet widespread. There are several reasons for this, such as (DRM, 2003):

Figure 22: DRM Architecture (Rosenblatt, 2002)

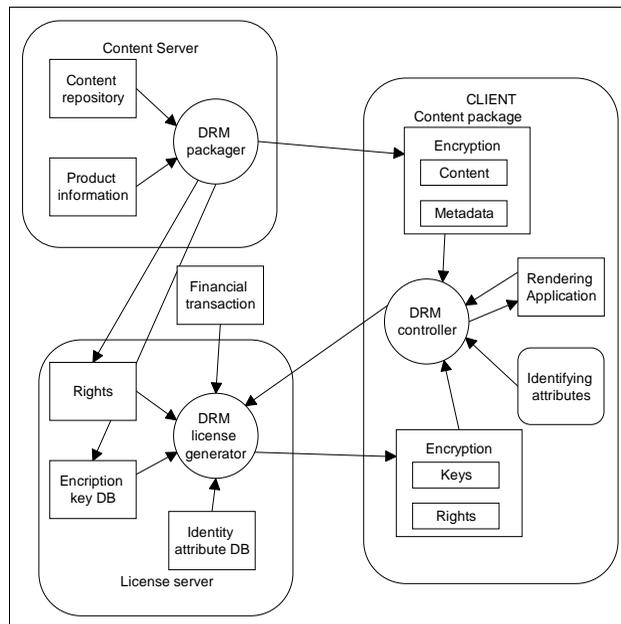


Figure 23: Content standards hierarchy (Rosenblatt, 2002)

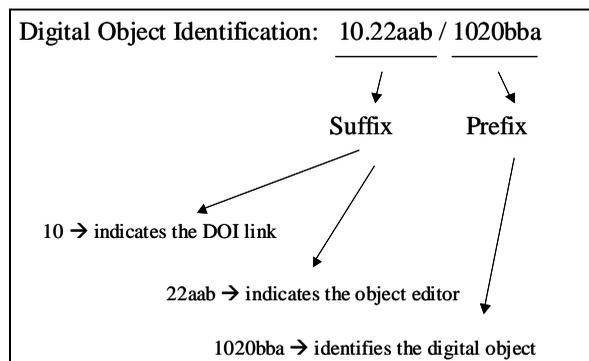
Individual Publishers	Content Management	<indec2>RDD Rights/holder Management		Business Models
	Content Industries IP Identification	DOI <i>XrML, ODRL, ICE</i>	<i>ONIX, PRISM, LOM, NewsML</i> Products Metadata	<i>PDF, Flash, Real, WMP</i> Formats & Players
E-Commerce	Payment Schemes	<i>Passport, Liberty Alliance</i> Authentication		<i>RSA, BlowFish, AES, RC5, etc.</i> Encryption
	<b>HTTP, HTML, XML</b> <b>The Internet</b>			

- Users do not like going through registration or authentication procedures, which many DRM solutions require.
- Publishers have not built the internal systems necessary to manage content, which is required infrastructure for content distribution.
- Current DRM systems are not interoperable with each other.

Some or these limitations, especially the latter, result from a lack of technology standards for DRM. In fact, the management of digital rights depends on the ability to communicate unambiguously, and this is difficult without standards (even with standards). Standards would help foster interoperability and competition application from software vendors. The standard organizations are making an important effort to develop standards that allow the integration of all DRM technology components. Figure 23 shows a general hierarchy of the standards that are related to the Internet. There are four categories of content standards: identification, rights, metadata and formats. In the following subsection we analyze XrML, one of the most important content standards.

Apart from XrML, another crucial standard is the Digital Object Identifier (DOI) (DOI, 2003). DOI, like ISBN codes used in the publishing industry, is a scheme for uniquely identifying digital content. A central agency is in charge of registering unique identifications for digital objects. Each digital object's identi-

Figure 24: An example of Digital Object Identification



ification is an alphanumeric string composed of different elements that ensure uniqueness (see *Figure 24*). This standard is having a huge success, and both publishers and DRM application developers are using the DOI in growing numbers.

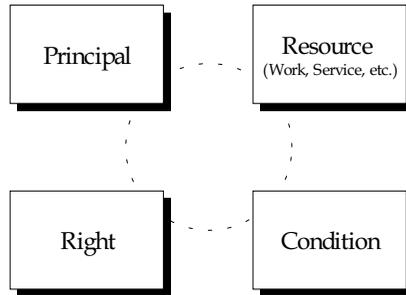
## Extensible Rights Markup Language

XrML (Erickson, 2002b) is an XML-based language to specify rights and conditions to control the access to digital content and services. Using XrML, anyone owning or distributing *digital resources* (such as content, services, or software applications) can identify the *parties* allowed to use those resources, the *rights* available to those parties, and the terms and *conditions* under which those rights may be exercised. These four elements are the core of the language and establish the full context of the rights that are specified.

The XrML basic model (*Figure 25*) is composed of four entities and the relationship between them. These entities are as follows:

- The *principal*. A principal encapsulates the identification of a party to whom rights are granted. Each principal identifies exactly one party. A principal specifies the party by using a unique piece of information. Usefully, this information has some associated authentication mechanism by which the principal can prove its identity.
- The *right*. A right is the “verb” that a principal can be granted to exercise against some resource under some condition. Typically, a right specifies an action (or activity) or a class of actions that a principal may perform on the associated resource.
- The *resource*. A resource is the “object” on which a principal can be granted a right. A resource can be a digital work (such as an e-book, an audio or video file, or an image), a service (such as an email service, a B2B

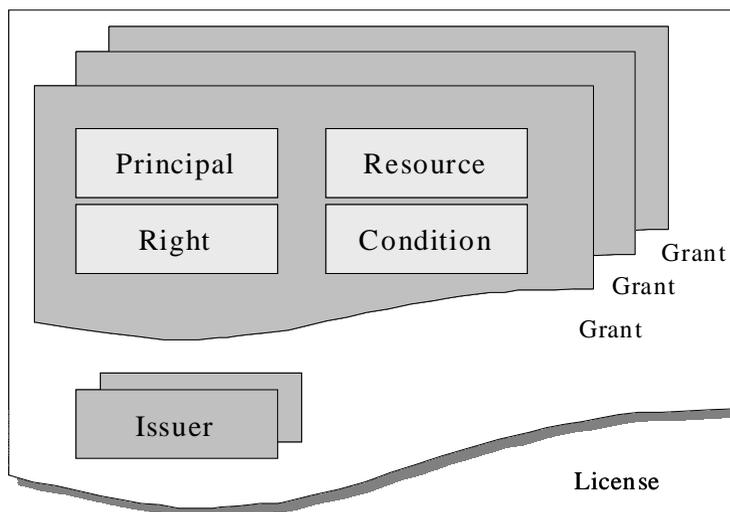
Figure 25: XrML data model



transaction service), or even a piece of information that can be owned by a principal (such as a name or an email address).

- The *condition*. A condition specifies the terms, conditions, and obligations under which rights can be exercised. A simple condition, for instance, is a time interval within which a right can be exercised. A slightly more complicated condition consists in requiring users to hold a valid prerequisite issued by some trusted entity. Combining conditions via logical connectives, the expressive power of digital rights statements is greatly increased since the eligibility to exercise a specific usage right can depend in the eligibility to exercise other rights. Moreover, a list of conditions can be put in conjunction to form a condition requiring that the conditions all be met simultaneously.

Figure 26: XrML license



Conceptually, the basic building block of an XrML DRM license is the assertion “grant.” To state such an assertion meaningfully other parameters must be specified, such as the identification of the principal who issued the grant and the identification of the license. For this reason, in a system environment, the central XrML construct is a coarser “license” object (see *Figure 26*). A license is composed of three elements: a set of grants that convey to certain principals certain rights to certain resources under certain conditions; an identification of the principal or principals who issued the license and thus bestow the grants upon their recipients; and additional information such as a description of the license and validity date.

Readers already familiar with digital certificates and other similar structures might notice the absence of the identification of the principal or principals to whom certain rights are conveyed.

For instance, the code shown in *Figure 27* represents an unlimited rights grant of printing and viewing an e-book to Alice for an up-front payment of USD \$15.

The policy is largely self-explanatory; namely, the `<KeyHolder>` tag contains the encryption parameters for secure communication of the access rights, while the `<fee>` (belonging to the service namespace, indicated by the prefix `sx`: in *Figure 27*) specifies the payment type (flat rate) and gives the unique UDDI identifier of the service being paid for. Then, within `<GrantGroup>`, each `<grant>` tag includes an action the principal is allowed to perform (in our case, the `<play>` and `<print>` actions belonging to the namespace indicated by the prefix `cx`: in *Figure 27*).

## CONCLUDING REMARKS

The intellectual property framework has long been regarded as essential for the “progress of science and useful arts.” With the growth of the network, it is clear that law alone will not be sufficient to protect the interests of creators. If technology is to be used to support the law, an open standardized infrastructure for the Management of Digital Rights will be essential for unambiguous communication. The Digital Management of Rights is still at an early stage of development; its implementation should focus much more on facilitating access than on prevention, trying to minimize those aspects that are deeply unattractive from the point of view of personal privacy and rights. On the other hand, failure to protect intellectual property could jeopardize content industries, depriving the general public of valuable artistic and cultural products. Copyright protection mechanisms, whether implemented through law or technology, will not gain public acceptance without appropriately balancing content protection and access facilitation. Depending on the medium and the marketplace, consumers may reasonably expect to choose freely the use they intend to make of digital content.

Figure 27: A XrML example

```

<keyHolder>
  <info>
    <dsig:KeyValue>
      <dsig:RSAKeyValue>
        <dsig:Modulus> Efgao6NYfm...</dsig:Modulus>
        <dsig:Exponent> AQAQAA==</dsig:Exponent>
      </dsig:RSAKeyValue>
    </dsig:KeyValue>
  </info>
</keyHolder>
<!-- Consumer must pay a one time fee of $15.00 for rights -->
<sx:fee>
<sx:paymentFlat>
  <sx:rate currency="USD"> 15.00</sx:rate>
  <sx:paymentRecord>
    <sx:stateReference>
      <uddi>
        <serviceKey>
          <uuid> D04951E4-332C-4693-B7DB-D3D1D1C20844</uuid>
        </serviceKey>
      </uddi>
    </sx:stateReference>
  </sx:paymentRecord>
</sx:paymentFlat>
</sx:fee>
<grant>
  <!-- The right to play/view is granted -->
  <cx:play/>
  <!-- the book -->
  <digitalResource licensePartIdRef="eBook">
</grant>
<grant>
  <!-- The right to print is granted -->
  <cx:print/>
  <!-- the book -->
  <digitalResource licensePartIdRef="eBook">
</grant>
</grantGroup>
</license>

```

Discouraging piracy while providing for all reasonable uses of content is an important objective for DRM technologies. In this context, XML-based standards for DRM policies have many advantages: it is possible to distribute and manage DRM policies together with content; it is quickly the parsing, binding and checking of policies; it is possible to enforce DRM policies using standards engines; and exchanging policies is easy. For this reason, the design of semi-structured metadata for DRM policies' representation and enforcement is likely to remain an important topic of security research in the next years.

## REFERENCES

- Adams, A. (2000). Multimedia information changes the whole privacy ballgame. In *Proceedings of Computers, Freedom and Privacy 2000: Challenging the Assumptions* (pp. 25-32). ACM Press.
- Adams, A., & Sasse, M. (1999a). Privacy issues in ubiquitous multimedia environments: Wake sleeping dogs, or let them lie? In *Proceedings of INTERACT'99* (pp. 214-221), Edinburgh.
- Adams, A., & Sasse, M. (1999b). Taming the wolf in sheep's clothing: privacy in multimedia communications. In *Proceedings of ACM Multimedia* (pp. 101-107), Orlando.
- Bonatti, P., Damiani, E., De Capitani di Vimercati, S., & Samarati, P. (2001a, June). An access control system for data archives. In *16th International Conference on Information Security*, Paris, France.
- Bonatti, P., Damiani, E., De Capitani di Vimercati, S., & Samarati, P. (2001b, December). A component-based architecture for secure data publication. In *17th Annual Computer Security Applications Conference*, New Orleans, Louisiana.
- Curie, D., & Irvine, C. (1996, October). Surmounting the effects of lossy compression on steganography. In *Proceedings of the 19th National Information System Security Conference* (pp. 194-201), Baltimore, MD, USA.
- Damiani, D., De Capitani di Vimercati, S., Fernandez-Medina, E., & Samarati, P. (2002a, July). An access control system for svg documents. In *Proceedings Of the Sixteenth Annual IFIP WG 11.3 Working Conference on Data and Application Security*, University of Cambridge, UK.
- Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., & Samarati, P. (2002b, May). A fine-grained access control system for xml documents. *ACM Transactions on Information and System Security (TISSEC)*, 5 (2), 169-202.
- DOI. (2003, July). *The DOI Handbook. Edition 3.1.0*. Retrieved from the World Wide Web: <http://www.doi.org/handbook2000/DOIHandbookv3-1.pdf>.
- DRM. (2003). DRM concepts & standards. *GiantSteps*. Retrieved from the World Wide Web: <http://www.giantstepsmts.com/drm.htm>.
- DRM. (2003). *Information mechanics. The digital rights management home page*. Retrieved from the World Wide Web: <http://www.info-mech.com/drmhomepage.html>.
- Erickson, J. (2002a, September). *OpenDRM: A standards framework for digital rights expression, messaging and enforcement*. Retrieved from the World Wide Web: <http://xml.coverpages.org/EricksonOpenDRM20020902.pdf>.

- Erickson, J. (2002b, March). *XrML 2.0 Technical Overview*. Retrieved from the World Wide Web: <http://www.xrml.com/reference.asp>.
- Jajodia, S., Samarati, P., Sapino, M.L., & Subrahmanian, V.S. (2001, June). Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26 (2), 18-28.
- Johnson, N.F., Duric, Z., & Jajodia, S. (2000). *Information hiding: Steganography and watermarking - attacks and countermeasures*. Kluwer Academic Pub Books.
- JPEG. (2003). *JPEG*. Retrieved from the World Wide Web: <http://www.jpeg.org>.
- Kay, R. (2003, May). XACML. *Computerworld*. Retrieved from the World Wide Web: <http://www.computerworld.com/developmenttopics/development/story/0,10801,81295,00.html>.
- Koch, E., & Zhao, J. (1995, June 20-22). Towards robust and hidden image copyright labeling. In *Proceedings of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing* (pp. 452-455), Neos Marmaras, Greece.
- Kodali, N., & Wijesekera, D. (2002, November). Regulating access to smil formatted pay-per-view movies. In *Proceedings of ACM Workshop on XML Security* (pp. 53-60), Fairfax VA, USA.
- Loo, P. (2002, March). *Signal processing and communications laboratory* (PhD thesis). Department of Engineering, University of Cambridge. Retrieved from the World Wide Web: <http://www.eurasip.org>.
- MPEG-21. (2002, October). International organisation for standardisation. *MPEG-21 Overview v.5*. Retrieved from the World Wide Web: <http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm>.
- MPEG-7. (2002). International organisation for standardisation. *MPEG-7 Overview v.8*. Retrieved from the World Wide Web: <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>.
- Rosenblatt, B., & Dykstra, G. (2003, May). Integrating content management with digital rights management. *GiantSteps*. Retrieved from the World Wide Web: [http://www.giantstepsmts.com/drm-cm white paper.htm](http://www.giantstepsmts.com/drm-cm%20white%20paper.htm).
- Rosenblatt, W. (2002, January). *Digital rights management: Business & technology*. M. & T. Publishing Ltd.
- Samarati, P., & De Capitani di Vimercati, S. (2001). Access control: Policies, models, and mechanisms. In R. Focardi & R. Gorrieri (eds.), *Foundations of Security Analysis and Design* (LNCS 2171). Springer-Verlag.
- SAML. (2003). Advancing SAML, an XML-based security standard for exchanging authentication and authorization information. Retrieved from the World Wide Web: <http://www.oasis-open.org/committees/security/>.
- Sellars, D. (1999). An introduction to steganography. Retrieved from the World Wide Web: [www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.html](http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.html).

- SMIL. (2001, August). *World Wide Web Consortium. Synchronized Multimedia Integration Language (SMIL 2.0)*. Retrieved from the World Wide Web: <http://www.w3.org/TR/smil20>.
- SOAP. (2000, May). *World Wide Web Consortium. Simple Object Access Protocol (SOAP) 1.1*. Retrieved from the World Wide Web: <http://www.w3.org/TR/SOAP>.
- SVG. (2001, September). *World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.0 Specification*.
- Tsinarakis, C., Fatourou, E., & Christodoulakis, S. (2003, June). An ontology-driven framework for the management of semantic metadata describing audio-visual information. In *Proceedings of the 15th Conference On Advanced Information Systems Engineering* (pp. 340-356), Klagenfurt/Velden, Austria.
- VoiceXML. (2002, April). *World Wide Web Consortium. Voice Extensible Markup Language (VoiceXML) Version 2.0*. Retrieved from the World Wide Web: <http://www.w3.org/TR/voicexml20>.
- XACML. (2003a, February). *OASIS. Extensible Access Control Markup Language v.1.0*. Retrieved from the World Wide Web: [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).
- XACML. (2003b, June). *SUN. Introduction to XACML*. Retrieved from the World Wide Web: <http://sunxacml.sourceforge.net>.
- Xpath. (2001, December). *World Wide Web Consortium (W3C). XML Path Language (XPath) 2.0*. Retrieved from the World Wide Web: <http://www.w3.org/TR/xpath20>.

## **SECTION IV:**

# **SERVICE ORIENTED COMPUTING FRAMEWORKS**

## Chapter VIII

# Data and Application Security for Distributed Application Hosting Services

Ping Lin, Arizona State University, USA

K. Selçuk Candan, Arizona State University, USA

### ABSTRACT

*The cost of creating and maintaining software and hardware infrastructures for delivering web services led to a notable trend toward the use of application service providers (ASPs) and, more generally, distributed application hosting services (DAHSSs). The emergence of enabling technologies, such as J2EE and .NET, has contributed to the acceleration of this trend. DAHSSs rent out Internet presence, computation power, and data storage space to clients with infrastructural needs. Consequently, they are cheap and effective outsourcing solutions for achieving increased service availability and scalability in the face of surges in demand. However, ASPs and DAHSSs operate within the complex, multi-tiered, and open Internet environment and, hence, they introduce many security challenges that have to be addressed effectively to convince customers that outsourcing their IT needs is a viable alternative to deploying complex infrastructures locally. In this chapter, we provide an overview of typical*

*security challenges faced by DAHSs, introduce dominant security mechanisms available at the different tiers in the information management hierarchy, and discuss open challenges.*

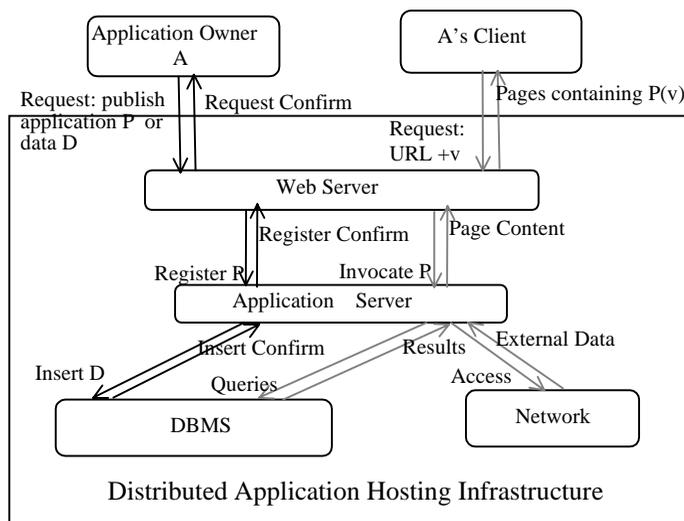
## INTRODUCTION

In an e-business setting, distributed computation with multiple parties' participation is typical. Most business tasks, for example calculating the collective financial health of a group of independent companies, are inherently distributed (Franklin et al., 1992). Consequently, most businesses need an information technology (IT) infrastructure capable of providing such distributed services. For most businesses, investing in a local, privately owned infrastructure is not economically meaningful. For instance, an e-commerce company may find deploying an infrastructure that can handle peak demand volumes, while sitting idle most other times wasteful. Therefore, businesses are willing to pay premium prices for third-party solutions that can help them reduce their infrastructural needs while providing them appropriate quality of service guarantees. Consequently, application service providers (ASPs) and distributed application hosting services (DAHSs), which rent out storage, (Internet) presence, and computation power to clients with IT needs (but without appropriate infrastructures) are becoming popular. Especially with the emergence of enabling technologies, such as J2EE (J2EE, 2003) and .NET (.NET, 2003), there is currently a shift toward services hosted by third parties.

