MANIPULATIONS AND ERRORS, DETECTION AND LOCALIZATION

Ph. Godlewski [1] & P. Camion [2]

[1]ENST dép. RESeaux et CNRS UA 820, 75634 Paris, France

[2]INRIA, B.P. 105, 78153 Le Chesnay, France

ABSTRACT

We investigate the possibility of using error correcting codes in digital signatures. A scheme combining one way functions and a MDS code is presented and analyzed. We then study an attack upon this scheme and upon more general ones called "random knapsack schemes" involving a linear combination $\Sigma_i\ T(x_i,i)$ of the message elements x_i.

I. INTRODUCTION

Digital signature schemes provide two kinds of authentication services : integrity of messages and identification of users. This paper is concerned with integrity aspects of digital signatures. Various terminologies and techniques are used in this context : MAC, MDC, MIC, seal, cryptographic checksum, one way hash function, compression, condensation ...([1],[2],[3]). The motivation is to prevent malicious changes in a transmitted or stored message x . The basic process is the following : associate with x a short "certificate" $s(x)$ which is transmitted or stored in a secure manner (i.e. with protection against active attack). We will restrict ourselves to systems which do not require the sender and the receiver to share a secret key K.

The basic requirements are :

(i) $s(.)$ is easily computable, $s(x)$ is concise (e.g. from 8 up to 128 bytes),

(ii) $s(.)$ is unforgeable : given y in the signature domain, it is computationally unfeasible to calculate a quasi inverse $s^{-1}(y)$ of y.

To avoid small falsifications (e.g. change of a name, of an amount in a payment message), we add an extra condition :

(iii) Two messages with the same length must differ from d symbols or blocks.

In the following we assume that the message $\boldsymbol{x}$ is composed of symbols x_i belonging to an alphabet X, then $\boldsymbol{x} = (x_1, x_2, \dots x_k)$, we set $[k]=\{1,2,\dots,k\}$.

We distinguish two types of attacks :

(a) Given $\boldsymbol{x}$ find $\boldsymbol{x'}$ such that $s(\boldsymbol{x'})=s(\boldsymbol{x})$.

(b) Find two messages $\boldsymbol{x}$ and $\boldsymbol{x'}$ such that $s(\boldsymbol{x'})=s(\boldsymbol{x})$.

This two types have some similarities with the so called "known plaintext" and "chosen plaintext" attacks in a classical cryptographic system for confidentiality.

A more realistic attack of type (a), productive for the intruder, is the following :

(a') Given a message $\boldsymbol{x}$ and $y = s(\boldsymbol{x})$, a fraudulent message $\boldsymbol{x'}$ partially specified in a subset I of symbol positions, find x'_j for $j \in J = [k] \backslash I$ such that $s(\boldsymbol{x})=s(\boldsymbol{x'})$.

A similar attack (b') of type (b) can be defined.

In data networks, a reasonable goal should be to gather together different aspects of integrity, in particular :

- error detection and correction
- manipulation detection and localization.

Merging these items brings some "technical" problems. One major difficulty comes from the following fact : nearly all constructing methods for error-correcting codes are based on linear computations which are well known for their cryptographic weakness.

We study several schemes which use linear combinations of the different elements of the message.

II . RANDOM KNAPSACK SCHEMES

When designing an integrity signature scheme without secrete key, a basic need is to dispose of a one-way function ϕ. In contrast with well known public key algorithms such as RSA, there is no necessity here to invert ϕ with the help of some hidden trap door information. Then we can consider purely random generated knapsack :

Generate k random numbers $a_1, a_2, \dots, a_k$ bounded by M ; and calculate $s(x) = \sum_i x_i\, a_i$. In this paragraph, the alphabet X is binary, $X = \{0,1\}$.

When k is large enough, this scheme is deeply insecure against attack of type (b) as shown by our next proposition.

Proposition 1 : Given k integers $a_1, a_2, \dots\, a_k$ with $a_i \leq M$, it is always possible to find $I, J \in \{1,2,\dots,k\}, I \neq J$, such that
$\sum_{i \in I} a_i = \sum_{j \in J} a_j$
in $O(k\log(k))$ operations when $M \leq k^{\log(k)/4}$.

For instance if $k = 2^{20}$ (message with 128 Kbyte) and $M = 2^{100}$, an attack needs about 20.10^6 additions.

Proof : After sorting, we can assume $a_{i-1} \leq a_i$ for $1 < i \leq k$. We derive a new sequence of length k : $b_1 = a_1$ and $b_i = a_i - a_{i-1}$ for $1 < i \leq k$. There exists an element a'_1

in $\{b_i\}$ such that $a'_1 \leq M/k$. If $a'_1 = b_j$, we then discard from the sequence $\{a_i\}$, the two elements a_j and a_{j-1} involved in a'_j. Then we determine an other element a'_2 such that $a'_2 \leq M/(k-2)$. Iterating the process $k'=k/4$ times, we then obtain k' elements $a'_1, a'_2, \ldots, a'_{n'}$ such that $a'_i < M/(k-2i) \leq 2M/k$.

