Methodology for Digital Money based on General Cryptographic Tools*

Stefano D'Amiano[1] and Giovanni Di Crescenzo[2]

[1] Computer Science Department,
Cornell University, Ithaca, NY, USA

[2] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, 84081 Baronissi (SA), Italy

Abstract. In this paper we investigate methodologies for off-line digital cash using general cryptographic tools. First we give a way for off-line spending of coins using non-interactive zero-knowledge proofs of knowledge with preprocessing. Under this paradigm and using other general cryptographic tools, we show how to obtain the property of dividability of coins and give a technique for avoiding double spending of coins.
Chaum and Pedersen considered a model in which the Bank discovers the author of a double spending of a coin immediately after that coin has been deposited, and proved that in this model transferred coins grow in size. We consider a different model and show how to obtain transferability of coins without any increase in size.

1 Introduction

An electronic cash system is the digital equivalent of paper cash and can be seen as a collection of protocols with one distinguished player called Bank. It usually supports transactions of four different types, each one being the digital equivalent of a real-life transaction: opening an account, withdrawing, spending and depositing a coin.

An electronic cash system has the property of off-line spending of coins if the Bank is not involved in the protocol for spending a coin. The main approaches to the construction of an electronic cash system with the property of off-line spending are three and are based respectively on blind signatures, on zero-knowledge proofs of knowledge and on oblivious authentication.

A blind signature scheme is a protocol between two parties in which a first party (in a cash system, the Bank) signs a message received by the second party (in a cash system, a user). The signature given is blind in the sense that the first party does not get useful informations about the message signed. The first blind signature scheme has been given in [4] using RSA. Other electronic cash systems based on blind signatures are in [3, 12].

* Partially supported by Italian Ministry of University and Research (M.U.R.S.T.) and by National Council for Research (C.N.R.).

An oblivious authentication scheme is a protocol between two parties in which a first party (in a cash system, the Bank) issues to a second party (in a cash system, a user) a digital document in which the user's identity is embedded in some way, together with a witness of this identity. The authentication given is oblivious in the sense that the first party is not able at a later stage to link the digital document to the time in which he has issued it. An oblivious authentication scheme has been given in [13] where it was implemented using one-way functions plus a preprocessing stage and using RSA.

A non-interactive zero-knowledge proof system of knowledge with preprocessing is a protocol between two parties in which, after an interactive preprocessing, the first party (in a cash system, the spender) can send a single message to the second party (in a cash system, the receiver) such that this message represents a zero-knowledge proof (see [15]) that the first party knows a witness for the truth of a certain statement. Such a system can be used by a spender to prove to the receiver the knowledge of a signature of a coin released by the Bank. A non-interactive zero-knowledge proof system of knowledge with preprocessing based on general complexity assumptions and a cash system based on it have been given in [6] (see also [21]).

Our results.
Taking the paper [6] as a starting point, we investigate methodologies for off-line digital cash using general cryptographic tools. First we give a general way for off-line spending of coins using non-interactive zero-knowledge proofs of knowledge with preprocessing. Under this paradigm and using other general cryptographic tools, we show how to obtain the property of dividability of coins and give a technique for avoiding multiple spending of coins even when the coin is transferred to many users before being deposited to the Bank.

In [5] it is showed that transferred cash grows in size. The model considered requires that the Bank discovers the author of a double spending of a coin immediately after that coin has been deposited. By considering a different model, we show how to obtain transferability of coins without any increase in their size.

Organization of the paper.
In Section 2 we give some notations and review some background cryptographic notions that will be useful in our construction. In Section 3 we recall definition and properties of electronic cash systems. In Section 4 we give a general way for off-line spending of coins using non-interactive zero-knowledge proofs of knowledge with preprocessing. In Section 5 we show how to obtain the property of dividability of coins. In Section 6 we give a technique for avoiding multiple spending of coins. In Section 7 we give a sketch of proof of the correctness of our construction.

2 Notations and basic tools

Notations. We use the symbols $\circ$ and $\oplus$ to denote respectively concatenation and bitwise xor of binary strings. If S is a probability space, then "$x \leftarrow S$" de-

notes the algorithm which assigns to x an element randomly selected according to S. If $p(\cdot,\cdot,\cdots)$ is a predicate, the notation $Prob(x \leftarrow S; y \leftarrow T; ... : p(x,y,\cdots))$ denotes the probability that $p(x,y,\cdots)$ will be true after the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, ... The notation $\{x \leftarrow S; y \leftarrow T; \cdots : (x,y,\cdots)\}$ denotes the probability space over $\{(x,y,\cdots)\}$ generated by the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \cdots$. By the writing (A,B) we denote a pair of Turing machines A and B. By the writing (o_a,o_b)=(A,B)(i_a,i_b) we denote an execution of the protocol (A,B), where A's input is i_a, B's input is i_b, A's output is o_a and B's output is o_b.

Basic tools.

Now we briefly review the concepts of pseudo-random functions, secure commitment schemes, signature schemes and non-interactive zero-knowledge proofs of knowledge with preprocessing, all playing an important role in our construction.