Most DAHSs typically deploy a large number of servers to host their customers' business logic and data. They employ hardware- or software-based load balancing components to provide quality of service guarantees to customers. In addition, DAHSs can also place or replicate applications and data in servers closer to the end-users to eliminate network delays. Examples include Akamai and MirrorImage.

A typical application hosting infrastructure (*Figure 1*) consists of three major components: database management systems (DBMSs), which maintain business data, application servers (ASs), which encode business logic of the customers, and web servers (WSs), which provide the web interface between end-users and the business applications that are hosted by the DAHS. Although there are various modes of DAHS operation, a common way hosting services are used is as follows: (1) the customer (or application owner) with an application program publishes this application along with the relevant data onto the servers of the host. (2) Whenever they need, the customers (or its clients) access this application remotely by passing appropriate parameter variables to the host using the web interface. (3) User requests invoke appropriate program scripts in the

Figure 1: Components of a distributed application infrastructure



application server, which in turn issue queries to the underlying DBMS to dynamically generate and construct responses. In other words, the host runs the application with the local or provided data and sends the result back to the requesting party. (4) The host charges the application owner based on the resources (such as bandwidth, CPU, or storage) required for the processing of the request. *Figure 1* shows a typical application hosting infrastructure and its operation flows.

Distributed application hosting services, on the other hand, pose various security challenges due to their inherently distributed and mostly open natures. Without proper security provisions, customers will not choose to outsource services, hence DAHS will not survive. In this chapter, we provide an overview of various security challenges faced by DAHSs, discuss the techniques developed to address these challenges, introduce the security technologies and protocols used at different tiers in the Internet information management hierarchy, and discuss still-open issues.

The structure of the chapter is as follows. In the next section, we give an overview of the basic data and application security services and security tools. We also enumerate the security challenges faced by DAHSs. Then, we discuss security mechanisms adopted by widely used information management architectures. In the subsequent sections, we discuss the techniques available to DAHS to tackle these challenges and highlight the open problems.

## OVERVIEW OF DATA AND APPLICATION SECURITY

In data, content, and information delivery systems, security generally refers to the ability of a system to manage, protect, and distribute sensitive information so that information is free from eavesdropping, tampering, and spoofing and the service is available to all legitimate users. Therefore, security plays an essential role in e-commerce and e-business applications, where quality of information and service delivery means money, and military systems, where secure information and service links directly to national safety and human life.

Basic data, application, and system security services needed by such systems can be categorized as follows:

- **Authentication:** All entities in the system should be properly identified.
- **Authorization and confidentiality:** Access to applications or information should be restricted only to those entitled entities. Data or application code should be unintelligible to any non-entitled entity.
- **Integrity:** Data or applications should be free from unauthorized modification and damage.
- **Availability:** Data, application, and the system should be available to users despite attacks or damages.
- **Auditing:** Records of security relevant events (such as authentication, authorizing decisions, or abnormal events) should be kept for assessing attacks and intrusions or for evaluating effectiveness of security policies and mechanisms.

Although they address different security challenges, at their foundations, all these services rely on basic cryptography techniques. For a comprehensive background in cryptographic protocols and tools refer to Section I of this book. In this chapter, we focus on the security challenges peculiar to DAHSs.

### Security Challenges and Security Mechanisms in Distributed Application Hosting Services

Distributed application hosting services (DAHSs) face a number of security challenges. In small, closed local area networks, the mutual trust between clients and hosts is high. Clients can fully trust all hosts and the communications are reliable. However, in open, wide area networks, such as the Internet, where hosts are added and removed dynamically, it is very possible that clients and hosts have little mutual trust. In an environment where servers may not always be honest, data security constitutes a major concern to users. Executing an application remotely exposes the application code and data to non-trusted,

potentially malicious, distributed computing infrastructures. A malicious host may make use of clients' private information to gain illegal profits or cause damages to systems by tempering. Certain applications, such as business transactions and military applications, do not lend themselves well to the risks involved in simple outsourcing computation tasks to third parties. How to protect application code and input data from malicious executing environments, therefore, is a critical challenge.

As discussed earlier, techniques for data security occupy a wide spectrum. Most mechanisms aim at protecting data from unauthorized accesses. On the other hand, users' queries to data servers or private inputs to outsourced applications may also be of value and, without proper protection, important information may be leaked to the untrusted or compromised servers. For example, in a stock database, the type of stock a user is querying is sensitive information and may need to be kept private, sometimes even from the database server. Hence, traditional network-level encryption schemes may not be enough.

The security concerns in DAHSs can be broadly categorized as follows:

- System resources may be accessed by malicious or illegal clients so that sensitive information is leaked.
- Legal clients may access more system resources than they are entitled to, hence damaging these resources or preventing other clients from accessing these resources.
- Clients' application, data, or requests may be leaked, modified, or lost when they are being transported by insecure communication channels or executed by malicious hosts.

A qualified application hosting infrastructure should provide proper mechanisms to tolerate any faulty or malicious actions listed above. Although these mechanisms have already been briefly introduced earlier in this section, in the remainder of the chapter we focus on the DAHS specific challenges in *authentication*, *authorization*, and *confidentiality*.

- ***Authentication in DAHSs:*** Authentication means verification and validation. Identity authentication enables verifying the identities of the entities participating in the application hosting services (either the clients or the hosts) to make sure that both ends at the communicating channel have the right to perform their tasks. Services will be denied to unauthorized clients. Data authentication, on the other hand, verifies the origin and the integrity of data. In DAHSs, data owners usually outsource their data and delegate their services to untrusted third-parties. Hence DAHSs should provide mechanisms to enable clients to verify query answers. Application can be delivered across networks for remote execution. This gives rise to two

authentication issues: (1) to authenticate that the application is safe and does not contain malicious code; and (2) to authenticate that the application has been correctly executed on untrusted remote sites.

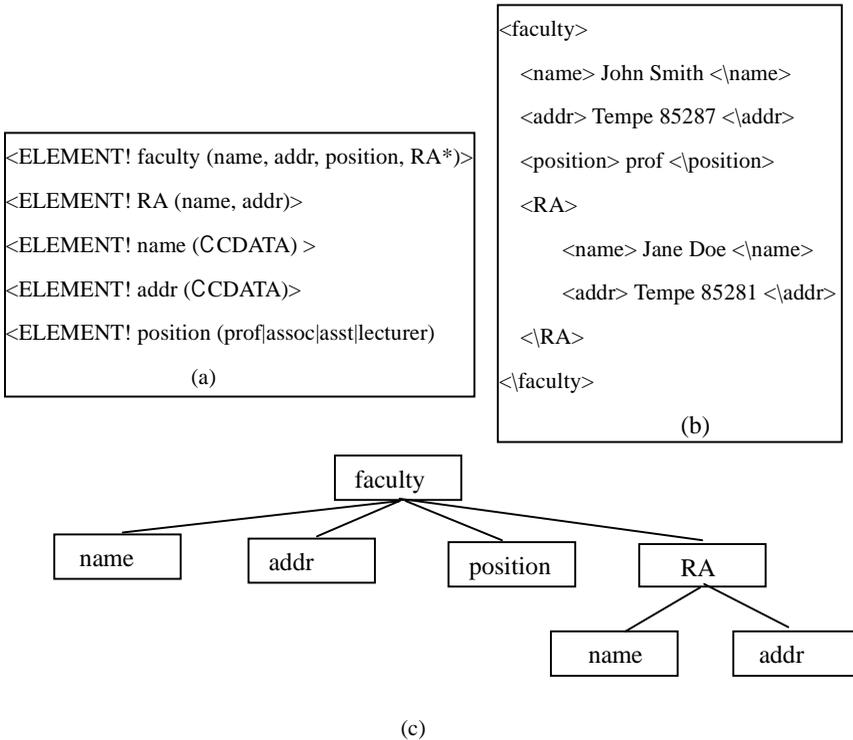
- **Authorization in DAHSs:** In order to protect resources of DAHS hosts, security policies should be specified by hosts to restrict clients' access to resources. If any violation occurs, proper action will be taken by hosts, including the termination of service.
- **Confidentiality in DAHSs:** Private information of DAHS clients (including outsourced application code and data) is not leaked to or modified by any third party when transported through the Internet, nor DAHS hosts when the code is executed or the data is stored. Confidentiality can be achieved by encryption, private information retrieval, computation hiding, and information hiding; noticing that in some cases, users' queries also need to be kept private, private information retrieval prevents query as well as results from being disclosed to the host; computation hiding seeks to hide users' private input data or code from partially trusted hosts; and information hiding is used to hide not only the content but also the existence of communication from possible attackers.

Traditional security mechanisms like authentication, access control, and cryptography have been well studied and established through various industry standards. On the other hand, some of the required technologies, such as private information retrieval, computation hiding, and information hiding are new research areas and the underlying techniques are not standardized yet. We will revisit authentication, authorization, and confidentiality challenges in DAHSs and discuss the related technologies in greater detail. Having covered the background in security technologies, however, we now proceed to compare security challenges and provisions of popular data and information management architectures that form the basis of DAHSs.

## COMPARISON OF SECURITY PROVISIONS OF VARIOUS ENABLING SYSTEMS

The diversity of distributed application environments and usage scenarios of computer systems contribute to the diversity of the security concerns faced by DAHSs. Since many applications, such as those involved with e-commerce, contain common modules, independent software developers can save a great deal of time by building their applications on top of existing modules that already provide required functionalities. This calls for a distributed architecture, where different modules can locate each other through directory services and can exchange information through messaging systems. J2EE and .NET are two

Figure 2: (a) An XML DTD, (b) A matching XML document, and its (c) graph representation



popular distributed application architectures, and Chapter 9 of this book provides a detailed comparison of the security measures these architectures provide. In this section, we discuss security concerns and mechanisms in various enabling software and hardware systems.

## Web Services and XML

Web services are standardized ways of integrating web-based applications. The primary function of web services is to allow independent web entities to communicate with each other without considering the underlying IT systems. Web service integration is done through programmatic interfaces, which are operating system independent and programming language neutral. Currently, there are various standards that enable web service integration. Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI) are open enabling standards. XML is used to organize the data with tags enabling applications to understand the structures of each

other's data; SOAP is a light-weight protocol, based on XML, to encode the data for exchange between web applications; WSDL is an XML-based language to describe web services so that web applications can identify services that they need; and UDDI is a directory that lists available services on the Internet and enables applications to discover each other.

- ***Hierarchical structure:*** XML data has a graph structure explicitly described by user-defined tags. The basic objects of XML data are elements, attributes, and references (idrefs). An element is a semantically whole unit. It may include subelements as components as well as attributes describing its relevant features. An idref links an element to another. Objects of an XML document are connected via element-subelement, element-attribute, element-link relationships and, therefore, form a graph (Figure 2). If references are removed, the graph is degraded into a tree structure; hence the structure of any XML object may be regarded as a hierarchy.
- ***Declarative structure:*** XML data may be *validated* or *well formed*. A validated XML data conforms to a structure defined by a given Data Type Definition (DTD). A well-formed XML data follows grammar rules of XML but has no associated DTD file to define its structure. A well-formed XML data may partially conform to some DTD or not conform to any DTD.

XML encryption is the standard process specified by W3C to encrypt any XML web resource (Reagle, 2003). It supports symmetric and asymmetric encryption as well as super-encryption (i.e., encryption of data that has already been encrypted). It can also support block encryption (encryption algorithms, such as DES, that divide plaintext into fixed-length blocks and then encrypt each block respectively) and stream encryption (encryption algorithms, such as SEAL, that manipulates the plaintext in the units of bits). It supports various key exchange protocols, such as RSA-v1.5, RSA-OAEP (Fujisaki et al., 2000), and Diffie-Hellman. With XML encryption, both the schema and the content of XML documents can be hidden from non-entitled entities.

Transport layer security protocols, such as SSL/TSL or network layer security protocols (IPsec) lay the foundation for secure messaging in web services. These protocols enable client/server authentication, data integrity, and confidentiality as discussed before. XML signature (Section AUTHENTICATION) and XML encryption can be used within SOAP to provide message-level authentication and persistent confidentiality. IBM, Microsoft, and VeriSign proposed WS-Security specification to describe how to attach signatures and encryption headers to SOAP messages (Atkinson, 2002).

At the higher levels, there is a need for service-level security models. For example, IBM provides a general security model for web services in .NET

*Table 1: Oracle and DB2 comparison (Sources: Oracle, 2003; DB2, 2003)*

	Oracle	DB2
Authentication	Provided	Provided
View	Provided	Provided
Access Control	Row level access control	Table (view) level access control
Encryption	Row level encryption	Table level encryption
Auditing	Fine grained	Less granular

(2003). In this model, a web service is accessed by requiring each requester to carry with the request messages some proof that certain specified claims are satisfied. The associated claims and relevant information constitute the policy for the web service. A security token for an entity encodes security information, such as user name or digital certificate. The requester proves required claims by attaching its security token to request messages or by asking the security token from a special web service called Security Token Service (which may ask for its own claims). IBM and Microsoft are developing WS-Policy to describe capabilities and constraints of the security policies on entities, WS-Trust to describe the trust models for Web services, WS-Privacy to enable web services and requesters to declare privacy requirements and practices, WS-Secure Conversation to describe message exchange authentication and management, WS-Federation to describe the management of trust among heterogeneous systems, and WS-authorization for describing how to manage authorization data and security policies (.NET, 2003).

Although most web applications reside in and above the application server tier that use these two technologies, data storage and management is usually left to database management systems (DBMSs). Next, we will consider security provisions of two popular DBMSs.

## Database Management Systems (Oracle and DB2)

Views, private virtual databases, access control, and encryption have long been practiced by back-end database systems to achieve security. In *Table 1* we compare security mechanisms used by two popular database management systems: Oracle (Oracle, 2003) and DB2 (DB2, 2003).

- Oracle's security mechanisms are embedded into the database. For DB2, there are also various tool suites to help secure DB2 data. Oracle and DB2 both provide authentication. DB2's authentication works closely with the underlying operation system's security features to verify user IDs and passwords.
- Both of their access controls are role based. Oracle can provide access control explicitly to row level data. In DB2, access control can be explicitly granted to tables or views and row level access control is gained implicitly by defining views on rows.
- Both of them provide database encryption. Through GUI interface provided by the Encryption Wizard of Oracle, encryption can be conducted at schema, table, or column levels. With application tools such as IBM DATA Encryption, encryption for DB2 data can be conducted at table levels.
- A view is a specific way of looking at a database. Generally, a view is constructed by arranging data in some order and making only some parts of the data visible. Views can hence be used as a mechanism to grant various types of access rights on the same data. Oracle and DB2 both provide views and view-based access control.
- Oracle and DB2 both provide auditing to capture relevant data. Oracle provides session auditing at the schema, table, and column levels to trace users' encryption/decryption operations. DB2's auditing facility acts at the instance level, where an instance is a database manager that maintains a set of databases that cannot be accessed by other instances.

While application and database servers are dominant architectural components for e-businesses and web service providers, they also form the basis of *grids* that integrate scientific as well as business applications and data.

## Data and Computation Grids

Grid computing refers to the system design approach that benefits from resources available from various devices in a network to solve a complicated problem that usually needs a huge number of processing cycles or a large amount of data from different origins. Grid computing makes all heterogeneous systems across an enterprise or organization virtually shared and always available to any legal members. Consequently, through sharing of computing capabilities and data, grid computing accelerates processing of problems that are too complicated for a single machine to handle. Hence, it enables complex business problems, computing intensive scientific problems, or large data analysis problems to be solved rapidly. The need to share resources and to execute untrusted code from any member of the grid introduces various security concerns for grid computing. In grid systems, authentication and authorization should always be present. Data grids should integrate a variety of databases regardless of which operating

system they reside on. Although each database may have its own security requirements and provisions, the data grid should provide access control that respects each data source's policy while satisfies users' data needs to the greatest possible extent. Although, there are no established standards, Butt et al. (2002) proposed a technique, which uses run time monitoring and restricted shells, to enable maximum legitimate use permitted by security policies of a shared resource. The application architectures, DBMSs, and resources on a grid are connected with each other as well as the end users through wired or wireless networks. Hence, network security is essential.

## Wired and Wireless Networks

Firewalls and secure socket layer (SSL) communication are two typically used techniques to achieve network security. Firewalls isolate to-be-protected servers from the open Internet so that only messages from authenticated sources can penetrate through. Firewall authentication relies on package filtering or stateful package inspection to check the originating and destination address associated with messages. The main deficiency of the firewall technique is that it does not inspect the content of the messages. An attacker who achieves access by misrepresenting his/her identity may exploit this security hole by sending malicious content. SSL is an open protocol, developed by Netscape, to provide secure HTTP connection and data transmission between web browsers and web servers. Theoretically, it is a protocol at the transport layer of network protocol stack. Its function is to establish secure communication sessions between clients and servers. Before data communication starts, by a handshake protocol, SSL allows a client and a server to authenticate each other through asymmetric cryptography and X.509 certificates, as well as to negotiate the encryption algorithm and cryptographic keys to be used during secure data communication.

Figure 3: TSL and WTLS

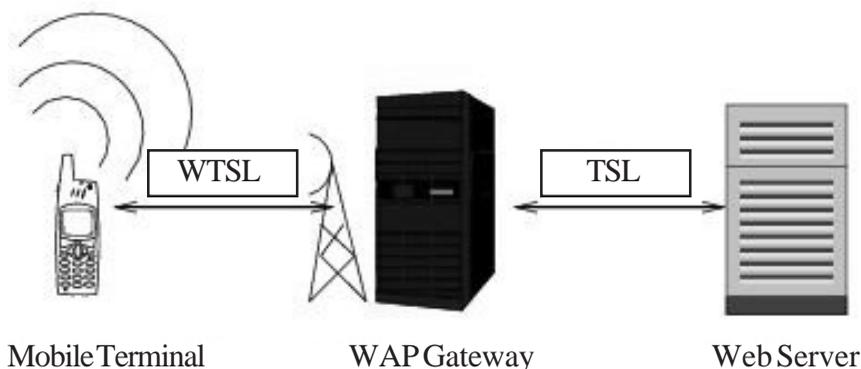


Table 2: Comparison of TSL and WTSL (Source: Wright, 2000)

	TSL	WTSL
Usage Environment	Wired networks	Wireless networks (WAP)
Protocols	TCP	TCP or UDP
Support for session suspend and resume?	YES	YES
Compatible with SSL	YES	NO

From that moment on, all data is encrypted by symmetric cryptography. To check data integrity, data is transported along with keyed MAC checksum. The SSL version often used in wired networks is TLS.

In wireless networks, the counterpart of TLS is WTSL. A mobile device can establish secure communication session to a wireless access protocol (WAP) gateway through WTSL. Then, on behalf of the mobile device, this WAP gateway can establish secure communication session to the target server through TSL over wired networks (*Figure 3*). Since WTSL is not compatible with TSL, the WAP gateway has to do translation between them. Therefore, data is encrypted all the way between the mobile client and the server except on the WAP gateway where the translation occurs. If the WAP gateway is compromised, confidential data may be leaked. Two possible ways to achieve end-to-end security are to eliminate the WAP gateway or to add application level encryption. *Table 2* gives a comparison of TSL and WTSL.

- TSL is a *de facto* standard to establish secure session between clients and servers in the Internet. WTSL is its counterpart in wireless networks (WAP). Both of them are transport layer protocols. They both support encryption, public key infrastructure, digital signature, certificate, etc.
- TSL relies on reliable network connection (TCP). WTSL can be established on unreliable network connection (UDP).
- To adapt for mobile environments where connection is not stable and may be lost easily, WTSL supports suspended or resumable sessions. Support for resumable session is also an option for TSL.
- TSL is derived from SSL. WTSL, on the other hand, is not compatible with SSL.

Having covered the security challenges and provisions in enabling architectures, in the remainder of the chapter we will focus on authentication, authorization, and confidentiality related challenges and provide an overview of the solutions proposed and techniques developed to address these issues. We will review the state-of-the-art as well as state-of-the-practice, highlight open

challenges and directions, and discuss impact of data and code security on various applications.