Assuming than $k=2^u, M=2^v$, we have at our disposal a new sequence $\{a'_i\}$ of length $k'=2^{u-2}$ with elements bounded by $M'=2^{v-u+1}$. We consider the recursion :

$u^{(t+1)} = u^{(t)} - 2$

$v^{(t+1)} = v^{(t)} - u^{(t)} + 1$, with $u^{(0)} = u$ and $v^{(0)} = v$,

then we obtain :

$u^{(t)} = u - 2t$ and

$v^{(t)} = v - tu + t^2$.

Note that $v^{(t)}$ reaches its minimum v_{min} for $t=t_{min}=u/2$, then $v_{min}= v-u^2/4$. If $v_{min}<0$ all the elements of the sequence $\{a_i^{(tmin)}\}$ vanish. This occurs if $v<u^2/4$ or $M \leq k^{\log(k)/4}$. Each step of the algorithm requires $k^{(t)}/2 = 2^{u^{(t)}-1}$ additions and a sorting, that is $O(k^{(t)}log(k^{(t)}))$ additions. Then the total complexity is less than $k\,logk + (2k/3)(k+logk) < 4\,k\,log(k)$ additions ; that is $O(k\,logk)$. The algorithm needs no more than $0(klog(k)log(M))$ binary operations.

Notice that this algorithm is not probabilistic : at each step, the worst case is considered. To perform attack of type (a), algorithms which require more computational effort exist. A probabilistic algorithm will appear as a consequence of proposition 2.

III . ERROR-LOCALIZING CODES SCHEME

We present a scheme combining one way function and error correcting code :

Split the message x into blocks $x_i \in \mathrm{F}_2^u$ of length u (e.g. $u \approx 100$), $x = (x_1,x_2, \ldots, x_k)$, then use a one way injective function $\phi_i(.)$ from F_2^u to F' (e.g. |F'| $=q \approx$

2^{128}). For instance $\phi_i(x_i)$ can be written as $\phi_i(x_i) = \phi(i \mid x_i)$, where "|" stands for concatenation. We therefore obtain k symbols $\gamma_i = \phi_i(x_i)$ in F'. Encode $(\gamma_1, \gamma_2, \dots, \gamma_k)$ with a $[n,k,d\ ]$ error correcting code over F'. The n-k (e.g. $n\text{-}k = 4$) redundancy symbols $\gamma_{k+1}, \gamma_{k+2}, \dots, \gamma_n$ form the signature s.

Detection and correction

We consider codes over very large alphabet F' of cardinality q . Then $n < q$, and we can restrict ourselves to MDS code. It is well known that most of the error correcting codes are far from perfection. More precisely, the density $\Delta = 2^{-\delta}$ of the packing is small ; Δ is the fraction of the space F'^n which lies inside the spheres B_t of radius t centered on the code words. Therefore, most of space may be used for detection. For a $[n,k,d]$ code C, we have $t = [(d-1)/2]$ and $\Delta = |C| \,.\, |B_t| \,/\, |F'^n|$. One can consider than a part $log\,|B_t|$ of the redundancy (i.e. $(n\text{-}k)log(q)$ bits) is used for correction, the remaining part $\delta = (n\text{-}k)log(q) - log\,|B_t|$ for detection. A straightforward estimation gives :

$$\delta \approx log(q)\ \ [\, n - k - t\,(\,1 + log_q n\ \text{-}log_q t\,)\,]$$

For our application, we have $t = [\frac{n-k}{2}]$ (cf. MDS codes), and possible order of magnitude of the parameters is : $1 \le t \le 10$, $n < 2^{32}$, $q > 2^{100}$. Then, we have $t < n << q$ and $\delta \approx log(q)\ [(n\text{-}k)/2]$. Half of the signature symbols are used to detect error or manipulation.

Localization or correction

Using Berlekamp-Massey algorithm, it is possible to localized errors in $O(n.d)$ operations over F'. But, due to the presence of the one way functions ϕ_i, the error evaluation on the γ_i can not be exploited to correct errors on the x_i. . However, for some type of messages, errors can perhaps be corrected by try and error procedures for instance, by exploiting natural redundancy of a language.

The error correction algorithm can be carried out only if it is possible to invert each ϕ_i for each position i in error using some (secret) trap door information.