Pseudo-random Functions. The concept of pseudo-random function has been introduced by Goldreich, Goldwasser, and Micali [14]. Intuitively, we can say that a collection $F = \{f_s\}$ of functions $f_s : \{0,1\}^n \rightarrow \{0,1\}^{n^c}$ is c-pseudo-random if the output of a function randomly chosen from the collection and evaluated on arguments chosen by a polynomial time algorithm cannot be distinguished from the output of a truly random function. In [14] it is shown how to construct a c-pseudo-random collection of function for any constant c from any pseudo-random generator. In the sequel, whenever the constant c is clear from the contest, we will just say pseudo-random function.

Secure Commitment Schemes. Given the coin tosses s, r and a bit b, a secure commitment scheme for the bit b is an efficient algorithm E such that: 1) it is hard to compute a commitment which can be decommitted both as 0 and as 1, and 2) it is computationally hard to distinguish commitments of 0 from commitments of 1, where the string $d = E(b,s,r)$ is a commitment to bit b, and the strings s, r are a decommitment of d as the bit b. The secure commitment scheme can be easily extended to any string x. If $x = b_1...b_m$ is an m-bit string then $E(x,s,r)$ is intended to consist of the m different commitments $E(b_i,s,r_i)$, $i = 1,...,m$, where $r = r_1...r_m$ is an nm-bit string. In this case, it can be seen that property 2 can be extended to any two strings. Also, we observe that if $d = E(x,s,r)$ and s and r are known, then the string x committed to by d can be efficiently computed. This definition of commitment is inspired by the bit-commitment protocol of Naor [18] which is based on any pseudorandom number generator. The commitment is based on a random challenge given to the committer. For a random challenge and a random generator, there is no way to cheat in a commit phase. Whenever a random reference string σ is available, we can implement the commitment scheme by applying Naor's protocol by choosing the generator (one way function) and the long-enough challenging string non-interactively from some portion of the random string σ (used just for this purpose). All the commitments will, then, use this choice (for more details on this, see [18]).

Signature Schemes. A signature scheme is a triple (G, S, V) of efficient algorithms. G is a generation algorithm that on input a random string outputs a pair (pk, sk), where pk is a public key and sk is a secret key. S is a signing algorithm that on input a message m and the secret key sk, outputs a signature sig_m. V is a verification algorithm that on input a message m, the public key pk and the signature sig_m, verifies that sig_m is a signature of message m computed using the signing algorithm S. Two important properties of signature schemes are that any party can verify the correctness of a signature of a given document, and that no party can forge a signature for a new document, even after seeing arbitrary signature samples adaptively chosen. Signature schemes have been introduced by [10] and further elaborated by [16], [17], [19] and [22].

Zero-knowledge Proofs of Knowledge. Zero-knowledge proofs of membership were introduced in [15]. Roughly speaking, a zero-knowledge proof of membership is a two-party protocol in which one party, called the Prover, convinces the second party, called the Verifier, that a certain input string x belongs to a language L without releasing any additional knowledge. Subsequently, in [23] and [11] the concept of a zero-knowledge proof of knowledge was considered. Here, the Prover wants to convince the Verifier that he *knows* a witness w such that the polynomial-time relation $R(x, w)$ holds.

Non-Interactive Zero-Knowledge Proofs of Knowledge with Preprocessing. The concept of a non-interactive zero-knowledge (NIZK) proof has been introduced by [2] where it is proved that, if a random string readable by both the prover and the verifier is available, it is possible for the prover to give non-interactive zero-knowledge proofs for any NP language. (See [1] for formal definitions and proofs.) The definition of a NIZK proof of knowledge has been given in [7] where it is shown a NIZK proof of knowledge for any polynomial-time relation.
In a NIZK proof of knowledge with preprocessing, after an interactive preprocessing stage, a prover can give any polynomial (in the length of the preprocessing stage) number of NIZK proofs of knowledge to the verifier. Let R be a polynomial-time relation and L_R be the language of strings x such that there exists w for which $R(x, w) = 1$. Here we recall the definition of [6] of NIZK proof system of knowledge with preprocessing for the relation R.

Definition 1. Let $\mathrm{P} = (\mathrm{P}_1, \mathrm{P}_2)$ and $\mathrm{V} = (\mathrm{V}_1, \mathrm{V}_2)$, where P_1 and V_1 are interactive probabilistic poly-time Turing machines, P_2 is a probabilistic poly-time Turing machine, and V_2 is a deterministic poly-time Turing machine. The pair (P,V) is a *Non-Interactive Zero-Knowledge Proof System of Knowledge with Preprocessing* for the polynomial-time relation R if:

1. *Completeness:* For all constants c, all $(x, w) \in R$ and all sufficiently large n,

$$\mathrm{Prob}((\alpha, \beta) \leftarrow (\mathrm{P}_1, \mathrm{V}_1)(1^n); (x, Proof) \leftarrow \mathrm{P}_2(\alpha, x, w) :$$

$$\mathrm{V}_2(\beta, x, Proof) = 1) \geq 1 - 1/n^c.$$

2. *Soundness:* There exists an extractor algorithm $E = (E_1, E_2)$ such that for all pairs $P' = (P'_1, P'_2)$ of efficient algorithms, for all d and all sufficiently large n,

$$\text{Prob}((\alpha,\beta) \leftarrow (P'_1, E_1)(1^n); (x, Proof) \leftarrow P'_2(\alpha);$$
$$w \leftarrow E_2(\beta, x, Proof) : (x,w) \in R) > (1 - n^{-d}) \cdot$$
$$\text{Prob}((\alpha,\beta) \leftarrow (P'_1, V_1)(1^n); (x, Proof) \leftarrow P'_2(\alpha) : V_2(\beta, x, Proof) = 1).$$