## AUTHENTICATION

Authentication involves verification of the claims made by parties in a secure environment. A common authentication task is (a) the verification of the identity of a communicating entity: without knowing for certain the identity of a client, for example, it is not possible to decide whether to accept or deny the access request. In addition, (b) data exchanged within a DAHS also needs to be authenticated to verify that it is really from the claimed origin and that its content is really what the source claims to have sent. In such an untrusted environment, answers to database queries should also be verified. Finally, (c) application authentication involves verifying whether execution of the application on untrusted DAHS servers is correctly performed or not. In this section, we will consider various authentication tasks in DAHSs.

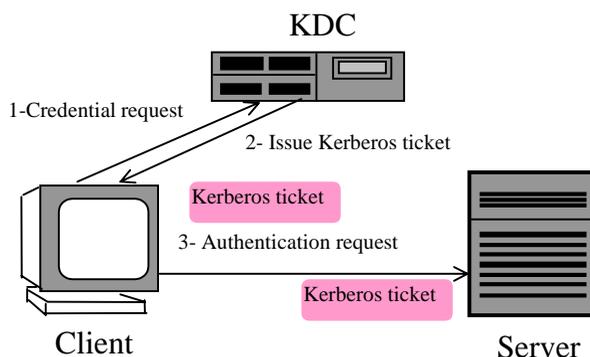
### Identity Authentication

Identity authentication protocols may be classified into two: (1) trusted third party authentication and (2) authentication without a trusted third party.

#### *Trusted Third Party Authentication*

Trusted third party authentication relies on a third party that is trusted by both communicating parties. Kerberos protocol (Kohl & Neuman, 1993), which is the *de facto* standard for network authentication, is a typical example. Kerberos relies on symmetric cryptography. In this protocol, the trusted third party is called the Key Distribution Center (KDC). Authentication protocol consists of three steps (as shown in *Figure 4*):

*Figure 4: Kerberos authentication*



- i) The client sends a request, which includes its and the server's identities ( $c$  and  $s$  respectively), together with a nonce  $t$ , to the KDC. The nonce is used to prevent replay of the request and should be a value, such as a timestamp, that cannot be changed.
- ii) KDC creates a session key  $K_{c,s}$ , and sends the session key and the nonce that are encrypted with the client's secret key  $K_c$ , back to the client. The KDC also issues credentials with which the client can access the server. A credential is a ticket,  $T_{c,s} = \langle c, K_{c,s}, \textit{expiration\_time} \rangle$ , that can be used to identify the client at the server. The ticket is encrypted with the server's secret key  $K_s$ , to prevent the client from tempering with it.
- iii) Finally, the client sends to the server an authenticator which includes
  - (1) the current timestamp  $t_c$ , encrypted using the session key, and
  - (2) The ticket  $T_{c,s}$  which was encrypted by the KDC with the server's secret key.

The server, after receiving the authenticator in Step 3, can establish the identity of the client by (1) decrypting the ticket, (2) extracting the identity of the client and the session key, and then (3) using the session key to decrypt the authenticator to see if the timestamp is current. If so, under the assumption that the KDC is trustworthy, the request is known to come from the stated client. If the identity of the server is also required to be authenticated to the client, (4) the server will respond with an incremented timestamp encrypted with the session key. This will enable the client to know that the server is able to read the timestamp, which means that the server has access to the session key, thus it is indeed the target server. After the authentication is over, the client may confidently communicate with the server using the session key.

Note that, in the above protocol, each time a client communicates with a server, trust between the client and the server has to be established using server's and client's secret keys. To reduce the probability of disclosure of the secret keys, the Kerberos protocol can be enhanced by allowing a Ticket Granting Server (TGS). In the enhanced protocol KDC has access to client's and TGS's secret keys, whereas the server's secret key is only known to the TGS.

One major advantage of the Kerberos protocol is that it only involves efficient symmetric encryption. On the other hand, it relies on the absolute security of KDC. If KDC is corrupted, the security system will be compromised. In order to prevent a corrupt third party to break the security of the system, other protocols aim to eliminate the need for a trusted third party.

### *Authentication without a Trusted Third Party*

Public key cryptography can serve as an authentication protocol (Nace & Zmuda, 1997). Let us assume that there are two communicating parties,  $S_a$  and

$S_b$ . Each party has a key pair ( $Pub$ ,  $Priv$ ), which includes a public key and a private key, respectively. Let us denote  $S_a$ 's key pair as ( $Pub_a$ ,  $Priv_a$ ) and  $S_b$ 's key pair as ( $Pub_b$ ,  $Priv_b$ ). The authentication procedure is as follows:

- i)  $S_a$  and  $S_b$  exchange their public keys.
- ii)  $S_a$  generates a random number  $R_a$ , sends it to  $S_b$ .
- iii)  $S_b$  responds with  $Priv_b(R_a)$ , and another random number  $R_b$ .
- iv)  $S_a$  decrypts  $Priv_b(R_a)$  with  $Pub_b$ . If she obtains  $R_a$ , she knows the other party is  $S_b$ , for only  $S_b$  can sign it with  $Priv_b$ .
- v)  $S_a$  responds with  $Priv_a(R_b)$ .
- vi)  $S_b$  decrypts  $Priv_a(R_b)$  with  $Pub_a$ . If he obtains  $R_b$ , he knows the other party should be  $S_a$ , for only  $S_a$  can sign the number with her private key.

After the trust has been established,  $S_a$  and  $S_b$  can communicate with each other in this way: before sending a message, each party encrypts the message with the other party's public key. The other party, after receiving the encrypted text, decrypts it with his/her own private key to retrieve the plain text.

The main advantage of the public cryptography authentication is that its security depends only on the two communication parties themselves. One main disadvantage of public cryptography authentication, however, is that it utilizes the inefficient asymmetric cryptography. Also, if a malicious third party intercepts the public keys being exchanged, it can replace them with different public keys and pose as one of the communication parties. A key exchange protocol, like Diffie-Hellman, may serve as a solution to this problem.

## Data Authentication

Data authentication involves verifying data's origin and integrity. Digital signatures can be used to prove the origin of a data message and hash (or digest) values can be used to check the integrity of the data being exchanged. In fact, by signing on the checksum hash value, both the origin and integrity can be verified. Basic tools for data authentication, therefore, include signature algorithms (such as DSA and RSA/SHA-1) and digest algorithms (such as MD5, MAC, and SHA). However, different types of data have different structures or usage contexts; hence the ways to digest or sign them may vary.

In DAHSs, data owners make their database available at third party servers. Since a single database contains more than one data object and since accesses to the database are through declarative queries (instead of explicit object ids), authenticating database accesses require techniques more elaborate than simple digests and signatures.

A correct database answer to a given query should be complete and inclusive. A complete answer must include all data elements that satisfy the query and an inclusive answer should not include any data that does not satisfy

the query. If the server hosting the database is trusted, then one possible authentication solution is to let the server certify answers by signing on them using a private key. However, in a DAHS, data owners may outsource their databases to untrusted third party publishers; hence, new protocols that authenticate database query results from untrusted publishers are needed. Some of these techniques are discussed next.

### *Authentic Database Publication*

Devanbu et al. (1999) propose a generic model for authentic third party data/database publication (*Figure 5*). The model consists of the following steps: (1) the data owner sends the database to the third party publisher; (2) the data owner signs the database digest and sends it to its clients; (3) a client queries the database stored at the publisher; (4) the publisher processes the query, sends the answer and some verification information to the client; and (5) using the verification information and the digest, the client verifies whether the answer it received is correct or not.

Query results can always be verified by submitting the whole database as the verification information to the client. Clearly, this would be very expensive. Hence it is crucial to develop proper database digest techniques that enable exchange of minimal verification information.

Devanbu et al. (1999) show that Merkle Hash Trees can be used to efficiently verify answers to selection, projection, join, and set operation queries that are common in relational databases. This protocol relies on the existence of database index trees, which are used for providing efficient access to the contents of the database. A trusted party (e.g., the data owner) recursively digests nodes of the index tree such that every leaf digest is a hash over the corresponding data value and every non-leaf digest is a hash over its children's

*Figure 5: Third party data publishing*

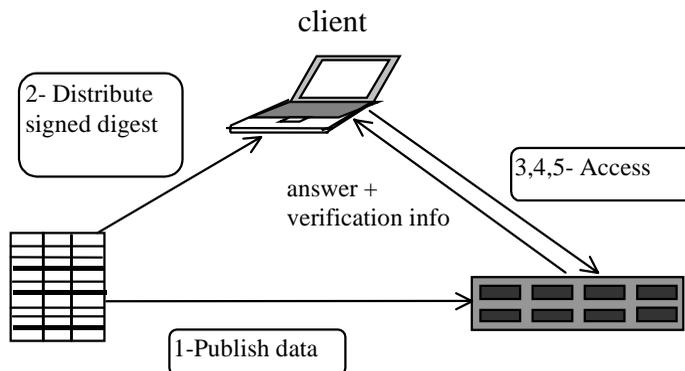
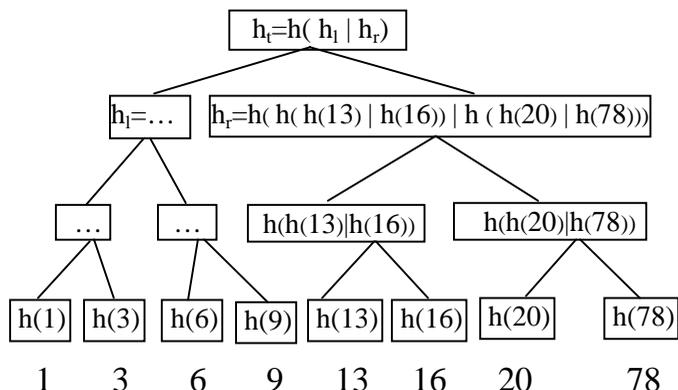


Figure 6: A balanced binary Merkle hash tree



digests (Figure 6). Merkle hash trees have two properties that enable authentication: Given the correct root digest,

- any modification to the tree structure can be detected; and
- the existence of a given subtree can be verified.

These two properties are fundamental for the verification of the inclusiveness of query answers. By requiring leaves of the tree being sorted according to some total order, it possible to enhance the Merkle hash tree with a third property: Given the correct root digest and a sequence of leaf values,  $q = \langle t_i, \dots, t_j \rangle$ ,

- the completeness of the sequence can be verified.

This enhanced property is fundamental in verifying the completeness of query answers. Based on these results, Merkle hash trees can be used for authenticating inclusiveness and completeness of relational query results.

As discussed in the subsection on web service and XML, on the other hand, in DAHSs and the web, the de facto standard to organize data is XML. Hence, next we look into mechanisms for authenticating data and databases published in XML format. First, we concentrate on signing XML data and documents and then we will discuss authentication procedures for third party XML database publication.

### XML Data Signatures

World Wide Web Consortium (W3C), which develops interoperable technologies and standards for the Web, has established an XML signature standard (Eastlake, 2003). This standard includes:

- a digest algorithm (SHA-1),
- a signature algorithm (DSA or RSA/SHA-1),
- a message authentication code (a hash function with a shared secret key), and
- transform algorithms to be applied to XML documents before they are digested. Transform algorithms add flexibility to the XML signature. For example, with path filtering transformations, the signer can choose to sign only nodes on a specified path in a given XML document.

W3C also specifies a progressive digesting procedure called DOMHASH (Maruyama et al., 2000) to recursively digest a DOM tree (Hégaret et al., 2003) (the native tree presentation of an XML document) from the leaf to the root, so that each node (element or attribute) has a digest value and the digest of a non-leaf element is a hash over its subelements and attributes' digests and its content. This strong version of the digest enables efficient comparison of two versions of an XML document to find the different parts.

XML signatures can be used to verify the integrity of a given XML document or some selected parts of it, but it cannot be used to verify the inclusiveness and completeness of answers to XML queries. Next, we discuss techniques for authentication of results to a query executed on an XML database published at an untrusted third party.

#### *Authentic Third-Party XML Database Publication*

Merkle hash trees work well for node selection queries (as discussed earlier), however, is not directly applicable for XML path queries, which require identification of all paths in a given XML document that match a given condition. Devanbu et al. (2001) proposes the following approach for creating XML digests using the DTD of a given XML document:

- i) The XML document owner builds an enhanced Merkle hash tree for each path type in the DTD and associates the root digest of the resulting Merkle hash tree with the corresponding path type.
- ii) The owner builds another enhanced Merkle hash tree from all path type digests and associates the root digest of this second tree with the given XML document.
- iii) The owner then signs the document digest and sends it to clients for verifying query results.

This enables efficient verification of the results to simple path queries in addition to the selection queries. For each path query, the publisher finds all path types that match the query and for each matching path type, it constructs a certificate. Using the XML digest provided by the owner, the client can verify

whether the certificate provided by the publisher includes all and only the results of the correct path type. Furthermore, the client can pre-calculate all path types that match the query and see whether for each matching path type, a certificate is received. Hence the client can verify the completeness and inclusiveness of the answer. This protocol, however, has the following limitations:

- It is computationally costly. Besides the cost of building Merkle hash trees, the owner needs to pre-calculate all subtrees for each path type. To verify each query, the client needs to find all matching path types.
- It has a large space overhead for the publisher: for each possible path type, a large certificate has to be maintained.
- It requires clients to know the DTD of the document to verify answers.
- It can not handle complicated path queries and queries over XML data that do not have a corresponding DTD.

Bertino et al. (2002) propose an alternative protocol that is cheaper and that does not need DTDs. This protocol utilizes DOMHASH to calculate root digest. See Chapter 6 of the book for a detailed discussion. Since this protocol is based on only one DOMHASH value for the entire XML document, it is cheaper than the previous protocol, which needs hashing for every path type. Also, since the path types do not need to be known in advance, this protocol does not need the DTDs. One disadvantage of the protocol, however, is that it lacks the ability to verify answers to selection queries, hence it is less flexible.

## Application Authentication

In distributed environments, application code can move among various distributed entities: (1) application code (such as Java applets) can be distributed from servers to clients for local execution; (2) application code can travel from thin mobile clients to powerful servers for execution; (3) in DAHSs, application code can be published and outsourced to remote server by the application owners; and (4) the code can travel between DAHS servers to achieve load balancing and process migration. For application code distribution, the recipient (either the client or the server) must validate the origin and the integrity of the code before loading, installing, and executing it. Otherwise, the recipient can be subject to a malicious or tampered source, which can gain unauthorized access to the recipient's resources or can receive a virus which can break down the recipient's machine and spy for sensitive information. The source also should use authentication techniques; otherwise, a malicious recipient may try to deceive the owner by providing false results. Furthermore, if the application code and the associated data visit multiples servers to conduct steps of a computation, a malicious server can modify the state of the code or the data it carries before the code moves To Whom It May Concern: the next server. For example, a malicious

airline server may raise the lowest airfare a mobile airfare agent code has computed from all prior visited airlines to cheat the client.

As in data authentication, checksums and digital signatures once again constitute the set of basic tools for application code authentication. Prior to transmission, the code owner can sign the code with its digital certificates. Upon receipt, the recipient can verify the signature and decide whether the origin and integrity of the code can be trusted. If the code is verified, the recipient then has to determine the execution permissions for the code. Although checksums and digital signatures can be used to identify the owner and recipient of the code, if these entities are themselves not trusted, we need additional mechanisms to verify the application code and the execution results. To prove that an application does not contain any malicious code as to damage internal data structure of the host, or overuse resources of the host, the application owner can provide an encoding of a proof that the code complies with the security policy of the recipient. Application code augmented as is called a *proof-carrying code*. To prove that the application is executed properly, on the other hand, the owner can benefit from *replicated execution* or *execution traces*. In a multi-agent system where there are multiple agents interacting, in order to prove that an agent is secure for interactions (i.e., the agent will not access services or data that it is not entitled to), the service provider can utilize agent's current state, records of previous interactions with other agents, and the analysis of possible consequences of the interaction. Next, we discuss proof carrying codes, replicated execution approaches, execution traces, and secure agent interactions individually.

### *Proof-Carrying Codes*

Proof-carrying code protocol (Necula & Lee, 1998), for proving that an application does not contain any malicious code, consists of two phases:

- i) The recipient of the code extracts from the untrusted application code safety predicates that can be proved if and only if the code conforms to the security policy defined by the recipient. Generally, a security policy defines (1) the language in which the application should be written, (2) the conditions under which the application can be trusted, (3) the interface between the recipient and the application code, and (4) the methods for inspecting application code and discovering potential security violations. The safety predicate is constructed by inspecting the instructions of the application to find ones that may violate the security policy and generating for each such instruction a predicate that proves the safe execution of the instruction. The recipient sends the safety predicates to a proof producer (such as the application owner) and the proof producer returns the proof back to the recipient.

- ii) The recipient, then, checks the proof via a proof checker. If the recipient verifies the correctness of the proof, it can safely install and execute the application. This protocol is general and the recipient does not have to trust any party (either the owner or the proof producer). However, the proof size can be very large: it usually grows linearly with the size of the application code, but it can grow exponentially in worst cases.

### *Replication*

One way to ensure the correct execution of the application code is via server (or recipient) replication and voting (Minsky et al., 1996). In this protocol, the execution of the code is divided into stages and each stage is executed at multiple servers:

- i) The owner dispatches the stage one execution to multiple server replicas. Every replica sends the stage one output to all replicas of stage two.
- ii) In the following stages, each replica chooses its input to be the majority of the outputs received from the previous stage. It then conducts its computation and sends the output to all replicas of the next stage. At the last stage, the replicas send their outputs back to the client.
- iii) Finally, the client determines the output to be the majority of the outputs it received from the replicas corresponding to the last stage of the execution.

In addition to the high network and computation bandwidth requirements, this protocol fails if the majority of the servers at a certain stage are malicious or compromised. To prevent a third-party server from spoofing as one of the replicas, this protocol can be extended with authentication schemes to verify each replica.

Yee (1997) proposes a code replication scheme in which a copy of the code is executed by visiting the sequence of servers in the reverse order of stages. This scheme is capable of detecting certain types of tampering with the results. For example, given a mobile agent code that searches the lowest airfare by visiting airline servers in a particular order, one copy of the agent can travel the same servers in the reverse order. In this case, any inconsistency between two results implies tampering. This protocol is simple and introduces low overhead (only a copy of the original execution required). However, it is effective only when the order in which the servers are visited does not make any difference in the final result.

### *Cryptographic Traces*

Verifying the execution trace of an application on a given server is another way to authenticate results (Vigna, 1998). An execution trace for an application on a given server records the statements executed by the server and the values

of the data obtained. The traces are protected via the server's certified digital signature; hence the server cannot disown the trace. By retrieving the traces from suspected servers and using them to simulate the overall execution, the client can identify any tampering.

One main disadvantage of this protocol is that it is a post-execution method and cannot offer timely authentication. Another disadvantage is that it does not provide any mechanism to detect tampering a priori, so the client does not have any indication regarding when to ask for traces from a server.

### *Secure Agent Interactions*

Bonatti et al. (2003) present a framework intended for multi-agent environments with authenticated the secure interaction between agents. The system keeps track of all agents' (1) histories, actions they performed, and messages they exchanged; (2) states that contain their current knowledge about the system; and (3) consequence operations, which describe what knowledge an agent can derive from its state and its reasoning capability. A secret is specified by the service provider agent in terms of actions and data forbidden to be accessed. Based on these, secure interaction in a multi-agent system is defined in terms of secure histories that do not leak any information via messages that are exchanged and consequences that do not violate any secrets. Maintaining correct and accurate information about an agent's state and consequence is almost impossible in practice. Hence, Bonatti et al. (2003) suggest using approximate information and proposes a rule-based logic language with which an agent can specify how to approximate the available information about other agents.

## **AUTHORIZATION**

Even when identities of the parties in a DAHS environment have been verified through authentication, there is still possibility of other forms of attacks by malicious entities. For instance, available resources should be protected and the access should be restricted, otherwise untrusted users may break down the system purposefully or even trusted users may, without any malicious intention, cause damage to the system by improper operations. Therefore, proper access control mechanisms that can ensure that the operations that an authenticated user (or an application program on behalf of the user) can invoke on a server lie within the limits of server's security policies are essential.

### **Security Policies**

Security policies specify what actions are allowed and what are not allowed. A policy has three dimensions: subjects, objects, and access types. Subjects are users or programs that work on behalf of the users. Objects represent resources

to be protected from subjects. Access types include actions, such as read or update that subjects can execute on objects. There are various security policy models (Sandhu & Samarati, 1994). In this section, we briefly discuss discretionary, mandatory, and role based policies.

### *Discretionary Policies*

A set of authorization rules defines the access mode for each subject and object pair. Every access is checked against this set of authorization rules. This model is simple and good for cooperative but independent environments, however, it cannot restrict what subjects will do with the information after they fetch it from the servers.

