Auditable, Anonymous Electronic Cash

(Extended Abstract)

Tomas Sander and Amnon Ta–Shma

International Computer Science Institute
1947 Center Street, Berkeley, CA 94704, USA
{sander,amnon}@icsi.berkeley.edu

Abstract. Most anonymous, electronic cash systems are signature-based. A side effect of this is that in these systems the bank has the technical ability to issue unreported, valid money. It has been noticed in the past that this may lead to a disaster if the secret key of the bank is compromised. Furthermore, the above feature prevents any effective monitoring of the system.
In this paper we build a fully anonymous, auditable system, by constructing an electronic cash system that is signature-free, and where the bank needs to have no secret at all. The security of the system relies instead on the ability of the bank to maintain the integrity of a public database. Our system takes a completely new direction for meeting the above requirements, and, in particular, it is the first to do so without the necessity of making individual transactions potentially traceable: payers enjoy unconditional anonymity for their payment transactions. The system is theoretically efficient but not yet practical.
Keywords: electronic cash, anonymity.

1 Introduction

Payment systems should be safe and sound. When we deposit money into a bank we believe the bank will honor its commitments to us, and even though individual banks may run into problems (e.g., because of bad investments they make) most bank accounts are insured by national governments protecting consumers from losses. In return the banking industry is highly regulated and monitored by governmental agencies. There are at least two sides to this monitoring. One is to ensure that transactions are executed correctly. The other is to check the financial stability of the bank where it is verified that banks reasonably control their risks in issuing credits and loans, investing their assets and their operational risks (including insider crime) [32, 31]. This monitoring becomes even more critical with electronic cash systems in which issuers can issue money themselves.

We call a system *auditable* if it allows to effectively control the money supply, i.e. if there can not be valid money that is not known to the auditor (we later give a more formal definition). An electronic cash system that has this property and also has a complete report of transactions (which is "standard" in monetary systems) allows monitoring the same way the banking system is monitored today. On the other hand, if this property is violated then monitoring may not reflect the actual reality. We believe that even just for that reason auditability is important and desirable. Unfortunately, not too much attention was given to this property and there are many systems (e.g. [10]) which violate it. I.e., in such systems the issuer can issue valid money that is not reported anywhere, and thus such a system can be "technically" functioning correctly, and apparently stable, while it is in fact insolvent.

Further abuses of anonymous systems exist. Some of these exploit another bad property that may exist in an electronic cash system, and which we call rigidity. We say a system is *non-rigid* if any withdrawn money (in a standard or a non-standard withdrawal transaction) can be later invalidated (see a formal definition in Section 4). We argue that auditable, non-rigid systems defend against most of the major known attacks and abuses of anonymous electronic cash systems. The challenge is to build a fully anonymous yet auditable and non-rigid system. Previous solutions to the problem use electronic cash systems with "escrow" agents that can revoke the anonymity of users or coins. Thus, they compromise the anonymity feature. We take a completely different direction and present a solution that gives users full statistical anonymity.

1.1 Our Solution

In our system there is no trustee and a users' privacy can never be compromised. Furthermore, our system is not signature based, and in fact it is "secret-free"; there is no secret key the bank needs to hold (and therefore nothing that can be stolen,e.g. by insiders!). Furthermore, in our system all relevant bank actions are public and publicly verifiable, and the system is also fully auditable in the same way banks are auditable today.

The security of the system relies on the ability of the bank to maintain the integrity of a public database, and the auditability property follows from the fact that all transactions (including issuing new coins) are made public and can not be forged.

Our system also protects from blackmailing attacks against the bank; the only way a blackmailer can force the bank to issue electronic coins that will be accepted as valid payments is by forcing the bank to add some public data. The bank (and the whole world) knows which data have been added during the blackmailing, and as our system is also non-rigid the situation can be later reversed. Furthermore, even if the bank is willing to follow all terms dictated by blackmailers, there is no way for the bank to issue non-rigid coins. This leaves, for example, kidnapers with the knowledge that once they free the hostage they are left with worthless coins. Certainly also the bank knows that paying the ransom can not lead to a release of the hostage. This loose/loose scenario will strongly discourage both players to use this non–rigid system in blackmailing scenarios. Our system also has the advantage that it can remain off line after a blackmailing attack on the bank has occurred once the system has been updated.

On the conceptual level it is the first system that simultaneously guarantees unconditional user anonymity together with strong protection against the blackmailing and the bank robbery attack, in which the bank's secret key for signing coins is compromised. Recall that previous work addressed potential criminal abuses of electronic cash systems mainly in the escrowed cash paradigm, where each user transaction is made potentially traceable by Trustee(s). As Camenisch, Piveteau and Stadler [17] put it, these type of systems "offer a compromise between the legitimate need for privacy protection and an effective prevention of misuse by criminals". Our system defends against these attacks without doing the "compromise" cited above, and this is achieved using a simple, straightforward approach.