3. *Zero-Knowledge:* For each V', there exists an efficient algorithm S such that for all $x_1, x_2, \ldots \in L_R$, for all efficient nonuniform algorithms D, for all constants e, and all sufficiently large n,

$$|\text{Prob}(y \leftarrow View_{V'}(1^n, x_1, x_2, \cdots) : D_{1^n}(y) = 1) -$$
$$\text{Prob}(y \leftarrow S(1^n, x_1, x_2, \cdots) : D_{1^n}(y) = 1)| < n^{-e},$$

where $\text{View}_{V'}(1^n, x_1, x_2, \cdots)$ is the view of V' in the executions of (P_1,V_1) and (P_2,V_2) on inputs $x_1, x_2, \ldots$

Essentially, we can think of (P_1,V_1) as the protocol for the preprocessing stage and of (P_2,V_2) as the (non-interactive) protocol for the proof stage.

A NIZK proof of knowledge with preprocessing has been given in [6]. An anonymous version of this protocol is in [9], where a verifier, after executing many preprocessing protocols, is not able to associate the conversation of a proof protocol to any preprocessing protocol. We will use this version in the construction of our electronic cash system.

3 Electronic cash systems

An Electronic Cash System is the digital counterpart of paper cash. It can be seen as a set of cryptographic protocols where the players are a Bank B and a collection of users $\{U_i\}$.

An electronic cash system should allow the supportability of transactions and operations representing the digital equivalent of (at least) all the real-life operations, as opening an account; withdrawing, depositing and spending a coin. Also, it can have some additional properties, as dividability of coins and off-line spending of coins or any desired piece of coins. On the other hand, as real-life coins are physical objects, they guarantee a satisfactory level of security to a bank and to the users. Then an electronic cash system should provide (at least) the same level of security guaranteed by its physical counterpart.

Our definition of electronic cash system is essentially based on those of [4, 13].

Definition 2. An off-line electronic cash system is a pair $(\mathcal{S}, \mathcal{Q})$, where $\mathcal{S} = \{Bank, U_1, \ldots, U_n\}$ is a set of interactive probabilistic polynomial-time Turing machines, and $\mathcal{Q}$ is the fourtuple (OPEN, WITHDRAW, SPEND, DEPOSIT). OPEN=O=(O_1,O_2) is a protocol executed by the Bank and a user U_i and allows user U_i to open an account; WITHDRAW=W=(W_1,W_2) is a protocol executed by

the Bank and a user U_i and allows user U_i to withdraw a coin from his account; SPEND=S=(S_1,S_2) is a protocol executed by two users U_i and U_j and allows user U_i to pass a coin to user U_j; DEPOSIT=D=(D_1,D_2) is a protocol executed by the Bank and a user U_i and allows user U_i to deposit a coin into his account. Moreover, the pair $(\mathcal{S}, \mathcal{Q})$ satisfies the following requirements:

1. *No forging:* For all integers $k > 0$, given the transcripts of k protocols $W_1,\ldots,W_k$, for each efficient nonuniform algorithm Adv, the probability that Adv computes $k+1$ coins $c_1,\ldots,c_{k+1}$ such that for each $i = 1,\ldots,k+1$, $S=(Adv,\cdot)(c_i,\cdot) = (\cdot, accept)$ is negligible in n.

2. *No tracing:* For all integers $k > 0$, given the transcripts of k protocols $W_1,\ldots,W_k$ and of k protocols $D_1,\ldots,D_k$, for each efficient nonuniform algorithm Adv, for each coin c such that $D=(\cdot,Adv)(c,\cdot) = (\cdot, accept)$ with nonnegligible probability, for each $i,j \in \{1,\ldots,n\}$, the probability that Adv computes $l \in \{i,j\}$ such that $W=(Adv,U_l)(\cdot,c_i) = (\cdot, accept)$ or $S=(U_l,\cdot)(c_i,\cdot) = (\cdot, accept)$ is at most $1/2+$ a term negligible in n.

3. *No double spending:* For each efficient nonuniform algorithm Adv, for each pair (c_1,c_2) of coins such that $S=(Adv,S_i)(c_i,\cdot) = (\cdot, accept)$ for $i = 1,2$ with nonnegligible probability, there exists an efficient algorithm $Detect$ such that, given the transcripts of the execution of protocol $S=(Adv,S_i)$ on input $(c_i,\cdot)$, for $i = 1,2$, outputs ID_{Adv} with nonnegligible probability in n.

4. *No framing:* For all integers $k > 0$, given the transcripts of k protocols $W_1,\ldots,W_k$ and of k protocols $D_1,\ldots,D_k$, for each efficient nonuniform algorithm Adv, for each coin c, for each $i,j \in \{1,\ldots,n\}$, the probability that Adv computes $l \in \{i,j\}$, and c_1,c_2 such that $S=(U_l,U_h)(c,\cdot) = (\cdot, c_1 \circ accept)$ and $S=(U_l,U_k)(c,\cdot) = (\cdot, c_2 \circ accept)$, for some $h,k \in \{1,\ldots,n\}$ is at most $1/2+$ a term negligible in n.