### *Mandatory Policies*

In this model, each subject and object is assigned a security level. For example, the security level of an object may reflect the sensitivity of the information associated with the object to unauthorized disclosure. The security level of a subject is called clearance and it may reflect the trustworthiness of the subject. Access is granted only if the security levels of the target object and the subject satisfy certain relationship. Generally, to read an object, the clearance level of the subject should be higher than or equal to the security level of the object; whereas, to write an object, the object being written must have equal or higher security level than the subject (Sandhu & Samarati, 1994). In this way, the information flow is guided in a way to prevent sensitive information flowing to lower level objects. This model fits well with stricter, such as military, environments.

### *Role Based Policies*

This model mediates subjects' access to objects according to subjects' activities (or roles). The model identifies all roles in the system and, with each role, it associates a set of operations and responsibilities. Access authorizations for objects are defined in terms of roles. Subjects are given permissions to play roles. A subject playing one role is granted all operations that are authorized for that particular role. Role based model is flexible. It does not assign access rights to subjects directly, but indirectly through roles. Hence, it avoids the cumbersome task of assigning and re-assigning access rights as the system evolves. This model is also space saving, as redundant specification of access rights assigned to users playing the same role is diminished.

## **Data Authorization**

In a DAHS environment, the sensitive information contained in each data source, managed on the behalf of the data owners, must be protected from unauthorized user accesses. Especially in a data grid, the underlying system

should provide authorization mechanisms for the databases that are being federated. Considering XML's role in distributed data and information exchange, special attention should be given to how the structure of XML data affects authorization.

### *Authorizing Federated and Mediated Databases*

DAHSs and data grids usually host applications that integrate heterogeneous packages (including application software and associated databases) outsourced from origins with different security policies.

Therefore, they need to enforce global security policies while respecting the local security policies of each individual data source. Different security models have been developed for federated and mediated databases. Idris et al. (1994) introduce a security model for federated systems where security levels may continuously change. Jonscher & Dittrich (1994) propose a decentralized authorization security model for tightly controlled federated databases. Blaustein et al. (1995) propose a security model that relies on bilateral agreements between data sources to identify how each party protects others' data. Wiederhold (1992; 1993) introduces a centralized model, based on mediators, which integrate heterogeneous databases with different security policies. Candan et al. (1996) introduce two cooperation principles that may be implemented by a mediated system:

- ***Cautious cooperation:*** If a user's query only needs to access information that the user is entitled to, the mediator will answer the query unless the global security policy directly forbids so, while each participating database's security policy is always respected.
- ***Conservative cautious cooperation:*** If a user's query only needs to access information that the user is entitled to, the mediator will answer the query unless from such query the user can infer information he or she is not allowed to access by global security policies, while each participating database's security policy is always respected.

Based on a rule-based mediator model that consists of a mediator  $M$ , a set of data sources  $\{d_1, d_2, \dots, d_n\}$  integrated by  $M$ , a global security policy  $G$ , and a set,  $V$ , of local security policies, Candan et al. (1996) further propose a formal approach for secure mediated databases. Global security constraints in  $G$  are modeled as:

- a set of facts of the form  $secret(A, i)$ , denoting users with security level  $i$  has no access right to the information (atom)  $A$  and

- a set of rules of the form  $secret(A,i) \leftarrow secret(A,j) \wedge i < j$ , enforcing that if  $A$  can not be accessed by certain security level, it can not be accessed by lower levels either.

Local security constraints,  $V$ , on the other hand, are modeled as:

- boolean functions of the form  $viol_d(d:f(\langle argument \rangle))$ , which identify whether executing function  $f$  with specified arguments on date source  $d$  for users of security level  $i$  violates  $d$ 's local security policy or not.

In order to prevent a malicious user from inferring unauthorized information through knowledge about the implemented security policy, Candan et al. (1996) adopt query transformation methods to ensure that the query simply fails (without raising violation alerts) if it violates any local security constraints.

Participating databases may have different security orderings; for example, the classification labels of security levels in different databases may be different or security levels with the same label may have different security orderings in different databases. To integrate such heterogeneous databases, there is a need for mechanisms to merge heterogeneous security orderings in a way that each individual security ordering is preserved and the constraints between security orderings of different databases are satisfied, while a maximal level of global security is maintained when there are conflicts. Bonatti et al. (1996) give a formal definition of this problem and propose two solutions: rule-based and graph-based approaches. The rule-based approach represents the semantics of security orderings and inter-database constraints using logic rules, while the graph-based approach represents them using a graph. Both methods can find a combined security ordering, for a given non-conflicting set of individual orderings, in polynomial time.

### *Authorization of XML Data*

As discussed in Section Security Challenges and Security Mechanisms in Distributed Application Hosting Services, XML security is an essential part of web-based information architectures. Therefore, developing access control mechanisms that understands the structure and properties of data in XML form to enforce selective dissemination of information over the web is essential for DAHSs. According to Bertino et al. (2001), an XML access control mechanism should at least:

- consider XML's rich, hierarchical structure and provide fine-grained authorization to components of a given XML data;

- provide both DTD-level and data-level authorization. DTD-level authorization applies to a set of data objects conforming to the same DTD, while data-level authorization applies to one particular document or its components;
- handle authorization to XML objects that are not conforming or partially conforming to a particular DTD;
- devise proper authorization propagation rules that refer to hierarchical relationships (such as DTD-data, element-subelement, element-attribute, and element-link) to propagate authorization policies of higher level components to lower level components. These rules should also provide mechanisms to solve propagation conflicts when a component has multiple inconsistent authorizations propagated from higher levels.

Due the rich structure of XML, a standard authorization mechanism for XML data remains an open challenge. Author-X (Bertino et al., 2001) is a tool that provides access control to XML data. Author-X satisfies the minimal requirements mentioned above. It adopts the discretionary access control model; the policies have the following format:  $\langle U, O, P, R, S \rangle$ .  $U$  denotes a user or a group of users to whom the authorization applies.  $O$  describes the object (DTD, XML data, or portions of them) to be protected.  $P$  denotes the access privilege (browsing or authoring) to be permitted or restricted.  $R$  provides the propagation rules (cascade the authorization to all descendants, limit the propagation to first-level descendants, or no-propagation). Finally,  $S$  denotes whether this is positive or negative authorization. Using negative authentication, a security manager can efficiently define authentications with exceptions (e.g., defining an authorization applying to a whole document except for some few elements).

Author-X defines the following conflict-resolution rules: (1) explicit authorizations override propagated ones; (2) if there are multiple propagated authorizations, the most specific one (lowest level in the hierarchy overrides the others); (3) if there are conflicts due to propagated rules at the same level, the negative authorizations override.

The process of authorization in Author-X is as follows: For the target XML data object, Author-X

- finds the associated DTD. If the XML document does not have an associated DTD, Author-X finds the DTD that the target document mostly conforms to (hence it handles the partially conforming documents);
- propagates all possible DTD-level and document-level authorizations and resolves all conflicts;
- prunes from the document all elements that do not have required positive authorizations (explicit or implicit); and

- evaluates the user query against the pruned document and extracts the target data.

The IBM alphaWorks XML Security Suite (XML Security Suite, 1999) is another tool that provides access control mechanism for XML documents. It shares some common features with Author-X: authorization propagation based on structure hierarchy and conflicting authorization resolution, implementation of an XML-based language (XACL) for specifying security policies, and fine grained authorization. On the other hand, unlike Author-X, IBM alphaWorks utilizes role-based access control, which is more suitable for e-commerce environments. It also accommodates context and provisional actions into the access control model. Hence the extended authorization policy can specify whether a given subject, under certain context (access request time or some other conditions), is allowed to access the given protection object in a given way or not. It can also specify provisional actions (such as logging the access decisions, encrypting specified elements, verifying signatures, reading, writing, or deleting specified elements) that have to be performed whether the access is granted or not. With this extended semantics, XML Security Suite integrates authorization and non-repudiation mechanisms for accessing XML documents and data objects; i.e., a subject cannot deny that it made an access attempt. IBM alphaWorks has two propagation directions, up and down, and when there are conflicting propagations, it arbitrarily selects one. This differs from the most specific based conflict resolution of Author-X.

Cho et al. (2002) propose a simple mandatory XML data security model in which XML elements may have associated security levels or inherit them from their parents. A user is allowed to access only those elements whose security levels are no more than her clearance level. To minimize the cost of checking security levels, for each query, the system rewrites a given XML query by identifying the appropriate amount of checking. Cho et al. (2002) achieve an optimal rewriting of the query. In general, checking whether or not a user has access right to a particular object from a given set of access control rules can be inefficient. Maintaining an explicit accessibility map that lists all users that can access a given object, on the other hand, is space-inefficient. Yu et al. (2002) introduce compressed accessibility maps to efficiently enforce authorization policies over XML data. The compression is achieved by taking advantages of the feature that XML data items grouped together have similar accessibility properties.

Open networks like the Internet are inherently insecure despite authentication and authorization schemes. Sensitive data or application code may be leaked to or compromised by attackers eavesdropping on the communication link between clients and hosts or disclosed to malicious hosts. The business logic or query results may be altered to cheat clients.

In the Section CONFIDENTIALITY, we discuss underlying confidentiality issues and related technologies.

## Application Authorization

Karnik (1998) summarizes several ways to authorize accesses to server resources by outsourced application programs in mobile agent-based systems. Similar approaches can also be implemented for application authorization in DAHSs:

- **Direct reference approach:** The server supplies the mobile agent with reference to the resource and screens all accesses via a security manager. The security manager checks against the security policies to see if each application method (or procedure) under execution is allowed to access the associated resources.
- **Proxy approach:** The server builds a specific resource proxy when a mobile agent asks for some resource. The proxy provides a safe interface to the resource. This safe interface looks the same as the original interface, but certain methods are disabled to prevent the agent from accessing the resource via methods that the security policy does not permit.
- **Capabilities approach:** This mechanism has been adopted in several distributed systems. Every agent, before accessing a resource, presents a credential containing its access rights to the server. Only after the server grants its approval can the agent access the resource. The credential is issued to the agent after its identity has been authenticated.
- **Wrapper approach:** The resource is encapsulated with a wrapper and agents have references only to this wrapper. The wrapper maintains access control lists and decides whether an agent has the authority to access the resource or not.
- **Protection domain approach:** There are two execution environments: one safe environment to host the agents and one trusted environment to provide access to the resource. The safe environment processes each potentially unsafe request of an agent according to its own security policy and screens unsafe requests. For safe requests, it calls methods of the trusted environment that provides access to the resource. The trusted environment can only be called by methods within this safe environment.

The proxy and capabilities approaches are flexible: an instance of a proxy or a capability can be generated dynamically and specifically to satisfy each application or agent code. The dynamicity of the capabilities approach, on the other hand, introduces various challenges: an application may propagate its capability to others or, if security policies change, capabilities may need to be revoked. The wrapper approach is simple and more static: there is only one

wrapper for each resource object and all applications share the same wrapper for that resource.

## CONFIDENTIALITY

Confidentiality (or privacy) means that private information of DAHS customers (including outsourced application code and data) is not leaked to or modified by any party. The degree of confidentiality of security schemes can be categorized into the following two classes:

- *Information theoretic privacy*: These schemes are built without any cryptographic assumptions and, hence, cannot be broken even with unlimited computation power.
- *Computational privacy*: These schemes are built on various cryptographic assumptions, usually about certain hard-to-compute problems. In these schemes, the goal is to ensure that there are no efficient computations (conducted by a randomized algorithm bounded by a polynomial time in the length of inputs) that can break the scheme. Cryptographic assumptions used by computational privacy schemes are based on one-way functions that are easy to compute but hard to inverse. The most popular one-way functions, such as integer factorization,  $\Phi$ -assumption, and quadratic residuosity problem, come from number theory (Goldreich, 1997).

Encryption schemes are the basic tools to achieve confidentiality. Besides the common data encryption methods that hide sensitive data, there have been some techniques for hiding other kinds of secret information from untrusted servers:

- *Information hiding* hides not only the communication content, but also the existence of communication between two parties, so that no suspicion arises that a secret communication exists.
- *Computation hiding* prevents host sites from gaining unauthorized information about the content of the published code, the data it executes on, or the outputs produced as a result of the computation.
- *Private information retrieval* aims to let users query a database without leaking to the database what data is queried.

In this section, we report on the state-of-the-art techniques for hiding applications, data, data distribution, and user's query from application and database servers. In Subsection Computation Hiding, we focus on how to keep data or application confidential from untrusted hosts and we describe solutions to the problem of computing encrypted functions with encrypted data. In

Subsection Private Information Retrieval, we give a general introduction to private information retrieval and in Subsection Private Informational Retrieval in XML Databases we specifically focus on private information retrieval in XML databases.

## Information Hiding

Cryptography deals with concealing the content of information. Information hiding, on the other hand, seeks to hide the existence of information. For some highly sensitive applications, such as military or intelligence agencies, even the existence of communication arouses suspicion. Hence to provide information hiding services, a DAHS should be able to hide the communication traffic itself, which cannot be achieved with ordinary cryptography.

Information hiding technologies include spread spectrum radio, which is used to hide wireless transmission; temporary mobile subscribers, which are used in digital phones to hide a given user's location; and digital watermarks, which are used to imperceptibly insert copyright information in digital images, video, and audio. As they are not heavily used in DAHSs and data grids, in this chapter, we will not discuss information hiding techniques in detail.

## Computation Hiding

In distributed computation environments, such as DAHSs, input data owners, application owners, and computation providers (servers) may be different parties distributed over networks. Computation hiding involves hiding secret data, proprietary code, or secret outputs during a distributed computation.

### *Secure Distributed Computing*

Secure distributed computing is also called fault-tolerant distributed computing or oblivious circuit evaluation. The basic underlying mechanism aims evaluating a function (or a circuit) to which each party has one secret input, so that the output becomes commonly known to all parties but all inputs remain secret. If there is a trusted agent, secure distributed computing is a trivial problem: each party can securely send its private input with the help of cryptographic protocols; the agent computes the function and then distributes the result. Secure distributed computation intends to solve the problem without the assumption of any trusted agent, i.e., to simulate a trusted agent over a set of mutually untrusted parties. Secure distributed computing protocols are built on various basic protocols:

- ***Bit commitment:*** A bit commitment protocol simulates the function a sealed opaque envelope used for committing a bit of information: once the bit is sealed and committed by the committer, the content of the bit can not

be changed (the envelope is closed and sealed); the receiver, upon receiving the sealed bit, cannot read it until the content is revealed by the committer (the envelope is opened and letter now can be read). Bit commitment schemes can be built on one-way functions (Naor, 1989). In secure distributed computing, bit commitment schemes are utilized to enable a party to commit to some information.

- ***Oblivious transfer:*** Oblivious transfer is a protocol to transfer a bit among the involved parties in a way that the information held by the sender and the receiver about the transfer is asymmetric. A simple version of the oblivious transfer protocol resembles an undependable post service: *A* wants to send a bit *b* to *B*. Let us assume that the bit is successfully transferred to *B* with a probability of 0.5. *A* does not know whether the bit is successfully transferred or not but *B* always knows the result of transfer; hence the overall information transfer is asymmetric. The importance of oblivious transfer is that it is a basic protocol from which more complicated secure distributed computing protocols can be built and to which all secure distributed computing protocols can be reduced (Kilian, 1988).
- ***Zero-knowledge proofs:*** An interactive proof system is a two-party protocol in which a prover owns a secret and wants to convince a verifier that it really has the secret through interaction with the verifier. An interactive proof system is computationally zero-knowledge if from the interaction a computationally bounded verifier knows nothing about the secret except the validity of the proof. Every language in NP has a computationally zero-knowledge proof system if a one-way function exists (Goldreich, 1997). This fact is very useful in constructing multi-party secure distributed computing protocols that can tolerate active attackers.
- ***Secret sharing protocol:*** A secret sharing protocol, with threshold *t*, enables a dealer to distribute a secret among several players such that each player has an individual share of the secret (called a *t*-share) and coalition of any group of maximum *t* players cannot reconstruct the secret from their shares. A verifiable secret sharing scheme is a scheme that, despite the cheating of a dishonest dealer and some of the players, honest players can still receive valid shares and identify when the dealer cheats. The general idea is to let the dealer distribute primary shares of the secret to all players and each player distributes subshares (called secondary shares) of its primary share to all other players. Inconsistency of primary share and secondary shares of any player would invoke a challenge-response process and all challenge-response reduce to a conclusion whether the dealer is repudiated (more than a certain number of players accuse it of cheating) or upheld (less than a certain number of players accuse it of cheating). Verifiable secret sharing schemes are fundamental in constructing fault-tolerant secure computation protocols.

- **Byzantine agreement:** In the Byzantine agreement problem, each party has an initial bit and wants to find if initial bits held by all parties have the same value. The challenge is to ensure that even when there are parties who are dishonest and act as if they have different initial values, all non-faulty parties are able to draw correct conclusions for their initial values. This challenge can be solved if no more than one-third of the parties are faulty (Lamport et al., 1982). Because of the fault-tolerance it provides, Byzantine agreement protocols are employed as backbones to construct sophisticated fault-tolerant secure computing schemes (Bazzi & Neiger, 1991).

Secure distributed computing protocols may involve only two parties (two-party secure computing) or more (multi-party secure computing). Most two-party secure computing protocols are built on cryptographic assumptions, such as oblivious transfer, and are based on one of the following two techniques (Franklin et al., 1992): In the first approach, one party scrambles the code and its secret input; after scrambling, this party interacts with the second party to transfer the scrambled code and input; the second party then evaluates the scrambled code with the scrambled input values and sends the output to the first party. In the second one, the two parties interact for every logical component of the code in a secure and interactive fashion.

In multiparty secure computing, much attention has been given on fault tolerance, i.e., resilience against active attacks. An active attacker can cause some of the parties to behave against the protocol. Most multiparty secure computing protocols follow three stages. In the input sharing stage, each party distributes shares of its private input to other parties, in the computation stage each party performs the required computation on shares of private inputs and generates the shares of the ultimate result, and, finally, in the output reconstruction stage shares of the final result are combined by individual parties to recover the result. Zero-knowledge proofs and verifiable secret sharing are essential tools for multiparty secure computing. Zero-knowledge proofs enable the misconduct of any party to be detected. Verifiable secret sharing (which enables each party to verifiably share its secret input with all other parties) guarantees that, if a party is caught cheating, the remaining honest parties can reconstruct its private input and simulate messages it would have sent in the later stages of the protocol. Collectively, these two protocols prevent honest parties involved in a multiparty secure computing from suffering.

Secure computing protocols that can withstand passive attackers are called private secure computing protocols. Secure computing protocols that can withstand active attackers are called resilient secure computing protocols. Two-party secure computing protocols are generally private protocols, as some basic

fault-tolerance techniques, such as Byzantine agreement and verifiable secret sharing, used for handling active attacks and requires more than two parties' engagements (Franklin et al., 1992).

### *Instance Hiding*

Instance hiding deals with the problem of computing with encrypted data; therefore, it can be regarded as a special case of secure distributed computing. In instance hiding, there is a weak party (e.g., a DAHS client) who has some secret input and wants to compute a function that requires a large amount of computation resources, with the help of strong parties (e.g., DAHS servers): the weak party sends each strong party its encrypted data, and from the partial results computed by the strong parties it reconstructs the actual output with minimal effort. Instance hiding is especially required in DAHSs, where a computation center enables weak, private computing devices, such as mobile terminals, compute hard problems.

Like many cryptography techniques, most instance hiding schemes are built on hard number theory problems, such as primitive root, quadratic residuosity, and discrete logarithm problems (Abadi et al., 1987).

Not all functions have information theoretic one-server (or one-oracle) instance hiding schemes. For instance, Abadi et al. (1987) prove that no NP-Hard problem has such a scheme. This result is also consistent with the related results that “there exists no information theoretic two-party secure distributed computing protocols for some functions” and that “there exists no information theoretic one-server private information hiding scheme (Subsection Private Information Retrieval) without sending the whole database.”

It has been proven, on the other hand, that for any function, even for NP-Hard functions, there always exists information theoretic multi-server (multi-oracle) instance hiding schemes, as long as servers are forbidden from communicating with each other.

### *Function Hiding*

Function hiding deals with the problem of computing with an encrypted function. Function hiding has crucial DAHS applications: the owner of a secret algorithm can make its code available at a DAHS server in a way that the algorithm of the program and its execution are prevented from disclosure despite intensive code analysis and reverse engineering. In this subsection, we describe various techniques for function hiding:

- ***Matrix modification method:*** This method is specifically for hiding polynomial functions that have matrix representations. Hiding is achieved by modifying the function matrix in a randomized way such that the real

output can be decrypted from the randomized output. Typical examples include Error Correcting Codes (ECCs) modification and similarity transform modification.