Weakness of the scheme in low characteristic

It is important to select the field F' with a high characteristic. For instance if the characteristic of F' is 2, i.e. $q = 2^v$, then it is possible to perform an attack of type (a') by modifying $|J| = O((n-k)v)$ blocks of a arbitrary fraudulent message x'. In this case the signature s is computed in $F_2^{v(n-k)}$ following the formula :

$$s = \Sigma_{i \in [k]} \gamma_i H^{(i)}$$

where γ_i and $H^{(i)}$ are respectively $1 \times v$ and $v \times v(n-k)$ binary matrices. The binary image of the $[n,k]$ MDS code over F' is then a $[nv,kv]$ code over F_2 with parity check matrix $H = [H^{(1)},...,H^{(k)}]$.

Let J the set of position used to adapt the fraudulent message to the desired signature, we consider the cheating procedure :

- For the legitimate message, compute $y_{(i)} = \gamma_i H^{(i)}$ for $i \in [k]$ and then $a = \Sigma_{i \in [k]} y_{(i)}$.
- For a fraudulent message x', choose randomly $\{x'_j\}$ for $j \in J$, compute similar quantities, $\gamma'_i = \phi_i(x'_i)$, $y'_{(i)} = \gamma'_i H^{(i)}$; $a' = \Sigma_{i \in [k]} y'_{(i)}$.
- Find $\{ \varepsilon_j ; \varepsilon_j \in F_2, j \in J \}$ such that $a - a' = \Sigma_{j \in J} \varepsilon_j (y_{(j)} - y'_{(j)})$; this is possible if the vectors $(y_{(j)} - y'_{(j)}), j \in J$, generate $F_2^{v(n-k)}$,

  which is true with high probability if $|J| \approx 2(n-k)v$.
- Let $x^\circ_j = \varepsilon_j x_j + (1 - \varepsilon_j) x'_j$ be the values of the final message x' in the positions $j \in J$.

The complexity of this procedure resides essentially in the computation of $O((n-k)v)$ additional one-way functions $\phi_j(x'_j)$. It is possible to specify similar procedures with smaller $|J|$ and for code defined on other fields with low characteristic. For such fields, the proposed scheme is therefore very weak.

In the following paragraph, we present an attack adapted for high characteristic.

IV . AN ATTACK UPON SCHEMES BASED ON LINEAR COMPUTATIONS

We consider a generalization proposed by Gaston Gonnet, Waterloo of the binary knapsack scheme for signature. To precise this scheme it is sufficient to present an attack of type (a') which consists in solving the following problem :

Problem A :

Given a finite set of indices J, an integer $a<M$, and a function $T(.,.)$ from $X \times J$ into $\mathbf{Z}$, find a sequence in $X^{|J|}$, $x=(x_j)_{j\in J}$ which satisfies

$\Sigma_{j\in J}\ T(x_j,j) = a$ (1)

Remark : Notice that solving problem A reduces to solving the following knapsack :

$\Sigma_{(x,j)\in J\times X}\ \xi(x,j)\ T(x_j,j) = a$,

subject to

$\forall\ (x,j),\ \xi(x,j) \in \{0,1\}$

$\forall j\ ,\ \Sigma_{x\in X}\ \xi(x,j) = 1.$

When we exhibit a sequence x for a set of indices J which verifies (1), we say that set J is a support for a . The goal is to find an algorithm to resolve the problem for small or medium support size $|J|$.

In [4], this kind of problem has been studied in a algebraic structure different from the additive group $(\mathbf{Z},+)$ of integers. The considered structure G is the group of invertible 2x2 matrices with entries in the field F_P . The algorithm proposed in [4] supposes the existence of a chain of subgroups H_i, $G \supseteq H_{\mu-1} \supseteq H_{\mu-2} \supseteq \ldots \supseteq H_1$ such that the indexes $[H_r : H_{r-1}]$ are not too large. The method can be applied to commutative groups with small prime exponent. When G contains a (cyclic) subgroup $\mathbf{Z}/P\mathbf{Z}$ with large prime P, a similar method can be used embedding $\mathbf{Z}/P\mathbf{Z}$ in $\mathbf{Z}$ and using the Chinese remainder theorem.

A probabilistic algorithm to solve Problem A

Let M be an upper bound for the possible values of a. We choose M as a product of coprime numbers $M = \prod_{r\in[\mu]} P_r$ where $P_r \approx P$, $P < 2^p$. We assume that $|J| = 2^\mu b$. It will appear that $|X^b| > 2^p$.

Setting $J = J_1^{(\mu)}$ and by successive dichotomies over J, we obtain :

$$J_s^{(r)} = J_{2s-1}^{(r-1)} \cup J_{2s}^{(r-1)} \text{ where } J_{2s-1}^{(r-1)} \cap J_{2s}^{(r-1)} = \emptyset \ , \ |J_s^{(r)}| = 2^{r-1} b$$

We thus get μ partitions of J for $r+1 = \mu, \mu-1, ..., 2, 1$:

$$J = \bigcup_{s\in[2^{\mu-r}]} J_s^{(r)}.$$

The algorithm has μ steps. The principle is to determine for each step r and each set $J' = J_s^{(r)}$, $s \in [2^{\mu-r}]$, a set of $K(P,r)$ solutions to the equation

$$\Sigma_{i\in J'} T(x_i,i) = a\ \delta_1(s) \quad \text{modulo} \quad P^{(r)} = \prod_{i\in[r]} P_i ,$$

where $\delta_1(s) = 1$ if $s = 1$, and 0 otherwise. Grossly, we choose $K(P,r) = O(P)$.