Instead of giving a complete description of protocols OPEN, WITHDRAW, SPEND, DEPOSIT, which would be hard to read, we divide the presentation into three parts. First we give a way for off-line spending of coins using non-interactive zero-knowledge proofs of knowledge with preprocessing. Then, using other general cryptographic tools as pseudo-random functions and secure commitment schemes, we show how to obtain dividability of coins. Finally, we give a technique based on signature schemes for avoiding multiple spending of coins even when the coin is transferred to many users before being deposited to the Bank.

4 NIZK proofs of knowledge and digital money

In this section we show how non-interactive zero-knowledge proofs of knowledge with preprocessing together with secure commitment schemes and signature schemes give a methodology for obtaining off-line spendable digital money.

4.1 The cash system

Let $E(\cdot,\cdot,\cdot)$ be a secure commitment scheme and let (pk_B, sk_B) be a pair of a public and a secret key specifying the signature scheme (G_B, S_B, V_B) of the Bank. Also, let (P,V) be the NIZK proof of knowledge with preprocessing for any NP-complete language given in [9]; we denote by (P_1,V_1) its preprocessing protocol and by (P_2,V_2) its proof protocol. Finally, let n be a security parameter, σ be a sufficiently long random reference string, and ID_B, ID_U be n-bit strings denoting the identity of the Bank and of user U respectively.

We assume that at the beginning of the cash system all the above tools are written on a public file PF. Then, in the protocol of the opening of an account, a user U and the Bank establish some common information which will allow user U to give non-interactive zero-knowledge proofs of knowledge at a later stage. Essentially, they run the preprocessing stage of the given proof system of knowledge.

Opening the account:

- **Bank and U:** run the protocol (P_1,V_1), where U runs algorithm P_1 and the Bank runs algorithm V_1.

The protocol for withdrawing a coin is made of two rounds: first the user U sends its request to the Bank, and then the Bank sends its authorization to user U. Informally, the protocol is the following: the user U randomly generates a string c and sends to the Bank a commitment to c computed using the secure commitment scheme E. The Bank answers to U with a signature of the commitment, computed using her signature scheme (G_B, S_B, V_B). The knowledge of this signature will represent her authorization to spend the coin c. More formally, the protocol for withdrawing a coin is the following:

Withdrawing a coin:

- **U:** randomly choose an n-bit string c and an n^2-bit string r;
  compute $com = E(c, \sigma, r)$ and send com to the Bank.
- **Bank:** compute $sig_{com} = S_B(sk_B, com)$ and send sig_{com} to U.
- **U:** if $V_B(pk_B, com, sig_{com}) = 1$ then accept the coin c.

Observe that the commitment to c is necessary, for otherwise the coin c would be later easily traceable from the Bank.

The protocol for spending a coin uses the non-interactive proof of knowledge. To pass a coin c to user U_2, user U_1 sends him the string c and a non-interactive zero-knowledge proof of knowledge of a commitment to c and of a signature of this commitment released by the Bank. If the proof is convincing, then user U_2 accepts the coin c from U_1. More formally, the protocol for spending a coin is the following:

Spending a coin:

- **U_1:** let T be the statement "there exist r, com, sig_{com} s. t. $com = E(c, \sigma, r)$ and $V_B(pk_B, com, sig_{com}) = 1$";
  use algorithm P_2, inputs r, com, sig_{com}, and σ to prove statement T;
  get as output $Proof$, a NIZK proof of knowledge of r, com, sig_{com} such that T is true;
  send $c, Proof$ to U_2.
- **U_2:** use algorithm V_2, statement T and σ to verify $Proof$;
  if all the verifications are successful then accept the coin c.

Observe that U_1 cannot directly pass sig_{com} to U_2, for otherwise the coin c would be later easily traceable from the Bank. On the other hand, he can prove the knowledge of a valid commitment com of some coin c and a valid signature sig_{com} for the commitment.

To transfer a coin c to another user U_3, user U_2 simply sends him the coin c and the string $Proof$ received by U_1.

The protocol to deposit a coin is the same than that for spending a coin, where a user U plays the role of the spender and the Bank that of the receiver.

The technique of signing a commitment in the protocol for withdrawing a coin and proving the knowledge of such a signature in the protocol for spending a coin has been used also in [9] to obtain an anonymous version of the NIZK proof system of knowledge with preprocessing given in [6].

5 Allowing dividability of coins

In this section we show how using the general paradigm for digital money described in the previous section and the cryptographic tools of pseudo-random functions and secure commitment schemes, it is possible to obtain the property of dividability of coins. This property has been first given in [20], using quadratic residues.

Let m be the (constant) number of different values that a coin can assume; that is, the value of a coin will be 2^k, for some $k \in \{0, \ldots, m-1\}$. Also, let $F = \{f_s, |s| = n\}$ a collection of pseudo-random functions, written on the public file PF.

Consider the protocol for withdrawing a coin: the Bank has to issue coins in such a way that at a later stage it will be possible for a user U_1 to divide the coin withdrawn in smaller fractions and thus to spend an arbitrary piece of this coin to a certain user U_2. Moreover, the user U_2 shall be able to do the same with the coin received.