ECCs modification technique uses an ECC based cryptosystem, such as McEliece public key cryptosystem (Loureiro & Molva, 1999), to hide matrix of polynomial functions. ECC-based security relies on the difficulty of decoding a large linear code with no visible structure.

Similarity transform modification technique transforms a function matrix,  $F$ , into its similarity matrix  $KFK^{-1}$  and the input  $x$  into  $Kx$ , where  $K$  is a random invertible matrix serving as the secret key. With similarity transform modification technique, the owner can hide a function with a loop structure within which there is a polynomial calculation. The security of this approach relies on the difficulty of finding the secret key (Badin et al., 2003).

- **Decomposition method:** This method hides polynomial functions. By asking the server to evaluate all possible terms of a given polynomial, the client can locally construct the result by selecting and merging the relevant components. This way, the server can not learn the polynomial.
- **Homomorphism encryption:** This method uses a homomorphism encryption function  $E$  such that, it is possible to compute  $E(x+y)$  directly from  $E(x)$  and  $E(y)$  (Sander & Tschudin, 1998).
- **Redundant computation:** This method uses dummy computation and dummy data to hide the real computation and real data.

In general, the types of functions that can be protected via function hiding are very limited, for it is very hard to find encryption schemes for general functions such that the encrypted functions remain executable. Up to now, most function hiding protocols are restricted to hiding polynomial functions.

Note that instance hiding and function hiding are closely related. Theoretically, instance hiding and function hiding problems are equivalent. In some instance hiding schemes, in order to compute with encrypted data, the function is also scrambled in a matching way. If the scrambling hides the original function, function hiding is achieved at the same time. Similarity transform modification technique mentioned above is a good example to this type of hiding. Such symmetric hiding schemes have very important applications in DAHSs: to hide both the private data and code from remote hostile executing environments the owner can publish secret algorithms on untrusted servers and let those servers provide computation service with the privacy of secret input data also preserved.

A more practical and hence more promising way to protect code and data from leakage and tempering is to provide a certified safe executing environment to the secret code (Smith et al., 1999). Such safe executing environments cannot

be inspected or tampered with and, hence, are called temper proof environments (TPEs). A TPE is provided and certified by a trusted manufacturer (TM). The security of TPE cryptographic protocol relies on the trust between the code owner and the trusted manufacturer as well as the security of TPE. To alleviate these requirements, some independent authority can periodically verify and certify the TPE. The TPE is generally a temper-proof hardware; hence, its private keys cannot be leaked without actually destroying it. IBM has a product called “secure co-processor” that can serve as a TPE (Dyer et al., 2001). When it is turned on, the secure co-processor generates public/private key pairs and outputs the public key that can be used for encrypting the code and the data. Since the host does not know the private key to decrypt the code or the data, the only thing it can do is to upload the hidden messages to the TPE, which internally decrypts the code and data before execution.

## Private Information Retrieval

Private Information Retrieval (PIR) is a family of encryption protocols seeking to protect users’ private queries from malicious data hosts. This is a relatively new research area in database security. Traditional database security mainly deals with preventing, detecting, and deterring improper disclosure or modification of information in databases. When the database services are made available through third party hosts that cannot be fully trusted, users’ queries may be improperly utilized to achieve certain malicious goals. For example, users may not trust their queries being executed on a third party stock database server. Through PIR, users can query the untrusted data source while protecting the confidentiality of their queries and the answers.

The basic idea behind any information theoretic PIR scheme is to replicate the database to several non-communicating servers and ask from each copy of the database a subset of the data that is independent of the target data in a way that the user can reconstruct the target data from query results. Chor et al. (1995) show that PIR can be translated into an oblivious transfer problem and that, if one copy of database is used, the only way to hide the query in the information theoretic sense is to send the whole database to the user. In order to reduce even one bit in communication between the server and the user, replication of the whole database is required.

In PIR schemes, communication is the major cost; hence, in order to achieve practical use, it is crucial to reduce the communication cost as well as the number of replicas required. The best-known  $k$ -server scheme requires  $O(n^{2k-1})$  communication (Chor et al., 1995) (where  $n$  is the database size). However, to achieve communication to a subpolynomial in the size of the database, more than a constant number of servers are needed (Chor et al., 1995).

In DAHSs, replication is not a preferable solution to reduce communications in PIR. It is very likely that database owners are reluctant to replicate databases

to other servers that they can not keep in contact. Moreover, it may not be possible to prevent third party servers from communicating with each other. In computationally private PIR schemes, therefore, the user privacy requirement is relaxed so that what the servers can see with respect to any two retrieval indexes are indistinguishable for any polynomial time server. PIR schemes built on cryptographic assumptions can further reduce the communication and the number of replicas. If a one-way function exists, then there is a two-server scheme, such that the communication is  $f$  for any  $\epsilon > 0$  (Chor et al., 1997). Under the quadratic residuosity assumption, a one-server scheme can be constructed with sub-polynomial communication (Kushilevitz & Ostrovsky, 1997). Under the  $\Phi$ -hiding assumption, a one-server scheme with a poly-logarithmic communication is possible (Beimel et al., 1999).

### *Private Informational Retrieval in XML Databases*

As discussed earlier, XML has an inherent tree-like structure. Queries over trees are generally expressed in the form of tree path descriptions. XQuery, the standard for XML query language proposed by W3C, is a typical example. Its core is Xpath, the W3C standard for addressing parts of an XML document by describing path patterns.

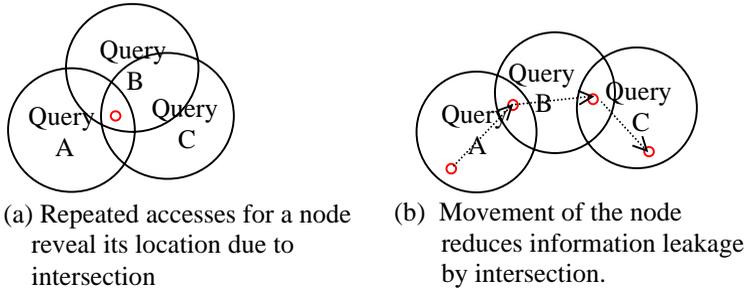
XML documents can be queried in a navigational manner, i.e., traversing documents along paths described by these patterns and performing join operations on the resulting paths when necessary. However, navigation is often inefficient. Using index mechanisms on the XML structures (i.e., path information) or element values and traversing index trees instead of traversing XML document can reduce query processing time.

On the other hand, if the tree (XML tree or index tree) is traversed in plain to identify the matching paths, the query as well as the result is revealed to the server. Thus hiding the traversal of the tree is necessary for hiding data and queries from untrusted XML servers.

Lin and Candan (2003) propose a protocol to hide tree traversal paths. This protocol allows clients to outsource their sensitive data on servers without any prior trust. In this protocol, tree nodes are encrypted before being outsourced; hence, their contents (if the nodes are XML elements, also the element types) are hidden from the untrusted data store. Furthermore, to hide the XML tree structure, each time a user wants to retrieve a node, he or she asks for a set of nodes called the redundancy set including the target node and additional random nodes. The redundancy set hides the requested node from the server. However, repeated accesses for a particular node can reveal its location, since the node is always within the intersection of the redundant sets of these queries. *Figure 7(a)* demonstrates this situation. To prevent the server from inferring the locations of the nodes based on repeated node accesses, after each node is retrieved, the node is swapped with a randomly chosen empty node. Thus the

Figure 7: The intersection property and the movement of the target node

Big circles represent retrieval of a redundancy sets. Small circles denote the target nodes.



node moves with respect to each retrieval, making any correct guessing of its location temporary and of no permanent use. *Figure 7(b)* depicts the movement of target node. In this figure, there are three consecutive queries A, B, C all of which wants to retrieve the same node. As shown in the figure, the protocol moves the location of the target node after each retrieval. It does so without violating the XML tree structure.

For a reasonable security level, the size of the redundancy set need not be large to hide long paths. If the size of the redundancy set is  $m$ , then the probability of finding the target node from a given set is  $\langle U, O, P, R, S \rangle$ , the probability of finding the parent-child relationships in a tree is  $1/m^2$ , and the probability of finding a given path from root to a leaf is  $1/m^{\text{path length}}$ . Hence this protocol is sufficient to hide tree-structured data and queries from any polynomial computation-bounded servers. To enable multiple users to query a tree simultaneously, which is mandatory for an open and public data store, we also devised deadlock free concurrency control mechanisms (Lin & Candan, 2003). Unlike the information theoretic private information retrieval schemes, the proposed technique requires no replication of the database and the communication cost is  $O(m \times \text{tree depth})$  which is adaptable and generally much less than the size of the database. Compared with general computationally private information retrieval schemes, the proposed technique is much simpler and does not rely on any cryptographic assumptions except for the ones on which the underlying encryption schemes are built.

A detailed study of the security guarantees provided by this protocol requires proper modeling of the user queries as well as the interaction between clients and the server. A request for a redundancy set constitutes a call. Each query path retrieval then can be represented as an ordered set of calls. If there

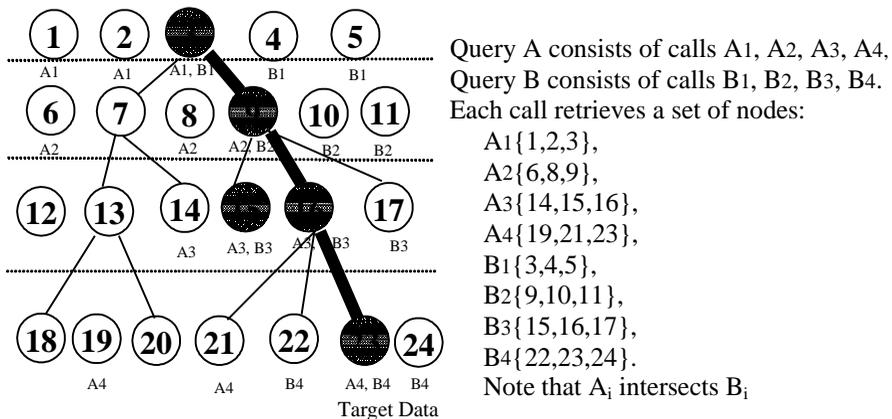
are multiple users accessing the system, calls of concurrent queries can be intermixed. Supposing that there is a transport layer security mechanism that hides the identity of owner of each call, we can model DAHS server's view of data accesses as a stream of calls from unknown origins. The server might still be able to infer the tree structure by (a) observing the call stream it receives, (b) guessing which calls in the stream belong to the a single query, (c) guessing which queries it observes are identical or similar, and then (d) looking at the intersection of the redundancy sets of the corresponding calls in each identical query. That is, the server can analyze the calls and intersections of their redundant sets to learn about the tree structure.

Figure 8 depicts a possible attack. This figure shows the hypothetical case where a query is posed twice consecutively; i.e., without interference from any other queries. In this case, the corresponding calls (e.g., A2 and B2) of the two queries intersect. Hence the query path (depicted by bold black line) is revealed.

In order to measure the degree of security that can be provided by the private information retrieval system, it is necessary to understand the probabilities with which the server can use correlations between queries to break the information retrieval security. The server can use the following analyses to attack the system:

- Given two sequences of calls, the server can try to calculate the likelihood of these two sequences containing identical (or similar) queries.
- Given two sequences of calls that are known to contain two identical queries, the server can try to identify the individual calls of them.
- Given the calls of two identical queries, the server can try to discover the query path.

Figure 8: Consecutive queries for the same data reveal the path

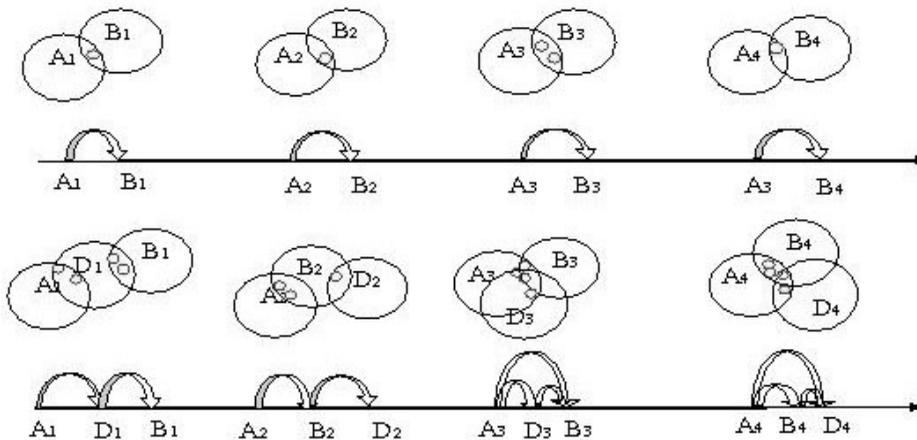


These attacks by the server would rely on the intersection property mentioned above. Intuitively, such an attack by the server can be prevented by ensuring that intersections do not reveal much information. This is achieved by modifying the client/server protocols such that the redundant sets intersect at multiple nodes as well as by inserting appropriate dummy/interfering calls. As shown in *Figure 9*, these methods destroy the relationship between queries and intersecting calls, preventing attackers from exploiting the intersection property.

In *Figure 9*, each axis tracks the time when calls are posed. An arrow from one call to another represents intersection between them. *Figure 9(a)* shows that, when there are no interfering calls, the intersection property can be used by a malicious server to identify the existence of identical queries in a given sequence of calls. In addition, if the sizes of the intersections are small, the server can also learn the actual data nodes. *Figure 9(b)*, on the other hand, shows how by adding dummy calls (D1, ... D4) and by increasing the sizes of the intersections, one can limit the information the server can learn studying the intersection of redundant sets. Intuitively, each  $D_i$  call adds ambiguity and reduces the probability with which the server can identify the calls that correspond to identical queries. Note that in order to provide efficient and provable security, the process of introducing dummy calls have to follow a strategy that randomizes the intersections with minimal overhead.

This private information retrieval scheme requires legal clients to have access to encryption/decryption keys and be able to perform encryption and decryption operations. Where encryption and decryption constitute heavy computation costs for clients with very limited computation power and memory, we suggest the use of assistant hardware, such as smart cards, to reduce the

*Figure 9: Increased size of intersection and dummy/interfering calls reduce information leaked through the intersection of redundant sets*



encryption/decryption execution costs as well as to securely disseminate keys. Chapter 3 of this book details smart card applications.

## CONCLUSION

In this chapter, we provided an overview of the security mechanisms and tools for Distributed Application Host Services (DAHSS) that rent out Internet presence, computation power, and data storage space to clients with infrastructural needs. ASPs and DAHSS operate within the complex, multi-tiered, and open Internet environment and, hence, they introduce many security challenges that have to be addressed effectively. In this chapter, we discussed security challenges in DAHS from three main aspects: authentication, authorization, and confidentiality. For each aspect, we surveyed current techniques and tools to address these security challenges and discussed open research challenges.

## REFERENCES

- Abadi, M., Feigenbaum, J., & Kilian, J. (1987). On hiding information from an oracle. In *Proceedings of the 19th ACM Symposium on Theory of Computing* (pp. 195-203).
- Atkinson, B., Della-Libera, G., & Hada, S., et al. (2002). Specification: Web Service Security (WS-Security), version 1.0.05. Retrieved September 15, 2003, from the World Wide Web: <http://www-106.ibm.com/developerworks/library/ws-secure/>.
- Badin, R., Bazzi, R. A., Candan, K. S., & Fajri, A. (2003). Provably secure data hiding and tamper resistance for a simple loop program. *Proceedings of 2003 AeroSense Technologies and Systems for Defense and Security*, (pp. 21-25).
- Bazzi, R. A., & Neiger, G. (1991). Optimally simulating crash failures in a Byzantine environment. In *Proceedings of the Fifth International Workshop on Distributed Algorithms* (pp. 108-128).
- Beimel, A., Ishai, Y., Kushilevitz, E., & Marikín, T. (1999). One way functions are essential for single-server private information retrieval. In *Proceedings of the 31<sup>st</sup> Annual ACM Symposium on Theory of Computing* (pp. 89-98).
- Bertino, E., Castano, S., & Ferrari, E. (2001). Securing XML documents with Author-X. *IEEE Internet Computing*, 5 (3), 21-31.
- Bertino, E., Catania, B., Ferrari, E., Thuraisingham, B. M., & Gupta, A. (2002). Selective and authentic third-party distribution of XML documents (MIT Sloan working paper No. 4343-02).
- Blaustein, B. T., McCollum, C. D., Notargiacomo, L., Smith, K. P., & Graubart, R. D. (1995). *Autonomy and confidentiality: Secure federated data*

- management*. The 2<sup>nd</sup> International Workshop on Next Generation Information Technologies and Systems.
- Bonatti, P. A., Kraus, S., & Subrahmanian, V. S. (2003). Secure agents. *Annals of Mathematic and Artificial Intelligence*, 37 (1-2) 169-235.
- Bonatti, P. A., Sapino, M. L., & Subrahmanian, V. S. (1996). Merging heterogeneous security orderings. In *European Symposium on Research in Computer Security* (pp. 183-197).
- Butt, A. R., Adabala, S., Kapadia, N. H., Figueiredo, R., & Fortes, J. A. B. (2002). Fine-grain access control for securing shared resources in computational grids. Retrieved September 15, 2003, from the World Wide Web: <http://computer.org/proceedings/ipdps/1573/symposium/15730022babs.htm>.
- Candan, K. S., Jajodia, S., & Subrahmanian, V. S. (1996). Secure mediated databases. In *IEEE Conference on Data Engineering* (pp. 28-37).
- Cho, S., Amer-Yahia, S., Lakshmanan, L. V. S., & Srivastava, D. (2002). Optimizing the secure evaluation of twig queries. In *Proceedings of the 28<sup>th</sup> Very Large Data Bases Conference* (pp. 490-501).
- Chor, B., & Gilboa, N. (1997). Computationally private information retrieval. In *Proceedings of the 29<sup>th</sup> Annual ACM Symposium on the Theory of Computing* (pp. 304-313).
- Chor, B., Goldreich, O., Kushilevitz, E., & Sudan, M. (1995). Private information retrieval. In *Proceedings of 36<sup>th</sup> IEEE Conference on the Foundations of Computer Sciences* (pp. 41-50).
- DB2. (2003). *DB2 relevant information*. Retrieved May 15, 2003, from the World Wide Web: <http://www7b.boulder.ibm.com/dmdd/>.
- Devanbu, P. T., Gertz, M., Kwong, A., Martel, C. U., Nuckolls, G., & Stubblebine, S. G. (2001). Flexible authentication of XML documents. In *ACM Conference on Computer and Communications Security* (pp. 136-145).
- Devanbu, P. T., Gertz, M., Martel, C. U., & Stubblebine, S. G. (1999). *Authentic third-party data publication, 14<sup>th</sup> IFIP 11.3 Conference on Database Security*.
- Dyer, J. G., Lindemann, M., Perez, R., Sailer, R., Doorn, L. V., Smith, S. W., & Weingart, S. (2001). Building the IBM 4758 secure coprocessor. *IEEE Computer*, 34 (10), 57-66.
- Eastlake, D., & Reagle, J., et al. (2003). *W3C XML Signature WG*. Retrieved on September 15, 2003, from the World Wide Web: <http://www.w3.org/Signature>.
- Franklin, M., Galil, Z., & Yung, M. (1992). *An overview of secure distributed computing* (Technical Report, TR CUCS-008-92). Columbia University.
- Fujisaki, E., Okamoto, T., Pointcheval, D., & Stern, T. (2000). *RSA-OAEP is still alive!* Retrieved September 15, 2003, from the World Wide Web: <http://eprint.iacr.org/>.