Basic procedure : It consists in determining from 2 sets V_1 et V_2, each with $O(P)$ elements of the form $(T(x_{i_1}, i_1), ..., T(x_{i_t}, i_t))$, $t = 2^{r-1}b$, for $r \geq 1$, a set V', $V_1 \times V_2 \supseteq V'$, and $|V'| = O(P)$ in which every $2^r b$-tuple's components add up to 0 (or a) modulo P_i, $i < r$. If all the numbers are specified modulo P_r this procedure requires essentially a sorting and then $O(P\log P)$ additions. Indeed V_1 (resp. V_2) is sorted according to the value of the component's sum of its elements modulo P_r. After the two sortings are performed, then selecting the matching couples needs $O(P)$ comparisons. More precisely, if $|V_1| = \alpha_1 P$ and $|V_2| = \alpha_2 P$, finding out all matching couples need $(\alpha_1 + \alpha_2)P$ comparisons since two elements have been compared, the smallest is dropped.

For a fixed step r of the algorithm, this procedure is applied $2^{\mu-r}$ times for determining $2^{\mu-r}$ sets $V_s^{(r)}$. Each set $V_s^{(r)}$, $s \in [2^{\mu-r}]$, contains $O(P)$ elements with support $J_s^{(r)}$.

Algorithm complexity :

The basic procedure is applied $2^{\mu-1}+2^{\mu-2}+...+2^0 \approx 2^\mu$ times. If we assume that the complexity of computing one value $T(x_j,j)$ is O(1), the overall complexity is $\kappa \approx 2^\mu$ $P \log(P)\, p = p^2\, 2^{\mu+p}$ for a number $|U|=2^\mu b$ of symbols used to adapt the signature.

If, we set $|X|=2^\alpha$, $M=2^m$, $P=2^p$, we then get : $m=p\mu$, $\alpha b \approx 2p$.

For $\alpha=1$, we obtain $b \approx 2p \approx 2m/\mu$, $\kappa \approx 2^{\mu+m/\mu}\,(m/\mu)^2$ which reaches its minimum for $\mu \approx \sqrt{m}$, we then have $\kappa \approx 2^{2\sqrt{m}}\, m$ and $\approx |U| \approx 2^{\sqrt{m}+1}\sqrt{m} \approx \sqrt{2\kappa}$.

If we consider larger blocks (e.g. $\alpha \approx 100$) we can choose $b=1$, and we obtain the same type of result : $\kappa \approx 2^{2\sqrt{m}}$ m and $\approx |U| \approx 2^{\sqrt{m}}$.

Proposition 2 : Using a probabilistic algorithm, it is possible to solve problem A in $O(2^{2\sqrt{m}})$ operations modifying only $|U| \approx 2^{\sqrt{m}+1}\sqrt{m}$ symbols ($|U| \approx 2^{\sqrt{m}}$ if $|X| > 2^{\sqrt{m}}$) .

Application : $M \approx 2^{100}$, in the binary case (cf. paragraph 2) , it is possible to forge a (fraudulent) message with the same signature by adapting $|U| \approx 2^{\sqrt{100}}\, 2\sqrt{100} \approx 20$ 000 bits, the process needs about 10^6 operations.

If the signature domain is sufficiently large (say m≈1000 bits) this attack is clearly ineffective. The security of the scheme proposed in § **III** remains an open problem when the field F is $\mathbf{Z}/q\mathbf{Z}$ where q is a prime such that $log(q) \approx 128$, and C is a $[n,k]$ code with $n\text{-}k \approx 8$, leading to a signature which is m = $(n-k)log(q)$ $\approx 128.8=2^{10}$ bits long.

REFERENCES

[1] D.W. Davies and W.L. Price, "Security for computer Networks", John Wiley and Sons, Chichester 1984.

[2] R.R. Jueneman, "A High Speed Manipulation Detection Codes", Proceeding of crypto 86, Springer-Verlag 1987, pp.327-346.

[3] M. Campana and M. Girault, "How to Use Compressed Encoding Mechanisms in Data Protection", Securicom 88, March 15-17, pp.91-110.

[4] P. Camion, "Can a Fast signature Scheme Without Secret Key be Secure?", in AAECC, Lecture Notes in Computer Science, n°228, Springer-Verlag.