To this end, we require that each owner U_1 of a coin c of value 2^k, for some $k \in \{0, \ldots, m-1\}$, can compute two coins c_1 and c_2 of value 2^{k-1}, and so on recursively. Thus, to the coin c one can associate a complete binary tree $T_{c,k}$ of height k in which each node is associated to a coin: that is, the root is associated to c, each of the 2 nodes at level 1 is associated to a coin of value 2^{k-1}, and each

of the 2^h nodes at level h is associated to a coin of value 2^{k-h}, for $h = 2, \ldots, k$. In order to reach our goal, the tree $T_{c,k}$ has to satisfy the following properties: a) before the execution of the spending protocol, U_1 is able to compute any coin he desires in the tree whose root is associated to c (this allows user U_1 to divide a coin owned into smaller fractions and thus pass any piece of it to a user U_2), and b) after U_1 has given a coin d at level k to U_2, U_2 is able to compute any coin he desires in the subtree whose root is associated to d (this allows U_2 to do the same with the coin received).

Let us informally describe how the protocol for withdrawing a coin is modified in order to obtain the property of dividability of coins. In order to withdraw a coin of value 2^k, a user U_1 randomly chooses $c \in \{0,1\}^n$ and computes $e_i = f_{s_i}(c)$, $r_i = f_{s_i}(\sigma)$, $c_i = e_i \circ s_{2i} \circ s_{2i+1}$ and $d_i = E(c_i, \sigma, r_i)$, where the s_i's are randomly chosen so that $s_i = s_{2i} \oplus s_{2i+1}$, for $i = 1, \ldots, 2^k - 1$. Then U_1 sends to the Bank a $(2^{k+1} - 1)$-tuple whose components represent the nodes of the complete binary tree $T_{c,k}$ of height k. Each node i at level h of $T_{c,k}$ is associated to a commitment d_i to the coin c_i of value 2^{k-h} that is represented by the concatenation of the following strings: e_i and two strings s_{2i}, s_{2i+1} that allow to decommit d_{2i}, d_{2i+1} respectively. In such a way, for each coin c, the associated tree $T_{c,k}$ allows to compute all possible subdivisions of c into coins of smaller values. Then, for each $i = 1, \ldots, 2^{k+1} - 1$, U_1 receives from the Bank a signature sig_{d_i} of the commitment d_i, the index i and the value v_i of the coin c_i, where sig_{d_i} is computed using the scheme (G_B, S_B, V_B). The knowledge of sig_{d_i} will represent the authorization from the Bank to spend the coin c_i of value v_i.

In our scheme, given the random string r_1, U_i can compute the random strings r_i used for the commitment at each node i of the tree $T_{c,k}$, and thus obtain all the coins that are possible subdivisions of the coin c requested. Moreover, as we will see later, to spend a part of c, say of value 2^{k-h}, a second user U_2 gives only the random string used for a commitment at a node at level h. It only remains to describe how a user computes the random strings r_i at each node i of a tree using the random string r_1 associated to the root. The commitment d_i in node i of the tree $T_{c,k}$ is computed using as random string $r_i = f_{s_i}(\sigma)$, where σ is the random reference string written on the public file PF, f_{s_i} is a pseudo-random function and s_i is a random string committed in the node i. Then, given s_i, a user can compute r_i, decommit the node i, and compute the indices s_{2i}, s_{2i+1} and the random strings r_{2i}, r_{2i+1}. Thus, knowing a coin c_i at a node i allows to compute coins c_{2i} and c_{2i+1} at its two children. On the other hand, knowing a coin c_{2i} at a node $2i$ does not allow to compute the coin c_i at node i.

Now we formally describe the protocol for withdrawing a coin of value 2^k.

Withdrawing a coin:

- **U:** randomly choose two n-bit strings c, s_1;
  for $i = 1, \ldots, 2^k - 1$,
  randomly choose the n-bit strings s_{2i} and compute $s_{2i+1} = s_i \oplus s_{2i}$;
  for $i = 1, \ldots, 2^{k+1} - 1$,

compute $e_i = f_{s_i}(c)$, $r_i = f_{s_i}(\sigma)$, $c_i = e_i \circ s_{2i} \circ s_{2i+1}$ and $d_i = E(c_i, \sigma, r_i)$;
send the tree $T_{c,k} = (d_1, \ldots, d_{2^{k+1}-1})$ to the Bank.

- **Bank:** for $i = 1, \ldots, 2^{k+1} - 1$,
  let v_i be the value of coin d_i;
  compute $sig_{d_i} = S_B(sk_B, d_i \circ i \circ v_i)$ and send sig_{d_i} to U.
- **U:** for $i = 1, \ldots, 2^{k+1} - 1$,
  if $V_B(pk_B, d_i \circ i \circ v_i, sig_{d_i}) = 1$ then accept the coin c_i.

To deal with the case of dividability of coins, the spending protocol is modified in the following way. A user U_1 has received an electronic coin c of value 2^k from another user (or from the Bank) and wants to give a coin c_i of value 2^h to user U_2. User U_1 sends the string s_i, the commitment d_i and $proof_i$, a NIZK proof of knowledge of c_i, of a signature of i, v_i and d_i. Also, user U_1 sends to user U_2 all the strings $s_l, d_l, proof_l$ that are associated to nodes in the subtree rooted at c_i. If all the proofs are convincing, then user U_2 accepts the coin c_i from U_1. Formally, the protocol is:

Spending a coin:

- $\mathbf{U_1}$: send c to U_2;
  for $j \in \{1, \ldots, 2^m - 1\}$ such that c_j is associated to a descendant of c_i,
  let T_j be the statement "there exist r_j, sig_{d_j} such that
  $d_j = E(f_{s_j}(c) \circ s_{2j} \circ s_{2j+1}, \sigma, r_j)$, $r_j = f_{s_j}(c)$
  and $V_B(pk_B, d_j \circ j \circ v_j, sig_{d_j}) = 1$";
  use algorithm P_2, inputs r_j, sig_{d_j}, and σ to prove statement T_j;
  get as output $Proof_j$, a NIZK proof of knowledge of r_j, sig_{d_j}
  such that T_j is true;
  send $s_j, d_j, Proof_j$ to U_2.
- $\mathbf{U_2}$: for $j \in \{1, \ldots, 2^m - 1\}$ such that c_j is associated to a descendant of c_i,
  use algorithm V_2, statement T_j and σ to verify $Proof_j$;
  if all the verifications are successful then accept the coin c.

The protocol to deposit a coin is similar to that for spending a coin, where a user U plays the role of the spender and the Bank that of the receiver. The only difference is that, instead of directly sending the commitment d_i, user U sends a non-interactive zero-knowledge proof of knowledge of d_i, so that the Bank cannot use d_i to trace coin c_i to the user which had originally withdrawn it.

6 Avoiding multiple spending of coins

In this section we consider the problem of avoiding multiple spending of coins in the general paradigm for digital money described in the previous sections.

First of all we see the case in which dividability of coins is not allowed. To avoid the double spending of a same coin, while spending a coin, a user should give

another message such that one of these messages gives no significant information on the spender, but two of these messages for a same coin give sufficient information to determine the author of the double spender. Nice solutions to this problem have been given in [6, 13, 12]. As it is possible to use these ideas for our general paradigm, we immediately get a way for avoiding double spending in our setting by a simple application of results in the cited papers. However, these ideas seem difficult to extend to the case of transferability of coins. Thus, we give a different technique based on signatures which, even if working under complexity assumptions (this is not the case for the cited techniques), allows to determine the author of a double spending also in the case in which the coin is transferred to many users before being deposited to the Bank. We observe that we have a sequence of spending protocols followed by a deposit protocol, and the Bank has to realize that a double spending has occurred only after the deposit protocol. Thus, to allow the Bank to discover the author of such a fraud, it must be the case that the user playing the role of spender of a certain coin has to commit to this action in some sense. This can be done in the following way: while spending a coin c, user U_1 also sends a signature of the coin and of the identity of the receiver of that coin. This signature can be seen as a commitment to the fact that user U_1 is passing the coin c to user U_2.

The modification to the protocols of previous sections are the following: while opening an account, each user U generates a pair (pk_U, sk_U) for his signature scheme (G_U, S_U, V_U) and writes pk_U on the public file PF. Then, while passing coin c to user U_2, user U_1 computes the signature $dsig_{U_1,U_2,c} = S_{U_1}(sk_{U_1}, c \circ ID_{U_2})$ and sends it to U_2. In this way, the Bank realizes that a double spending has occurred as she receives two signatures from different users of a same coin. Then, in order to discover the author of a double-spending, the bank runs the procedure Detect. In this procedure the Bank broadcasts a message stating that a double-spending of coin c has occurred (for example using the public file). To prove this, the Bank writes on the public file the two different signatures $dsig_{\cdot,\cdot,c}$ of a same coin c. At this point each user U_j that has received coin c in some spending protocols, sends to the Bank the signature $dsig_{U_i,U_j,c}$ received by some user U_i with this coin, thus proving that he has received the coin c by user U_i. In this way each user involved in this phase will reveal an identity of another user who has spent coin c and thus the identity of the author of the double-spending will be revealed twice to the Bank. More precisely, the procedure Detect is the following:

Procedure *Detect* $(c, dsig_1, dsig_2)$.

- **Bank:** Broadcast a message in which it is stated that a double-spending has occurred and users who have received the coin c in some spending protocol have to prove their honesty; write on the public file $c, dsig_1, dsig_2$.
- **Each user U_j:** If he has received the coin c in some spending protocol then send the signature $dsig$ received with that coin.

- **Bank:** verify that the signatures received are properly computed by running the corresponding verification algorithms; also, reconstruct the complete history of the coin c; if a user U sends a message not properly computed or the identity of some user U is received from two different users then U is the author of a double spending.

Let us now consider the case in which dividability of coins is allowed. In this case a dishonest U could spend too many coins contained in the tree $T_{c,k}$, that is a set of coins for which the sum of values is greater than 2^k. To avoid this, we impose that once a user has spent a coin c_i in the tree $T_{c,k}$ he cannot spend coins c_j in $T_{c,k}$ such that c_i and c_j belong to a same path starting from the root and finishing to a leave of $T_{c,k}$. Thus we say that a coin c_i is *spendable* by user U_1 if all paths from the root to any leave of $T_{c,k}$ and containing c_i do not contain coins that have already been spent by U_1. Also, we call *inconsistent* two coins associated to nodes in a same path from the root to a leave of a tree $T_{c,k}$.