- Goldreich, O. (1997). On the foundations of modern cryptography. In B. Kaliski (ed.), *Advances in Cryptology Crypto '97* (pp. 46-74). Berlin and Heidelberg: Springer-Verlag.
- Hégaret, P. L., et al. (2002). *Document Object Model (DOM)*. Retrieved September 15, 2003, from the World Wide Web: <http://www.w3.org/DOM/>
- Idris, N. B., Gray, W. A., & Churchhouse, R. F. (1994). Providing dynamic security control in a federated database. In *Proceedings of the 20<sup>th</sup> Very Large Data Bases Conference* (pp. 13-23).
- J2EE. (2003). *J2EE relevant information*. Retrieved September 15, 2003 from the World Wide Web: <http://java.sun.com> on Java security APIs, such as JAAS, JSSE, JCE.
- Jonscher, D., & Dittrich, K. R. (1994). An approach for building secure database federations. In *Proceedings of the 20<sup>th</sup> Very Large Data Bases Conference* (pp. 24-35).
- Karnik, N. M. (1998). *Security in mobile agent systems* (Ph.D. Thesis). University of Minnesota. Retrieved September 15, 2003, from the World Wide Web: <http://www.cs.umn.edu/Ajanta>.
- Kilian, J. (1988). Founding cryptography on oblivious transfer. In *Proceedings of 20<sup>th</sup> ACM Symposium on Theory of Computing* (pp. 20-31).
- Kohl, J., & Neuman, C. (1993). *The Kerberos network authentication service*. Retrieved September 15, 2003, from the World Wide Web: <http://www.faqs.org/rfcs/rfc1510.html>.
- Kushilevitz, E., & Ostrovsky, R. (1997). Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38<sup>th</sup> IEEE Symposium on Foundations of Computer Sciences* (pp. 365-373).
- Lamport, L., Shostak, R., & Pease, M. C. (1982). The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4, 382-401.
- Lin, P., & Candan, K. S. (2003). Hiding traversal of tree structured data from untrusted data stores. In H. Chen et al. (eds.), *Intelligence and Security Informatics* (pp. 385). New York: Springer-Verlag.
- Loureiro, S., & Molva, R. (1999). Function hiding based on error correcting codes. In *Proceedings of Cryptec'99 International Workshop on Cryptographic techniques and Electronic Commerce* (pp. 92-98).
- Maruyama, H., Tamura, H., & Uramoto, N. (2000). *Digest Values for DOM (DOMHASH)*, RFC2803. Retrieved September 15, 2003, from the World Wide Web: <http://www.faqs.org/rfcs/rfc2803.html>.
- Minsky, Y., Renesse, R. V., Schneider, F. B., & Stoller, S. D. (1996). Cryptographic support for fault-tolerant distributed computing. In *Proceedings of the 7th ACM Special Interest Group on Operating Systems European Workshop* (pp.109-114).

- Nace, W. A., & Zmuda, J. E. (1997). *PPP EAP DSS public key authentication protocol*. Retrieved September 15, 2003, from GlobeCom IETF library.
- Naor, M. (1989). Bit commitment using pseudo-randomness. In G. Brassard (ed.), *Advances in Cryptology - Crypto '89* (pp. 128-136). Berlin and Heidelberg: Springer-Verlag.
- Necula, G. C., & Lee, P. (1998). Safe, untrusted agents using proof-carrying code. In G. Vigna (ed.), *Mobile Agents and Security* (pp. 61-91). Berlin and Heidelberg: Springer-Verlag.
- .NET. (2003). *.NET relevant information*. Retrieved September 15, 2003, from the World Wide Web: [www.microsoft.com/net/technical/security.asp](http://www.microsoft.com/net/technical/security.asp), [www.microsoft.com/net/business/security\\_net.asp](http://www.microsoft.com/net/business/security_net.asp).
- Oracle. (2003). *Oracle relevant information*. Retrieved September 15, 2003, from the World Wide Web: <http://otn.oracle.com/deploy/security/content.html>.
- Reagle, J., et al. (2003). *W3C XML Encryption WG*. Retrieved September 15, 2003, from the World Wide Web: <http://www.w3.org/Encryption/2001/>.
- Sander, T., & Tschudin, C. F. (1998). Protecting mobile agents against malicious hosts. In G. Vigna (ed.), *Mobile Agent Security* (pp. 44-61). Berlin and Heidelberg: Springer-Verlag.
- Sandhu, R. S., & Samarati, P. (1994). Access control: principles and practice. *IEEE Communications Magazine*, 32 (9), 40-48.
- Smith, S. W., Perez, R., Weingart, S., & Austel, V. (1999). *Validating a high-performance, programmable secure coprocessor*. 22<sup>nd</sup> National Information Systems Security Conference.
- Vigna, G. (1998). Cryptographic traces for mobile agents. In G. Vigna (ed.), *Mobile Agents and Security* (pp.137-153). Berlin and Heidelberg: Springer-Verlag.
- Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25 (3), 38-49.
- Wiederhold, G. (1993). Intelligent integration of information. *Proceedings of the ACM Special Interest Group on Management of Data International Conference on Management of Data*, 22 (2), 434-437.
- Wright, T. (2000). *Proposal for WAP-IETF co-operation on a wireless friendly TLS*. Retrieved September 15, 2003, from the World Wide Web: <http://www.ietf.org/proceedings/00jul/SLIDES/tls-wtls/>.
- XML Security Suite. (1999). Retrieved September 15, 2003, from the World Wide Web: <http://www.alphaworks.ibm.com/tech/xmlsecuritysuite>.
- Yee, B. S. (1999). A sanctuary for mobile agents. In J. Vitek & C. Jensen (eds.), *Secure Internet Programming: Security Issues for Distributed and Mobile Objects* (pp.261-273). Berlin and Heidelberg: Springer-Verlag.
- Yu, T., Srivastava, D., Lakshmanan, L. V. S., & Jagadish, H. V. (2002). Compressed accessibility map: efficient access control for XML. In

*Proceedings of the 28<sup>th</sup> Very Large Data Bases Conference* (pp. 478-489).

## Chapter IX

# Comparing the Security Architectures of Sun ONE and Microsoft .NET

Eduardo B. Fernandez, Florida Atlantic University, USA

Michael Thomsen, Florida Atlantic University, USA

Minjie H. Fernandez, Florida Atlantic University, USA

## ABSTRACT

*Platforms for web services have been reduced to two basic approaches: Microsoft .NET and Sun ONE (J2EE). We compare here these two platforms with respect to the security they provide to the web services that use them. We arrive to the conclusion that although the basic security architectures are fairly similar, their actual implementations differ. Microsoft's approach appears weaker because of their self-contained approach, and a failure to follow good principles of software and secure systems design.*

## INTRODUCTION

Several companies have announced strategies for supporting web services. They all use one of two basic architectures: Microsoft's .NET or Sun ONE.

Microsoft's architecture is also an implementation, while Sun ONE is an architecture with several implementations based on Java. Comparisons of these architectures barely mention security (Mougin, 2001; Sessions, 2001). We believe this aspect is one of the most fundamental factors in their success or failure and we look here at the security features in Microsoft .NET and Sun ONE web services architectures.

The basic purposes of an architecture to support web services are:

- Store the web service programs on shared systems, where users can find and access them.
- Some systems also need to store the user's data on these shared systems.
- Some systems store web services repositories or catalogs.

This brings up the typical security issues, which now have the following objectives:

- *Confidentiality*: The web service data may only be accessed by authorized users.
- *Integrity*: Web services data, code, or descriptions may only be altered by authorized users.
- *Code control*: The program code must not be able to perform illegal actions when invoked.
- *Access control*: Only paying subscribers can use the service.
- *Availability*: Code and data stored on the server must be available when needed.

Some of these issues should be resolved in the upper layers, where web services are defined. These layers are based on standards under development and we do not discuss these aspects here because they apply to both architectures. A survey of security aspects of web services is given in Fernandez (2002).

Because both the Microsoft .NET and Sun ONE architectures are quite new, there are many aspects still not tested in practice. These two platforms are still evolving and some of our specific conclusions may not be true anymore, but unless there is a change in their fundamental design our main conclusions should still hold.

The next section provides a general overview of both architectures. We then consider in detail the security architectures of Sun ONE and .NET, relating them to the standard work on security models. We end with a general discussion of both approaches.

## A GENERAL COMPARISON OF MICROSOFT .NET AND SUN ONE

Both the Sun and the Microsoft frameworks are based on specific programming languages, object models, and virtual machines, but the design approaches of their runtime environments are quite different. We summarize here a few aspects; several papers compare them in detail (Farley, 2001; Mougín, 2001; Sessions, 2001; Vawter, 2001).

Sun ONE is based on Sun's J2EE (Java 2 Enterprise Edition) specification, which implies that you'll have to use Java as programming language, but the programs can (in theory) run on any platform without recompiling. The Microsoft .NET framework builds on Microsoft's Common Object Model (COM), that has been improved significantly, including additions such as ActiveX, distributed computing, database connections, etc. Microsoft has also produced a new programming language, C# and a new runtime environment called Common Language Runtime (CLR), that provides garbage collection and some security features similar to the Java Virtual Machine (JVM).

J2EE accesses data sources using the JDBC API, which requires vendor specific drivers. These are available for most commercial and open-source databases. Another approach to persistent objects is Java Data Objects (JDO, 2003). Java uses JNDI (Java Naming and Directory Interface) in order to access directory services. Support for various directory services, such as Novell NDS and Directory Services Markup Language (DSML), among others, is available. Some XML data sources use drivers that are similar to the JDBC drivers.

Microsoft's first data access service was DAO (Data Access Objects), that later led to ActiveX Data Objects (ADO), which provides access to both relational data and directory services. The .NET framework with ADO.NET improves on some of the limitations of ADO (e.g., lack of scalability). Another change is that the data transmission format has been replaced by XML, which gives the opportunity for components to act as data sources for each other.

Although Sun ONE is based on Java, which is said to be platform independent, some implementations are not exactly cross-platform independent. On this part, Microsoft has made some attempts to make the .NET framework platform independent, such as submitting the CLR specification and the C# language for standardization, but more realistically, Sun ONE is dependent on Java, and Microsoft .NET incorporates Windows as a central unit. Furthermore, the .NET language independence is somewhat incomplete, since you'll have to add some extensions to the languages in order to make them comply with the CLR requirements, thereby making them COBOL#, Eiffel#, Perl#, etc.

Microsoft has been very active in the XML field for the past few years, and .NET provides strong XML support. As mentioned earlier, XML is used for data transmission, but also for general-purpose document handling. Web services can

Table 1: .NET and ONE side by side

	Microsoft .NET	Sun ONE
Programming Language	Any (but extensions needed)	Java
Runtime environment	CLR, code is compiled into native code before execution	JVM, Java bytecodes, either interpreted or compiled on-demand
Component sharing	IL, CLR	JVM with Corba and ORB
Data interchange protocol	XML, SOAP on HTTP	XML, SOAP on HTTP

be exported as SOAP interfaces (Simple Object Access Protocol). Sun has also been active in the XML field, and XML has been incorporated into their framework. *Table 1* shows a basic comparison of these approaches.

## SUN ONE SECURITY ARCHITECTURE

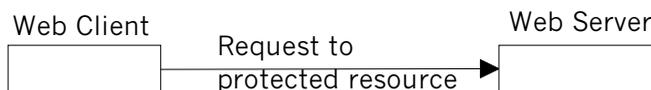
### Overview

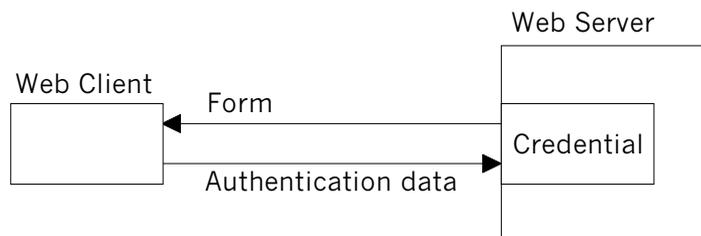
Let's take a look at a service request. A request for a protected resource goes through standard Internet protocols, such as HTTP (*Figure 1*).

The web server detects that the new request has not yet been authenticated so the appropriate authentication mechanism for the resource is invoked. This may be done by returning a form where the user can fill out a user name and a password, but there are several other options (see below). The data is transferred to the web server that validates the data, and sets the credential for the user (*Figure 2*).

Once the web server decides that the request is authorized, it consults the authorization rules (called *security policies* by Sun) associated with the resource. Next, the web server container tests the user's credentials against each role in the authorization rules. This process will either stop with an

*Figure 1: A service request*



*Figure 2: Client authentication*

“authorized” or a “not authorized” outcome depending of the web container being able to map the user to any of the permitted roles. If the user is authorized, the web server returns the result of the original request (*Figure 3*).

Later on, the user might perform some action that needs to be handled by an EJB component. When a JSP page performs a remote call to the EJB component, the user’s credential is used to establish a secure association between the JSP page and the EJB component (*Figure 4*). At this point it is the responsibility of the EJB containers to enforce the access control on their bean methods. This is done by consulting the authorization rules associated with the bean, and an “is authorized” or “not authorized” outcome is generated. EJBs may access databases, which may enforce additional security constraints, e.g., content-dependent access control, where access depends on specific rules in the databases.

The following sections describe in detail these actions.

## Authorization

Security in J2EE is based on roles; they use a type of Role-Based Access Control (RBAC) model. In an RBAC model users are assigned to roles and roles are given rights for resources based on their functions, e.g., administrator, manager, developer. RBAC corresponds to a specialization of the access matrix model (Summers, 1997). In the access matrix model, subjects (requesting entities) are authorized to access specific protection objects in specific ways

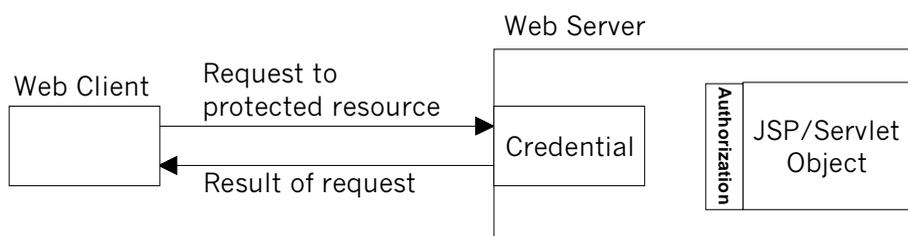
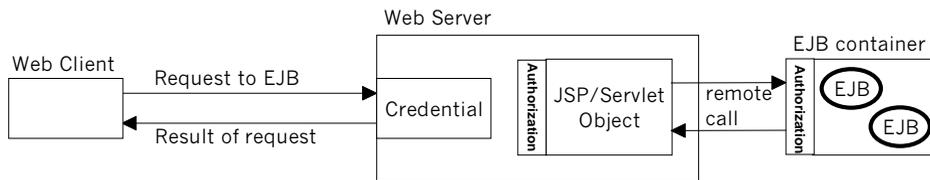
*Figure 3: Result of authorized request*

Figure 4: Application object access



(access types or access modes). The set of authorization rules defines the subjects' privileges and implements the institution security policies.

One of Sun's design goals when they created the J2EE platform was to separate the security aspects of the development of components, the assembly of these components into applications, and the deployment of these applications. Several roles have been defined: the *Component Provider* and *Application Assembler* specify which parts of an application require security, and the *Deployer* selects the specific security mechanisms used for that protection. This security is specified, among other information, in the deployment descriptor that is written in XML (Koved, 2001).

J2EE introduced the JAAS (Java Authentication and Authorization Service) to enforce server side security. Enforcement of security is based on two approaches: declarative and programmatic security. *Declarative security* is based on authorization rules defined in a J2EE system by the deployment descriptor. This is a contract between the Application Component provider and the Deployer. Groups of components can be associated with one deployment descriptor. When declarative security is not sufficient to express the security constraints of the application, the *programmatic security* approach can be used instead. It consists of four methods, which allow the components to make decisions based on the security role of the caller.

The trade-offs between the declarative and programmatic security approaches are: The former is more flexible after the application has been written, and usually more comprehensible, and therefore results in fewer bugs. More important, it is done in a uniform way across the system. The latter could provide more functionality when the application is being written, but is buried in the application and is therefore difficult to change, and usually only fully understood by those who developed the application. Programmatic security does not allow checking for compliance with institution policies and alone is not acceptable for systems that require a high level of security (1997). However, it could be useful to complement declarative security by adding special restrictions for application access, although it doesn't appear that it can be combined with declarative security in J2EE.

We have already mentioned that the J2EE authorization is based on roles. These roles are defined by the Application Component Provider, and are used by both the declarative and programmatic security approaches. If necessary each method in each bean can be assigned different permissions, i.e., this is a type of method-based access control (Fernandez, 1994). The Component Provider specifies what roles are being used by the application, a list of external resources accessed by the components, and references to all inter-component calls. A method permission model and information that identifies the sensitivity with respect to privacy of the information exchanged may also be provided. When the Application Assembler combines the components into an application, he has to create a consistent security view for the application as a whole. Finally, the Deployer is responsible for securing the application in the specific operational environment. The Application Assembler should supply the Deployer with information on which method parameters or return values require protection; this information is added to the deployment descriptor. It would be nice if the deployment tools included message security. This would allow the application assembler to decide which channels require integrity and/or confidentiality checks. Later, tools could be used by the Deployer to choose the best algorithm to solve the problem. It is important to note that the Component and Application Developers define the roles and the Deployer assigns users to the different roles. Several tools are currently available to help the Deployer configure the security mechanisms. These security mechanisms are implemented by the containers on behalf of the components hosted by the containers. As indicated earlier, J2EE components can access persistent data stored in relational databases. It is important that any authorization defined in a component is consistent with authorizations defined in the database system.

In the J2EE model, the deployment descriptor defines the security roles and associates these roles with the components in order to define the granted permissions. If a JSP/Servlet resource is not associated to any roles, permission to access the resource will be granted to all. This is risky business, and doesn't comply with the principles of closed systems and least privilege, fundamental for secure systems (Summers, 1997). The EJB components must, on the other hand, have associated at least one role to them. If unrestricted access is needed to an EJB component you'll have to map a role to the component which is permitted access to the resource without authentication. This way you cannot leave a component exposed to everyone because you forgot to assign a role to it.

Other authorization problems can occur if method level access control is applied to a component, because a less protected method could be used to undermine the policy enforced by a more rigorously protected method. For example, a method that allows to read a value directly. It is therefore recommended to partition the components that need different authorization rules, so that all methods of each component enforce the same guidelines. Actually, if one

applies properly the RBAC model, defining roles from use cases (Fernandez, 1997), this should not happen. Each use case defines the least amount of privilege needed by each actor to perform its job. Finally, when using XML documents, these may have their own authorizations, defined in terms of XACML or similar models. Mappings between these levels are needed to provide a consistent enforcement of the defined authorizations (Fernandez, 1999).

## Authentication and Other Aspects

The J2EE specification addresses two different client types for authentication, namely a web client, and an application client. We will focus only on the web client in this chapter. There are three possible mechanisms to authenticate a web client:

- **HTTP Basic Authentication.** This is an insecure authentication method, since the passwords are encoded using base64 encoding.
- **HTTPS Client Authentication.** This is a strong authentication mechanism that allows mutual authentication. It is based on X.509 certificates and requires the user to possess such a certificate. The authentication occurs over a channel that is protected by SSL.
- **Form-based Authentication.** Lets developers customize the authentication user interface presented by the web browser using standard HTML or JSP/Servlet based forms.

It is recommended by Sun to perform client authentication only when the client tries to access a protected resource. This is a type of lazy authentication, and is convenient for the user. When a user has identified himself, the login session will be maintained so that the user can access multiple applications for which the user has the necessary rights. The session state is stored on the server, and the client keeps references of the state either by URL rewriting or cookies. It is possible for the client to manage its own authentication context without using cookies if SSL is used. In a secure system, all resources should be explicitly protected, so lazy authorization is not a good approach in general.

EJB components need to be authenticated when they communicate with the enterprise data logic (e.g., an ERP system or a database) that is usually located in another security domain. The J2EE Connector Architecture Specification describes this issue in further detail. The following two mechanisms are required for the J2EE implementation, whereas the latter three are only recommended:

- **Configured Identity.** The principal and authentication data are specified at deployment time, and are therefore independent of the principal of the caller.

- **Programmatic Authentication.** The principal and authentication data are specified at runtime using APIs appropriate to the resource. The data can be obtained from the components' environment or as parameters.
- **Principal Mapping.** The resource principal is determined by mapping data from the security attributes from the calling principal.
- **Caller Impersonation.** In this scenario the resource principal acts on behalf of the caller's principal. This requires that the caller's identity and credentials are delegated to the underlying resource manager (e.g., an ERP system or a database).
- **Credentials Mapping.** This is used when the component and resource support different authentication domains. An example could be that a certificate used to identify the user could be mapped to some credentials for the resource.

Sun has joined the Liberty Alliance project (Liberty Alliance, 2002), a federated identity infrastructure, with a first phase that intends a web-based Single-Sign-On (SSO).

One way to ensure message integrity is to attach a signature to the message. By including a time-stamp and a serial number you can further ensure that the message cannot be re-sent by a third party. Finally, encryption can prevent anybody from reading the message — provided that the encryption mechanism is strong enough. It is the Deployer's responsibility to define the necessary security measures used, and it is the container's responsibility to enforce them. The way to do this is by appending the signature to outgoing messages and verifying signatures and timestamps on incoming messages, as well as notification of the caller in case of failure.

The Sun ONE specification describes some facilities for auditing (Sun, 2002). It is, however, the Deployer who is responsible for configuring the security mechanisms that are applied to the enterprise containers, and it is therefore also the Deployer's responsibility to apply proper auditing. Some general guidelines about what to audit and how to protect the audit data are mentioned in Sun (2002).

As indicated, Sun ONE is only an architecture, and there may be different implementations of it. Effective security will strongly depend on the specific underlying platforms, e.g., the web HTTP server, the application server, the operating system, and even the hardware. Sun's ONE server and IBM's WebSphere have good security reputations as application servers. They both use Apache as HTTP server, another strong component. Sun uses Solaris and IBM uses a variation of Unix (AiX) or Linux, which are reasonably strong operating system architectures. In other words, the security platforms normally used for Sun ONE appear strong based on their general architectures and their track record.

## SECURITY IN MICROSOFT .NET

Several security functions in .NET are similar to their counterparts in Sun ONE and we don't repeat them here. We concentrate instead on those aspects that are different. Further details of the .Net security architecture can be found in (LaMacchia, 2002).

As indicated earlier, the web services provided by Microsoft .NET are accessed over the Internet through HTTP, SOAP, and XML. The Common Language Runtime (CLR) implements the new code access security, including controlling that code cannot perform unauthorized actions. The security policy is managed and the security checks are performed when the code is loaded and executed. User access control is managed by the web server or the operating system that controls the server (*Figure 5*).

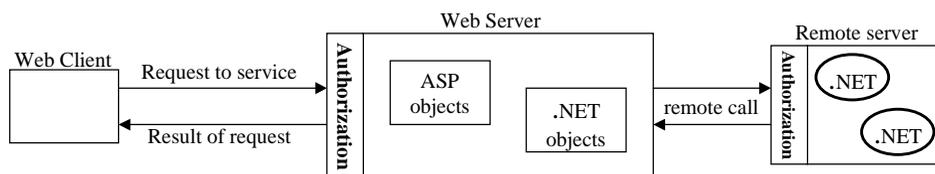
The code access security is used by the programmer to specify which permissions their code requires, and for the user to secure his system from malicious code. These permission requirements are stored in the components at the assembly level. Also programmatic security checks can be coded into the components if necessary, and as in the EJB components it is possible to have method-level access control.

When the code is loaded, the runtime library verifies that the code can be granted the permissions it has requested. Policies for granting permissions on the client are established by system administrators. One interesting feature is that the component and the security manager can negotiate the security level. For instance if a component wants to use file I/O, but can execute without it, it will request file I/O permissions. If these permissions are denied, the component can still execute, but without accessing files. This appears dangerous from a security perspective, a rogue program could try to access different resources until it succeeds.

User access also uses RBAC. Roles can be defined at development or deployment time. At runtime the identity of the user on whose behalf the code is running is determined, and access is granted or denied based on those roles. These roles are typically mapped to credentials in Microsoft's Active Directory.

Role access in .NET components can be inherited from process to component to interface to method. This is an example of Implied Authorization

*Figure 5: Accessing objects in .NET*



(Fernandez, 1975), where access to a composite implies access to components; in particular, this can be interpreted as inheritance of authorizations along a generalization hierarchy (Fernandez, 1994). In .NET higher-level-defined accesses override lower-level rights (Lowy, 2001), an opposite policy to the one defined in Fernandez (1994). For example, I may want to give access to the whole Student class to somebody but not allow that person to see the grades of specific students, something that cannot be done in .NET. A lower-level access is a more precise (finer) specification and it should override the higher and coarser access.

If you know exactly which computers should be allowed access to the application then the Internet Protocol Security (IPSec) or a firewall can be used to restrict access. Since this is not possible in most scenarios Microsoft suggests that you move this authorization to the protocol used to exchange messages. If you're sending and receiving messages using SOAP over HTTP you can use the authentication features available for HTTP. Microsoft's Internet Information Services (IIS) provide several authentication mechanisms for HTTP. This includes support for HTTP Basic, HTTP Basic over SSL, and Client Certificates. An additional authentication mechanism is the *Integrated Windows authentication*, which uses a proprietary password-hashing algorithm and requires both client and server to be Windows systems. Windows authentication is based on Kerberos, but cannot be used over a proxy server or a firewall. IIS 6.0 will also include Passport as authentication approach, although there are doubts about its security (Opplinger, 2003).

Another authentication method is based on custom security mechanisms built by the developer. It is possible to use the SOAP body to communicate the credentials. Since it is difficult to receive the data from the SOAP header, Microsoft does not recommend using the header for this purpose. One drawback of using the SOAP body is that you'll have to make the debugging yourself. Another drawback is that the SOAP messages are sent over HTTP, hence all data is sent in clear text. So, if this technique is used, you should definitely use SSL to encrypt the messages between the client and server. The main issue about using SSL is that it is a lot slower than HTTP itself. Therefore Microsoft suggests that you only encrypt part of the message body, and use digital signatures to ensure that the body has not been tampered with in transit.

The .NET framework includes cryptographic functions for encryption, digital signatures, and hashing. The supported algorithms includes: RSA and DSA for asymmetric encryption, and AES, DES, Triple DES, and RC2 for symmetric encryption. The implementation uses a stream-based model suitable for networking (MS, 2001). The XML Digital Signature Specification (XMLDSIG) is now also supported. This protocol can be used in combination with SOAP to ensure the integrity of the messages. This applies also to Sun ONE since they are following the same standards.

The current .NET framework specification does not say anything about auditing. We assume Microsoft expects you to use the auditing features included in Windows 2000, or to code your own.

Microsoft's server IIS has shown many weaknesses because of its complexity and poor design. This makes it probably the weakest component in the architecture now. Windows 2000 also appears to have a variety of security problems, in particular, in its Active Directory. Security improvements in IIS 6.0 may correct these problems, we have to wait and see.

## CONCLUSION

It turns out that the two platforms are remarkably similar when it comes to their general security architectures. Both use RBAC as authorization model, both support "deployment descriptors," method level access control on the component code, and declarative as well as programmatic access control to the components. Finally the two frameworks include an extended sandbox model for the executing code. In both systems, roles do not enforce content-dependent authorization, this aspect is delegated to the DBMS. An interesting feature of Microsoft is inheritance of rights, although this idea is not used in the most secure way.

Both frameworks also lack some security features. Sun's JSP objects are left unprotected if the deployer forgets to assign a role to them (open system policy). Fortunately, the EJB components do not suffer from that, so the situation can be controlled. Finally, auditing is not clearly integrated with the architectures: Sun relies on the J2SE logging facilities, while Microsoft delegates this function to the operating system. It would have been nice if both had chosen to use XML for writing the auditing log. This would allow a lot of log-tracking and statistical programs to work with both frameworks without much additional coding.

Their main differences are in their approach to authentication, their lower levels, and their general approach for developing software. For authentication, Sun follows the norms of the Liberty Alliance while Microsoft follows its own path. It is not clear now which approach is stronger but the insistence of Microsoft in using its own standards is not good for security. With respect to their lower levels, one needs to weigh Windows against Solaris or AiX as secure platforms. Solaris and AiX appear to have the edge here, although the security architecture of Windows appears well thought. A larger difference is in their approach to software development (Most of these defects are also present in other vendors, but to a lesser extent):

- No use of object-oriented programming. Most (or all) of their systems code has been done using procedural methods. Without proper software design, it is almost impossible to build dependable complex systems.

- Lack of modularity. As a marketing strategy, Microsoft builds systems that cannot be decomposed, i.e., they lack well-defined interfaces. This prevents checks across the interfaces, and vulnerability in one place propagates its effects to other places.
- No use of secure system design principles. These were formulated as early as 1975 (Saltzer, 1975) and include aspects such as open design, fail-safe defaults, and similar. Microsoft's approach to develop secure code is to look for specific vulnerabilities in the code, a very difficult task.

It is important to mention again that Sun ONE is an architectural specification and not a product. Anyone can create an implementation of the Sun ONE environment on any system. Sun has developed an implementation for their own Solaris operating system and Sun ONE server, while IBM, BEA, and Oracle, among others, have implemented (at least parts of) the specifications on other systems. This means that the effective level of security in each of these systems can be different from each other.

Another aspect, not quite clear yet, is how their security approaches match the security defined at the web services level (Fernandez, 2002). Standards for these levels are still being developed and there is no experience with their use.

## REFERENCES

- Farley, J. (2001). *Microsoft .NET vs. J2EE: How Do They Stack Up?* Retrieved from the World Wide Web: [http://java.oreilly.com/news/farley\\_0800.html](http://java.oreilly.com/news/farley_0800.html).
- Farley, J. (2001a, March). *Picking a Winner: .NET vs. J2EE*. Retrieved from the World Wide Web: <http://www.sdmagazine.com/articles/2001/0103/0103a/0103a.htm>.
- Fernandez, E.B. (1999) Coordination of security levels for Internet architectures. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*(pp. 837-841).
- Fernandez, E.B. (2002, September). Waltzing through port 80: Web services security. *Software Development*, 4-7, 14-15.
- Fernandez, E.B., & Hawkins, J.C. (1997, November). Determining role rights from use cases. In *Proceedings of the 2<sup>nd</sup> ACM Workshop on Role-Based Access Control* (pp. 121-125).
- Fernandez, E.B., Gudes, E., & Song, H. (1994, April). A model for evaluation and administration of security in object-oriented databases. *IEEE Trans. on Knowledge and Database Eng.*, 6 (2), 275-292.
- Fernandez, E.B., Summers, R. C., & Lang, T. (1975). Definition and evaluation of access rules in data management systems. In *Proceedings of the 1st*

- International Conference on Very Large Databases*, Boston (pp. 268-285).
- JDO. (2003). Java Data Objects. Retrieved from the World Wide Web: <http://access1.sun.com/jdo/>.
- Koved, L., Nadalin, A., Nagarathan, N., Pistoia, M., & Schrader, T. (2001). Security challenges for Enterprise Java in an e-business environment. *IBM Systems Journal*, 40 (1), 130-152.
- LaMacchia, B.A., Lange, S., Lyons, M., Martin, R., & Price, K.T. (2002). *.NET framework security*. Addison-Wesley.
- Liberty Alliance. (2002). Liberty Alliance Project. Available on the World Wide Web: <http://www.projectliberty.org>.
- Lowy, J. (2001, April). COM+ (and .NET) make security a joy. Presented at Software Development West, April 2001.
- Mougin, P., & Barriolade, C. (2001, October). Web Services, Business Objects and Component models. Retrieved from the World Wide Web: [http://www.orchestranetworks.com/en/docs/0105\\_whitepaper.cfm](http://www.orchestranetworks.com/en/docs/0105_whitepaper.cfm).
- Microsoft Corp. (2001). About .NET security. Retrieved from the World Wide Web: [http://www.getdotnet.com/team/clr/about\\_security.aspx](http://www.getdotnet.com/team/clr/about_security.aspx).
- Opplinger, R. (2003, July). Microsoft .NET Passport: A security analysis. *Computer*, p. 29-35.
- Saltzer, J., & Schroeder, M.D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63 (9), 1278-1308. Retrieved from the World Wide Web: <http://web.mit.edu/Saltzer/www/publications/protection/index.html>.
- Sessions, R. (2001, March 28). *Java 2 Enterprise Edition (J2EE) versus the .NET platform: Two visions for eBusiness*. (White paper). ObjectWatch, Inc. Retrieved from the World Wide Web: <http://www.objectwatch.com>.
- Summers, R.C. (1997). *Secure computing: Threats and safeguards*. McGraw-Hill.
- Sun. (2002). Sun ONE Architecture Guide. Retrieved from the World Wide Web: <http://www.sun.com/software/sunone/docs/arch/index.html>.
- Vawter, C., & Roman, E. (2001, June). *J2EE vs. Microsoft .NET—A comparison of building XML-based web services*.

# About the Authors

**Mariagrazia Fugini** is a professor of Computer Engineering at Politecnico di Milano, Italy. She received her Ph.D. in Computer Engineering in 1987 and was a visiting professor at University of Maryland, Technical University of Vienna and Technical University of Stuttgart. She teaches Fundamentals of Computer Science and Information Systems. Her research interests are in information system security and development, software reuse, information retrieval, information systems development and re-engineering. She participated in the TODOS, ITHACA, F3, WIDE and DEAFIN UE Projects, working on information system development and re-engineering, software reuse, data security, information retrieval tools, and workflow and web application design. She participates in, and coordinates, UE and national Projects on security, web based information systems for public administrations, in particular on web services to citizens and enterprises. She is co-author of the books *Database Security* (Addison-Wesley, 1995) and *Sicurezza dei Sistemi Informatici* (in Italian, Apogeo 2001). She cooperates with Public Administrations in the Italian E-Government Plan.

**Carlo Bellettini** is an associate professor of Software Engineering at the University of Milano, Italy. He received his Ph.D. in Computer Science in 1998 from the University of Milano and Laurea degree in Electronic Engineering in 1992 from the Politecnico di Milano. He teaches Software Engineering, Operating Systems, and Distributed Systems. His main research interests are in software reuse, real-time system specification and design, reverse engineering

and testing of web application. He participated in several UE Projects. Inside the project EXaMINE (Experimentation of a Monitoring and control system for managing vulnerabilities of the European Infrastructure for Electrical power exchange) he worked on dependability aspects of critical infrastructures.

\* \* \*

**Diana Berbecaru** is a research assistant with the TORSEC security group at the Politecnico di Torino. She holds a “laurea” (a.k.a., master’s) in Computer Science from the University of Craiova (Romania) and a Ph.D. in Computer Engineering from the Politecnico di Torino (Italy). Her research interests are in the fields of computer and network security, with special emphasis on digital signatures and e-government.

**Elisa Bertino** is professor of Database Systems in the Dipartimento di Informatica e Comunicazione of the Università degli Studi di Milano (Italy) where she is currently the chair of the department. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, at the Microelectronics and Computer Technology Corporation in Austin, Texas, at George Mason University, at Rutgers University, at Purdue University, and at Telcordia Technologies. Professor Bertino has published several books and more than 200 papers in all major refereed journals, and in proceedings of international conferences and symposia. She is member of the advisory board of the IEEE Transactions on Knowledge and Data Engineering and a member of the editorial boards of several scientific journals, including *ACM Transactions on Information and System Security*, *IEEE Internet Computing*, *the Journal of Computer Security*, *Data & Knowledge Engineering*, and the *International Journal of Information Technology*. Professor Bertino is a senior member of IEEE and a member of ACM and has been named a Golden Core Member for her service to the IEEE Computer Society.

**Guido Bertoni** received the Dr.Eng. in Computer Engineering from the Politecnico di Milano in 1999. Since 2000, he has been a Ph.D. student at the Politecnico di Milano, Italy. The topic of the Ph.D. program is the efficient implementation of cryptographic algorithms in embedded systems.

**Athman Bouguettaya** is the program director of Computer Science and the director of the E-Commerce and E-Government Research Lab at Virginia Tech, USA. His research interests are in web databases and web services. He is on the editorial board of the Distributed and Parallel Databases Journal, and he was a guest editor for a special issue of *IEEE Internet Computing on Database Technology on the Web*. He can be contacted at: athman@vt.edu.

**K. Selçuk Candan** is an associate professor at the Department of Computer Science and Engineering at the Arizona State University, USA. He joined the department in August 1997, after receiving his Ph.D. from the Computer Science Department at the University of Maryland at College Park. His research interests include development of models, indexing schemes, replication mechanisms, and retrieval algorithms for multimedia and Web data; dynamic quality-aware content caching and replication for multi-tiered information infrastructures; and development of novel security mechanisms for application and data services. He has published various articles in respected journals and conferences in related areas. He received his B.S. degree, first ranked in the department, in Computer Science from Bilkent University, Turkey (1993).

**Barbara Carminati** is a Ph.D. student at the Department of Computer Science of the Università degli Studi di Milano, Italy. Barbara Carminati received an M.S. in Computer Sciences from of the University of Milano (2000). Her main research interests include: database security, XML, XML security, secure information dissemination and publishing. She is also a lecturer at the Computer Science School of the University of Milano and has given industrial courses on topics such as database systems and security. She has a wide industrial experience in the area of database systems for customer care relationship systems.

**Pierre Cotte** has been working in semiconductors business for more than seven years. He is currently working for STMicroelectronics as Technical Marketing Manager. He previously worked for NEC Electronics and before that, he developed embedded software for medical system. He is graduated from the Ecole Polytechnique of Joseph Fourier University, Grenoble (France).

**Ernesto Damiani** holds a Laurea Degree in Ingegneria Elettronica from Università di Pavia and a Ph.D. degree in Computer Science from Università di Milano (Italy). He is currently a professor at the Department of Information Technology of the University of Milan. His research interests include distributed and object oriented systems, semi-structured information processing and soft computing. He is vice-chair of the *ACM Special Interest Group on Applied Computing (SIGAPP)*. He is the author, together with I. Sethi and R. Khosla, of the book *Multimedia MultiAgent Systems* (Kluwer, 2001).

**Sabrina De Capitani di Vimercati** is an associate professor at the Department of Information Technology of the Università degli Studi di Milano, Italy. She received her Laurea and Ph.D. degrees in Computer Science from the University of Milan in 1996 and 2001, respectively. Her research interests are in the areas of information security, databases, and information systems. She has been

an international fellow in the Computer Science Laboratory at SRI, CA (USA). She is co-recipient of the ACM-PODS '99 Best Newcomer Paper Award. The URL for her web page is <http://www.dti.unimi.it/~decapita>.

**Corrado Derenale** is a Ph.D. candidate with the TORSEC security group at the Politecnico di Torino (Italy) and he holds a Laurea (a.k.a., master's) in Computer Engineering from the same institution. His research interests are in the security of computer networks, with special reference to wireless environments.

**Eduardo B. Fernandez** (Eduardo Fernandez-Buglioni) is a professor in the Department of Computer Science and Engineering at Florida Atlantic University in Boca Raton, Florida (USA). He has published numerous papers on authorization models, object-oriented analysis and design, and fault-tolerant systems. He has written three books on these subjects. He has lectured all over the world at both academic and industrial meetings. He has created and taught several graduate and undergraduate courses and industrial tutorials. His current interests include patterns for object-oriented design and web services security. He holds an M.S. in Electrical Engineering from Purdue University and a Ph.D. in Computer Science from UCLA. He is a senior member of the IEEE, and a member of ACM. He is an active consultant for industry, including assignments with IBM, Allied Signal, Motorola, Harris, Lucent, and others. He is also a frequent proposal reviewer for NSF. For more information, visit <http://www.cse.fau.edu/~ed>.

**Minjie H. Fernandez** received her B.S. from Fudan University, Shanghai, China (1986). She received her M.S. in Computer Engineering at Florida Atlantic University, Boca Raton, Florida (1992). She is the author of several publications and has worked at Motorola, Siemens, and Rave L.L.C.

**Eduardo Fernandez-Medina** holds a Ph.D. in Computer Science from the Castilla-La Mancha University, and an M.Sc. in Computer Science from the Sevilla University. He is assistant professor at the Escuela Superior de Informática of the Castilla-La Mancha University (Ciudad Real, Spain) and author of several books and papers on security in databases and information systems. He participates in the ALARCOS research group of the Department of Computer Science at the University of Castilla-La Mancha. He is member of the ISO/IEC JTC1/SC27, and IFIP WG11.3. His research interests include secure information systems, including databases, datawarehouses, multimedia documents and DRM. He can be contacted at: [Eduardo.FdezMedina@uclm.es](mailto:Eduardo.FdezMedina@uclm.es).

**Elena Ferrari** is a professor of Computer Science at the University of Insubria in Como, Italy. She received a Ph.D. in Computer Science from the University

of Milano (1997). She has been a visiting researcher at George Mason University, Fairfax, Virginia, and at Rutgers University, Newark, New Jersey. Her main research interests include database and web security, multimedia databases, virtual reality environments, and temporal databases. Professor Ferrari has published more than 50 papers in these areas. She is a member of the editorial board of the *International Journal of Information Technology* and of the *VLDB Journal*. Professor Ferrari is a member of IEEE Computer Society and ACM.