Now, let us consider the spending protocol. Again, a user U_1 has received an electronic coin c of value 2^k from another user (or from the Bank) and wants to give a coin c_i of value 2^h to user U_2. The protocol is modified in the following way: first of all user U_1 chooses c_i as a spendable coin; then, while passing coin c_i to user U_2, U_1 computes the signature $dsig_{U_1,U_2,c_h} = S_{U_1}(sk_{U_1}, c_h \circ ID_{U_2})$, for each c_h in the subtree rooted at c_i, and sends it to U_2. Thus, a user spending correctly two inconsistent coins will pass two different signatures of a same coin, and so, also in this case, the Bank realizes that a double spending has occurred as she receives two different signatures of a same coin. In order to determine the author of such a double spending, the Bank proceeds exactly as before.

Remarks. Our solution allows transferability of coins without giving any increase in the size of the coins transferred. To obtain this result, our protocol cannot satisfy anonymous spending; that is, U_1's identity can be computed form the message sent by U_1 to U_2 in the protocol for spending a coin. On the other hand, if anonymous spending is required, the main result of [5] states that transferred coins grow in size. The time users have to keep the signatures received in the spending protocols depends from the implementation: for istance, if it is required that each coin must be deposited before the end of the day in which it has been issued, users keep their signatures for at most one day.

7 Proofs and properties

By properly putting together the procedures written in Sections 4, 5, 6, it is easy to construct the fourtuple $\mathcal{Q}$ =(OPEN, WITHDRAW, SPEND, DEPOSIT). Also, let $\mathcal{S} = \{Bank, U_1, \ldots, U_n\}$. In this section we give a sketch of proof that the pair $(\mathcal{S}, \mathcal{Q})$ is an off-line electronic cash system and see that the electronic cash system given satisfies also the properties of dividability and spending of pieces of coins and off-line spending of coins. First, we show that the four requirements of Definition 2 are satisfied.

No forging: Suppose that given the transcripts of k protocols $W_1,\ldots,W_k$, there exists an efficient algorithm *Adv* which computes $k+1$ coins $c_1,\ldots,c_{k+1}$ such that with nonnegligible probability for each $i = 1,\ldots,k+1$, $S=(Adv,S_2)\ (c_i,\cdot) = (\cdot, accept)$. Then the algorithm *Adv* can be used to efficiently compute a signature of a given message m without the knowledge of the secret key sk_B in the following way. On input m, generate coins $c_1,\ldots,c_k$ and run $S=(Adv,S_2)$ on input $(c_i,\cdot)$ for $i = 1,\ldots,k+1$, where $c_{k+1} = m$. The transcript of $S=(Adv,S_2)$ on input $(c_{k+1},\cdot)$ is a valid proof of knowledge of the signature of m. This can be used to contradict the properties of the proof system of knowledge (P,V) or of the signature scheme (G_B, S_B, V_B).

No tracing: Suppose that given the transcripts of k protocols $W_1,\ldots,W_k$ and of k protocols $D_1,\ldots,D_k$, there exists an efficient algorithm *Adv* which computes a coin c such that with probability at least $1/2+$ a nonnegligible term, for some $i,j \in \{1,\ldots,n\}$ and $l \in \{i,j\}$, it holds that $D=(D_1,Adv)(c,\cdot) = (\cdot, accept)$ and $W=(Adv,U_l)(\cdot,c) = (\cdot, accept)$; or $D=(D_1,Adv)(c,\cdot) = (\cdot, accept)$ and $S=(U_l,\cdot)(c_i,\cdot) = (\cdot, accept)$. Then the algorithm *Adv* can be used to efficiently compute with nonnegligible probability the identity of the withdrawer in some withdrawing protocol or that of the spender in some spending protocol of coin c. This contradicts the properties of the non-interactive zero-knowledge proof system of knowledge used.

No double spending: Suppose that there exists an efficient nonuniform algorithm *Adv* which computes a pair (c_1,c_2) of coins such that $S=(Adv,S_i)(c_i,\cdot) = (\cdot, accept)$ for $i = 1,2$. Then, a same coin is spent twice, and the Bank can realize that a double spending has occurred and identify the author of the double spending by running the procedure Detect described in the previous section. In fact, if a double spending has occurred, then a user U (by running algorithm *Adv*) has given a same coin c, respectively to a user V_1 and another user Z_1. Now, suppose that V_1 has given c to V_2, and so on until some V_h has given it to the Bank. Analogously, suppose that Z_1 has given the same coin to Z_2, and so on until some Z_k has given it to the Bank (where the V_i and the Z_j are not necessarily different). When running the procedure Detect, the Bank uses the signatures received to completely reconstruct the history of the coin c. In particular, she computes the two directed paths $V_h, V_{h-1},\ldots,V_1,U$ and $Z_k, Z_{k-1},\ldots,Z_1,U$ that have been taken (in the opposite direction) by the coin c during the spending protocols. Thus she recognizes the author of the double-spending U from the fact that he is the source of these two paths.

No framing: Suppose that given the transcripts of k protocols $W_1,\ldots,W_k$ and of k protocols $D_1,\ldots,D_k$, there exists an efficient algorithm *Adv* which computes a coin c such that for some $i,j \in \{1,\ldots,n\}$, computes $l \in \{i,j\}$ such that $S=(U_l,U_h)(c,\cdot) = (\cdot, c_1 \circ accept)$ and $S=(U_l,U_k)(c,\cdot) = (\cdot, c_2 \circ accept)$, for some $h,k \in \{1,\ldots,n\}$ with probability $1/2+$ a term nonnegligible in n, where c_1 and c_2 are the messages sent by U_l to U_k and U_h while spending coin c. Then the algorithm *Adv* can be used to efficiently compute a signature of a given message m without the knowledge of the secret key sk_{U_l}. This contradicts the

properties of the signature scheme $(G_{U_i}, S_{U_i}, V_{U_i})$.