**Thierry Fornas** has been working as field marketing specialist in STMicroelectronics since 1999. He developed Smartcard businesses in Network Access with strategic customers partners. He previously joined Schlumberger Smartcard entity in 1996, where he was specialist in smartcard Operating System development and evaluation from 1996 to 1999. He earned an Engineer Degree in Electronics Science in 1995 at ESIEE Paris. Then, he performed a Master of Sciences in Microelectronics in Great Britain at Southampton University.

**Jorge Guajardo** is currently a member of the technical staff at the Secure Mobile Solutions Division of Infineon Technologies AG, Germany. He received B.S. and M.S. degrees from Worcester Polytechnic Institute in 1995 and 1997, respectively, and is working toward his Ph.D. in Electrical Engineering at the Ruhr-Universität Bochum, Germany. His research interests include implementation of public-key and symmetric-key cryptosystems in hardware and software, efficient crypto-algorithms in constrained environments and embedded processors, algorithms for elliptic and hyperelliptic curves, lattice based cryptosystems and lattice reduction algorithms, and Galois field arithmetic and architectures.

**Sushil Jajodia** is BDM international professor of Information and Software Engineering and the director of the Center for Secure Information Systems at the George Mason University, Fairfax, Virginia (USA). He has authored four books, edited 19 books, and published more than 250 technical papers in refereed journals and conference proceedings. He received the 1996 Kristian Beckman award from IFIP TC 11 for his contributions to the discipline of Information Security, and the 2000 Outstanding Research Faculty Award from GMU's School of Information Technology and Engineering. Dr. Jajodia is the founding editor-in-chief of the *Journal of Computer Security* and on the editorial boards of *ACM Transactions on Information and Systems Security* and *International Journal of Cooperative Information Systems*. He is the consulting editor of the Kluwer International Series on Advances in Information Security. He is a senior member of the IEEE and a member of IEEE Computer Society and Association for Computing Machinery.

**Ping Lin** is a Ph.D. student in the Computer Science and Engineering Department at Arizona State University (USA). She is interested in data and code security in distributed computing systems. Her current research is focusing on private information retrieval for XML documents and queries. She has published various conference papers in this research area. She received her M.S. from the Institute of Software at the Chinese Academy of Sciences in 2000. Her master's thesis was about implementation of GSM protocols for mobile terminals. She received her B.S. in Computer Science from Xiangtan University in China in 1993.

**Antonio Lioy** holds a Master's (summa cum laude) in Electronic Engineering and a Ph.D. in Computer Engineering. Currently, he is associate professor at the Politecnico di Torino (Italy), where he leads the TORSEC research group active in computer and network security. This group has taken part to several national and international projects, and it manages the EuroPKI public-key infrastructure. Prof. Lioy is a reviewer of the European Commission in the trust and confidence area, and a consultant for various private and public bodies. His specific research interests are in the fields of network security, PKI, and digital documents.

**Zaki Malik** is a master's thesis student in the Computer Science Department at Virginia Tech (USA). His current research interests include privacy in B2B environments, digital government, and web services. He is a member of the IEEE. He can be contacted at: zaki@vt.edu.

**Gerald Maradan** manages Pay TV and Content Protection programs in the Smartcard and Secure Solutions Business Unit at STMicroelectronics in Paris. He earned an Engineer Degree from the Institut National Polytechnique de Grenoble (INPG) and a Post-Master degree of Science (Diplome d'Etudes Approfondies) in Microsystems Conception from the Institut National Polytechnique de Toulouse (INPT).

**Christof Paar** is the chair for Communication Security at the University of Bochum in Germany. He received his Ph.D. in Electrical Engineering from the Institute of Experimental Mathematics at the University of Essen, Germany, in 1994. From 1995-2001 he was faculty member in the ECE Department of Worcester Polytechnic Institute in Massachusetts (USA). At WPI he founded the Cryptography and Information Security Labs. He is co-founder of the CHES (Cryptographic Hardware and Embedded Systems) Workshop series ([www.chesworkshop.org](http://www.chesworkshop.org)), which is one of the leading forums for cryptographic engineering. He has more than 50 scientific publications in applied cryptography, and he is the editor of six conference proceedings and special issues dealing with cryptographic engineering.

**Mario Piattini** holds an M.Sc. and Ph.D. in Computer Science by the Politechnical University of Madrid and is a Certified Information System Auditor by ISACA (Information System Audit and Control Association). He is a full professor at the Escuela Superior de Informática of the Castilla-La Mancha University (Spain) and author of several books and papers on databases, software engineering and information systems. He leads the ALARCOS research group of the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. His research interests include advanced database design, database quality, software metrics, object oriented metrics, software maintenance. He can be contacted at: [mpiattin@inf-cr.uclm.es](mailto:mpiattin@inf-cr.uclm.es).

**Abdelmounaam Rezgui** is a Ph.D. candidate in the Computer Science Department at Virginia Tech (USA). His current research interests include security and privacy in digital government, Internet databases, and web services. Rezgui received an M.S. in Computer Science from Purdue University. He is a member of the IEEE and the ACM. He can be contacted at: [rezgui@vt.edu](mailto:rezgui@vt.edu).

**Pierangela Samarati** is a professor at the Department of Information Technology of the Università degli Studi di Milano, Italy. Her main research interests are in data and application security and privacy, access control policies, models and systems, and information protection in general. She has been computer scientist in the Computer Science Laboratory at SRI (USA). She has been a visiting researcher at the Computer Science Department of Stanford University (USA), and at the ISSE Department of George Mason University (USA). She is the chair of the IFIP WG11.3 group on data and application security and a member of the Steering Committees of ACM Special Interest Group on Security, Audit, and Control (SIGSAC). She is co-recipient of the ACM-PODS'99 Best Newcomer Paper Award. For more information, visit <http://www.dti.unimi.it/~samarati>.

**Michael Thomsen** is a student at the Technical University of Denmark, currently pursuing a degree in electrical engineering. He holds a Master of Space Studies from the International Space University in Strasbourg, France, and has been an international exchange student at Florida Atlantic University in Boca Raton, Florida. Among his interests are embedded systems, networks and security. He has been actively involved in system engineering and the design of the power supply for the student satellite DTUsat, which was launched on June 30, 2003. He has been doing an internship at the European Space Agency's Technology Center in Noordwijk, The Netherlands, working on a magnetic based attitude control mode for the Proba satellite. He is currently involved in the design of an integrated data handling system for a small satellite probe.

**Duminda Wijesekera** is an assistant professor in the Department of Information and Software Engineering at George Mason University, Fairfax, Virginia, USA. Prior to joining GMU, he was a senior systems engineer at Honeywell Space Systems in Clearwater, Florida. He has been a visiting post-doctoral fellow at the Army High Performance Research Center at the University of Minnesota, and an assistant professor of Mathematics at the University of Wisconsin. Dr. Wijesekera received a Ph.D. in Computer Science from the University of Minnesota and a Ph.D. in Mathematical Logic from Cornell University. In the past, he has worked in quality of service, multimedia systems and program verification. His current interests are in information security and formal methods.

# Index

## A

access control paradigm 204  
Advanced Encryption Standard (AES)  
2, 25  
anonymous probing agents 189  
Application Service Providers (ASPs)  
273, 274  
arithmetic primitives 36  
ASIC 2  
attribute certificate 66, 79  
attribute ontology 184  
Authority Information Access (AIA) 73  
authorization certificate 66

## B

basic constraints 76  
biometry 138  
block cipher algorithms 15  
block ciphers 13  
bridge CA 94

## C

cando and do rules 160  
Certificate Revocation List (CRL) 69  
confidentiality 12

copyright protection 235  
cryptography 1  
cryptography on smart cards 107  
CTCPEC 102

## D

data authorization 295  
Data Encryption Standard (DES) 21  
data filters 194  
dercando rules 160  
Digital Government (DG) 182  
Digital Home Networks (DHN) 99  
digital rights 230  
Digital Rights Management (DRM)  
231, 259  
Distributed Application Hosting Services  
(DAHSS) 273

## E

e-business 274  
Electronic Code Book (ECB) 19  
Elliptic Curve Cryptosystems (ECC) 2,  
33  
eXtensible Access Control Markup  
Language (XACML) 246

eXtensible Markup Language (XML)  
206, 279  
eXtensible Rights Markup Language  
(XrML) 265

## F

fields 6  
finite fields 10  
Flexible Authorization Framework (FAF)  
149  
FPGA 2

## G

Global Provision and Obligation Set  
(GPOS) 170  
Global System for Mobile Communica-  
tion (GSM) 99

## I

identity certificate 66  
Integrated Circuit (IC) 99  
integrity 12  
Internet Protocol Security (IPSec) 327  
ITSEC 101

## J

Java™ Authentication and Authorization  
Service (JAAS) 326  
Java™ Card API 119

## L

Local ECM (LECM) 132

## M

materialization 154  
memories 105  
Merkle Hash paths 224  
Merkle signature 221  
mesh 90  
Microsoft .NET 317  
Microsoft's Common Object Model  
(COM) 319  
mobile privacy preserving agents 194

modular multiplication 39  
Montgomery modular multiplication 42  
MPEG-7 242  
multimedia information 231  
multimedia security 230

## N

name constraints 76

## O

Online Certificate Status Protocol  
(OCSP) 71  
open platform card 119

## P

policy constraints 76  
privacy preservation requirement 181  
Private Information Retrieval (PIR) 307  
Private Personal Network (PPN) 134  
profile-based paradigm 204  
Public Key Infrastructure (PKI) 64  
public-key algorithms 14  
public-key certificate (PKC) 65

## R

Random Number Generators (RNG)  
106  
reputation management model 186  
Rights Management (RM) 259

## S

scalability 218  
Scalable Vector Graphics (SVG) 237,  
250  
SDSI/SPKI certificates 82  
secure and selective dissemination of  
information 199  
Secure Socket Layer (SSL) 283  
security policies 294  
Selective Dissemination of Information  
(SDI) 199  
Semantic Web 177  
service wrappers 191  
SIM (Subscriber Identity Module) 99

Simple Object Access Protocol (SOAP)  
320

smart card 99

SMIL 241

SSXDI system 208

steganography 233

stream ciphers 13

subscription-based paradigm 204

Sun ONE (J2EE) 317

symmetric-key algorithms 13

Systems on Programmable Chip  
(SoPC) 3

Systems-on-Chip (RSoC) 3

## T

trust list 90

Trust Service Provider (TSP) 194

## U

universal description, discovery and  
integration ( 279

## V

VoiceXML 241

## W

watermarking 234

Web privacy 179

Web Service Description Language  
(WSDL) 279

## X

X509 66

XML 236



## NEW from Idea Group Publishing

- **The Enterprise Resource Planning Decade: Lessons Learned and Issues for the Future**, Frederic Adam and David Sammon/ ISBN:1-59140-188-7; eISBN 1-59140-189-5, © 2004
- **Electronic Commerce in Small to Medium-Sized Enterprises**, Nabeel A. Y. Al-Qirim/ ISBN: 1-59140-146-1; eISBN 1-59140-147-X, © 2004
- **e-Business, e-Government & Small and Medium-Size Enterprises: Opportunities & Challenges**, Brian J. Corbitt & Nabeel A. Y. Al-Qirim/ ISBN: 1-59140-202-6; eISBN 1-59140-203-4, © 2004
- **Multimedia Systems and Content-Based Image Retrieval**, Sagarmay Deb ISBN: 1-59140-156-9; eISBN 1-59140-157-7, © 2004
- **Computer Graphics and Multimedia: Applications, Problems and Solutions**, John DiMarco/ ISBN: 1-59140-196-86; eISBN 1-59140-197-6, © 2004
- **Social and Economic Transformation in the Digital Era**, Georgios Doukidis, Nikolaos Mylonopoulos & Nancy Pouloudi/ ISBN: 1-59140-158-5; eISBN 1-59140-159-3, © 2004
- **Information Security Policies and Actions in Modern Integrated Systems**, Mariagrazia Fugini & Carlo Bellettini/ ISBN: 1-59140-186-0; eISBN 1-59140-187-9, © 2004
- **Digital Government: Principles and Best Practices**, Alexei Pavlichev & G. David Garson/ISBN: 1-59140-122-4; eISBN 1-59140-123-2, © 2004
- **Virtual and Collaborative Teams: Process, Technologies and Practice**, Susan H. Godar & Sharmila Pixy Ferris/ ISBN: 1-59140-204-2; eISBN 1-59140-205-0, © 2004
- **Intelligent Enterprises of the 21st Century**, Jatinder Gupta & Sushil Sharma/ ISBN: 1-59140-160-7; eISBN 1-59140-161-5, © 2004
- **Creating Knowledge Based Organizations**, Jatinder Gupta & Sushil Sharma/ ISBN: 1-59140-162-3; eISBN 1-59140-163-1, © 2004
- **Knowledge Networks: Innovation through Communities of Practice**, Paul Hildreth & Chris Kimble/ISBN: 1-59140-200-X; eISBN 1-59140-201-8, © 2004
- **Going Virtual: Distributed Communities of Practice**, Paul Hildreth/ISBN: 1-59140-164-X; eISBN 1-59140-165-8, © 2004
- **Trust in Knowledge Management and Systems in Organizations**, Maija-Leena Huotari & Mirja Iivonen/ ISBN: 1-59140-126-7; eISBN 1-59140-127-5, © 2004
- **Strategies for Managing IS/IT Personnel**, Magid Igbaria & Conrad Shayo/ISBN: 1-59140-128-3; eISBN 1-59140-129-1, © 2004
- **Beyond Knowledge Management**, Brian Lehaney, Steve Clarke, Elayne Coakes & Gillian Jack/ ISBN: 1-59140-180-1; eISBN 1-59140-181-X, © 2004
- **eTransformation in Governance: New Directions in Government and Politics**, Matti Mälkiä, Ari Veikko Anttiroiko & Reijo Savolainen/ISBN: 1-59140-130-5; eISBN 1-59140-131-3, © 2004
- **Intelligent Agents for Data Mining and Information Retrieval**, Masoud Mohammadian/ISBN: 1-59140-194-1; eISBN 1-59140-195-X, © 2004
- **Using Community Informatics to Transform Regions**, Stewart Marshall, Wal Taylor & Xinghuo Yu/ISBN: 1-59140-132-1; eISBN 1-59140-133-X, © 2004
- **Wireless Communications and Mobile Commerce**, Nan Si Shi/ ISBN: 1-59140-184-4; eISBN 1-59140-185-2, © 2004
- **Organizational Data Mining: Leveraging Enterprise Data Resources for Optimal Performance**, Hamid R. Nemati & Christopher D. Barko/ ISBN: 1-59140-134-8; eISBN 1-59140-135-6, © 2004
- **Virtual Teams: Projects, Protocols and Processes**, David J. Pauleen/ISBN: 1-59140-166-6; eISBN 1-59140-167-4, © 2004
- **Business Intelligence in the Digital Economy: Opportunities, Limitations and Risks**, Mahesh Raisinghani/ ISBN: 1-59140-206-9; eISBN 1-59140-207-7, © 2004
- **E-Business Innovation and Change Management**, Mohini Singh & Di Waddell/ISBN: 1-59140-138-0; eISBN 1-59140-139-9, © 2004
- **Responsible Management of Information Systems**, Bernd Stahl/ISBN: 1-59140-172-0; eISBN 1-59140-173-9, © 2004
- **Web Information Systems**, David Taniar/ISBN: 1-59140-208-5; eISBN 1-59140-209-3, © 2004
- **Strategies for Information Technology Governance**, Wim van Grembergen/ISBN: 1-59140-140-2; eISBN 1-59140-141-0, © 2004
- **Information and Communication Technology for Competitive Intelligence**, Dirk Vriens/ISBN: 1-59140-142-9; eISBN 1-59140-143-7, © 2004
- **The Handbook of Information Systems Research**, Michael E. Whitman & Amy B. Woszczyński/ISBN: 1-59140-144-5; eISBN 1-59140-145-3, © 2004
- **Neural Networks in Business Forecasting**, G. Peter Zhang/ISBN: 1-59140-176-3; eISBN 1-59140-177-1, © 2004

*Excellent additions to your institution's library! Recommend these titles to your Librarian!*

**To receive a copy of the Idea Group Publishing catalog, please contact 1/717-533-8845, fax 1/717-533-8661, or visit the IGP Online Bookstore at: <http://www.idea-group.com>!**

**Note: All IGP books are also available as ebooks on netlibrary.com as well as other ebook sources. Contact Ms. Carrie Skovrinskie at [cskovrinskie@idea-group.com](mailto:cskovrinskie@idea-group.com) to receive a complete list of sources where you can obtain ebook information or IGP titles.**

**30-DAY  
FREE TRIAL!**

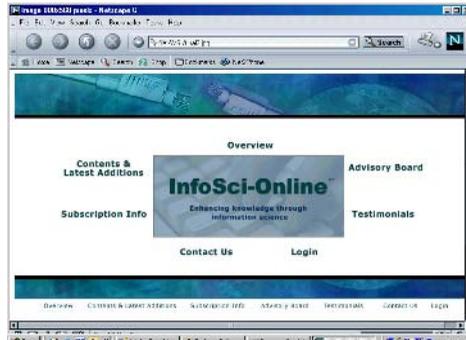
# InfoSci-Online Database

**www.infosci-online.com**

Provide instant access to the latest offerings of Idea Group Inc. publications in the fields of INFORMATION SCIENCE, TECHNOLOGY and MANAGEMENT

During the past decade, with the advent of telecommunications and the availability of distance learning opportunities, more college and university libraries can now provide access to comprehensive collections of research literature through access to online databases.

The InfoSci-Online database is the most comprehensive collection of *full-text* literature regarding research, trends, technologies, and challenges in the fields of information science, technology and management. This online database consists of over 3000 book chapters, 200+ journal articles, 200+ case studies and over 1,000+ conference proceedings papers from IGI's three imprints (Idea Group Publishing, Information Science Publishing and IRM Press) that can be accessed by users of this database through identifying areas of research interest and keywords.



#### **Contents & Latest Additions:**

Unlike the delay that readers face when waiting for the release of print publications, users will find this online database updated as soon as the material becomes available for distribution, providing instant access to the latest literature and research findings published by Idea Group Inc. in the field of information science and technology, in which emerging technologies and innovations are constantly taking place, and where time is of the essence.

The content within this database will be updated by IGI with 1300 new book chapters, 250+ journal articles and case studies and 250+ conference proceedings papers per year, all related to aspects of information, science, technology and management, published by Idea Group Inc. The updates will occur as soon as the material becomes available, even before the publications are sent to print.

InfoSci-Online pricing flexibility allows this database to be an excellent addition to your library, regardless of the size of your institution.

**Contact: Ms. Carrie Skovrinskic, InfoSci-Online Project Coordinator, 717-533-8845 (Ext. 14), [cskovrinskic@idea-group.com](mailto:cskovrinskic@idea-group.com) for a 30-day trial subscription to InfoSci-Online.**

A product of:



**INFORMATION SCIENCE PUBLISHING\***

Enhancing Knowledge Through Information Science

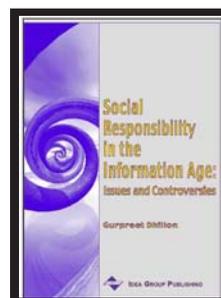
<http://www.info-sci-pub.com>

*\*an imprint of Idea Group Inc.*

# **Social Responsibility in the Information Age: Issues and Controversies**

Gurpreet Dhillon  
University of Nevada, USA

Understanding, appreciating and taking corrective steps to maintain and enhance social responsibility in the information age is important not only because of our increased dependence on information and communication technologies, but also because information and communication technologies pose complex challenges, which have had a lesser significance in the past. Although we have always acknowledged that increased awareness of social responsibility issues in the information age is essential, there has only been a sketchy review of the concerns and the inherent challenges.



*Social Responsibility in the Information Age: Issues and Controversies*, edited by Gurpreet Dhillon, is a first step in bringing together various viewpoints from around the world and in presenting a coherent argument. This exciting new title will address the changes information resource management, information technology and information systems have made upon society as a whole.

ISBN 1-930708-11-4 (h/c) • US\$74.95 • 282 pages • Copyright © 2002

**“Clearly the issue of social responsibility in the information age is an important one and it’s my hope that the collection of papers by eminent scholars in the field will help in enhancing our understanding of a topic area that is in its infancy.”**

*Gurpreet Dhillon, University of Nevada, USA*

**It’s Easy to Order! Order online at [www.idea-group.com](http://www.idea-group.com) or call our toll-free hotline at 1-800-345-4332!**

**Mon-Fri 8:30 am-5:00 pm (est) or fax 24 hours a day 717/533-8661**



## **Idea Group Publishing**

Hershey • London • Melbourne • Singapore • Beijing

***An excellent addition to your library***