Now we see that the electronic cash system given satisfies also the properties of dividability of coins, spending of pieces of coins and off-line spending of coins.

Dividability of coins and spending of pieces of coins: In our electronic cash system a coin c_i of value 2^h is the concatenation of the following strings: a string e_i computed as $f_{s_i}(c)$, where f_{s_i} is a pseudo-random function whose index s_i is known only to the owner of the coin c_i; and two strings s_{2i}, s_{2i+1}. These last strings are the indices of the pseudo-random functions that generate the random string $r_j = f_{s_j}(\sigma)$, for $j = 2i, 2i+1$ used to compute the commitments d_{2i}, d_{2i+1} respectively. Thus, given s_{2i}, s_{2i+1}, the owner of c_i can decommit d_{2i}, d_{2i+1} and compute the two coins c_{2i}, c_{2i+1} of value 2^{h-1} in which c_i can be divided. By repeating this process, given a coin c of value 2^k, any user U_i can obtain any coin of value 2^{k-h} as value, for $h = 1, \ldots, k$, and thus spend any coin of value i, for $i = 1, \ldots, 2^k$.

Off-line spending of coins: This property is immediately obtained thanks to the non-interactive proof used in the spending protocol of a coin.

Acknowledgements. Many thanks go to Alfredo De Santis, Tatsuaki Okamoto, Giuseppe Persiano and Moti Yung for helpful discussions.

References

1. M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge*, SIAM Journal of Computing, vol. 20, no. 6, Dec 1991, pp. 1084–1118.
2. M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications*, Proceedings of the 20th ACM Symposium on Theory of Computing, 1988, pp. 103–112.
3. S. Brands, *Untraceable Off-line Cash in Wallets with Observers*, in "Advances in Cryptology - CRYPTO 93", vol. 773 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 302–318.
4. D. Chaum, A. Fiat, and M. Naor, *Untraceable Electronic Cash*, in "Advances in Cryptology - CRYPTO 88", vol. 403 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 319–327.
5. D. Chaum and T. Pedersen, *Transferred Cash Grows in Size*, in "Advances in Cryptology - Eurocrypt 92", vol. 658 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 390–407.
6. A. De Santis and G. Persiano, *Communication Efficient Zero-Knowledge Proof of knowledge (with Application to Electronic Cash)*, in Proceedings of STACS 92, pp. 449–460.
7. A. De Santis and G. Persiano, *Zero-Knowledge Proofs of Knowledge Without Interaction*, Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science, 1992, pp. 427–436.
8. G. Di Crescenzo, *A Non-Interactive Electronic Cash System*, in Proceedings of Italian Conference on Algorithms and Complexity (CIAC 94), Springer Verlag.

9. G. Di Crescenzo, *Anonymous NIZK Proofs of Knowledge with Preprocessing*, manuscript.
10. W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transaction on Information Theory, vol. IT-22, no. 6, Nov. 1976. pp.644–654.
11. U. Feige, A. Fiat, and A. Shamir, *Zero-knowledge Proofs of Identity*, Journal of Cryptology, vol. 1, 1988, pp. 77–94.
12. N. Ferguson, *Single Term Off-Line Coins*, in "Advances in Cryptology - Eurocrypt 93", vol. 765 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 318–328.
13. M. Franklin and M. Yung, *Secure and Efficient Off-Line Digital Money*, in Proceedings of ICALP 93, vol. 700 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 265–276.
14. O. Goldreich, S. Goldwasser, and S. Micali, *How to Construct Random Functions*, Journal of the Association for Computing Machinery, vol. 33, no. 4, 1986, pp. 792–807.
15. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.
16. S. Goldwasser, S. Micali, and R. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack*, SIAM Journal of Computing, vol. 17, n. 2, April 1988, pp. 281–308.
17. S. Goldwasser and R. Ostrovsky, *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent*, in "Advances in Cryptology - CRYPTO 92", vol. 470 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 246–259.
18. M. Naor, *Bit Commitment using Pseudo-randomness*, in "Advances in Cryptology - CRYPTO 89", vol. 435 of "Lecture Notes in Computer Science", Springer-Verlag.
19. M. Naor and M. Yung, *Universal One-way Hash Functions and their Cryptographic Applications*, Proceedings of 21st ACM Symposium on the Theory of Computing, 1989.
20. T. Okamoto and K. Ohta, *Universal Electronic Cash*, in "Advances in Cryptology - CRYPTO 91", vol. 576 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 324–337.
21. T. Okamoto and K. Ohta, *Disposable Zero-knowledge Authentications and their Applications to Untraceable Electronic Cash*, in "Advances in Cryptology - CRYPTO 89", vol. 435 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 481–496.
22. J. Rompel, *One-way Functions are Necessary and Sufficient for Secure Signatures*, Proceedings of the 22nd ACM Symposium on Theory of Computing, 1990, pp. 387–394.
23. M. Tompa and H. Woll, *Random Self-Reducibility and Zero-knowledge Interactive Proofs of Possession of Information*, Proceedings of 28th Symposium on Foundations of Computer Science, 1987, pp. 472–482.