This raises the question whether other significant system abuses like money laundering can be effectively prevented by still preserving unconditional user anonymity. That the latter is possible was shown by the authors in [36] where an amount–limited, strongly non–transferable payment system was suggested. Amount–limitedness and non-transferability assure that a money launderer can not obtain the large amounts of anonymous electronic cash that are typically involved and needed in money laundering activities, neither by withdrawing them from the bank, nor by buying them on a "black market" for anonymous electronic cash. Using the techniques developed in [36] the current system can also be made amount–limited and non-transferable. The combined system then strongly defends against blackmailing, bank robbery and money–laundering abuses while offering unconditional privacy for users in their transactions. We therefore believe that the need for escrowed cash systems should be reexamined.

1.2 Organization of the Paper

In Section 2 we describe several attacks on electronic payment systems and previous work in which they were defeated. In Section 3 we outline the ideas underlying our electronic cash system and show how it differs conceptually from previous systems offering anonymity that were blind signature based. In Section 4 we give a formal description of model and requirements for our system. In Section 5 we describe the building blocks which we use in the protocol. In Section 6 we present the protocols of our auditable anonymous system and point in Section 7 to possible directions how the efficiency of the system can be improved.

2 Previous Work

We start by describing some major attacks on anonymous electronic cash systems:

Bank robbery attack: Jakobsson and Yung [26] describe an attack where the secret key of the issuer is compromised and the thief starts counterfeiting money. The attack is especially devastating if no one will be able to detect that there is false money in the system until the amount of deposited money surpluses the amount of withdrawn money. Obviously, by that time the whole market is flooded with counterfeited money, and the system may collapse. The Group of Ten report from 1996 [14] expresses "serious concern": "one of the most significant threats to an electronic money system would be the theft or compromising of the issuer's cryptographic keys by either an inside or an outside attacker." In the 1998 report [33] it is stated that "of direct concern to supervisory authorities is the risk of criminals counterfeiting electronic money, which is heightened if banks fail to incorporate adequate measures to detect and deter counterfeiting. A bank faces operational risk from counterfeiting, as it may be liable for the amount of the falsified electronic money balance." It further states "Over the longer term, if electronic money does grow to displace currency to a substantial degree, loss of confidence in a scheme could conceivably have broader consequences for the financial system and the economy."

Money laundering attacks: There are many possible ways to abuse an anonymous electronic cash system for money laundering (cf. e.g. [2]).

Blackmailing attack: Van Solms and Naccache [38] described a "perfect" blackmailing scenario that exploits anonymity features of (blind) signature-based electronic cash systems.

An auditor should not have to trust the issuer because an issuer can make profits from unreported money in several ways, e.g. for offering not properly backed credits or by assisting in money laundering activities (for documented cases in which financial institutions indeed assisted in unlawful activities see e.g. [1] and also [20]).

These types of attacks motivated a substantial amount of research started in [13] and [16], where electronic cash with revocable anonymity ("escrowed cash") was suggested. In, e.g. [26, 28, 15, 19, 34, 18], several systems following this approach have been described [1]. These cash systems allow a Trustee(s) to revoke the anonymity of each individual transaction. A typical revocability feature is "coin tracing" where the coin withdrawn by a user during a particular withdrawal session can be identified. This feature allows to defeat the blackmailing attack in the case a *private* user is blackmailed (as he may later ask to trace and blacklist the blackmailed coins).

Few systems of the ones mentioned above protect also against (the stronger) blackmailing attacks on the bank: a blackmailer may force the bank to enter a non–standard withdrawal protocol to withdraw coins (and thereby disable coin tracing mechanisms) or extort the bank's secret key. In the related bank robbery attack the secret key of the bank is stolen. Only the systems in [26, 19, 34, 25] prevent against these very strong latter attacks. Some of these systems require a third party involvement at withdrawal time and some can not remain off-line after such an attack had occurred.

[1] It has been pointed out before in [28] that systems like the one described in [37] are not vulnerable to several of these attacks as their security (for the bank) does not critically rely on (blind) signature techniques. Although the system [37] offers privacy to a certain extent it is not fully anonymous. We are not aware of a previous description of an electronic cash system that achieves full anonymity but does not rely on blind signatures techniques.

3 The Basic Idea

3.1 Stamps and Signatures

As stated above, whereas most previous systems offering anonymity are signature-based and the issuer gives (blinded) signatures to coins, ours is not. Our approach and its difference to previous ones can be best understood by considering the "membership in a list" problem. In this problem a bank holds a list of values $L = \{x_1, \ldots, x_k\}$, The elements in the list "correspond" to valid coins (and will be hash values of their serial numbers). If $x \in L$ there is a short proof of membership, whereas if $x \notin L$ it is infeasible to find such a proof, i.e. only when a user has a valid coin he can give such a proof, else he can not.

Let us now focus on the membership problem. One possible solution is to keep a public list of all values in L. However, such a solution requires the verifier to be able to access a large file. Another solution is for the bank to *sign* each value in L and to accept an element as belonging to L iff it carries a signature, which from an abstract point of view is how systems like, e.g., [10] work. Now, however, the security of the system relies on the secrecy of the secret key of the bank and the system becomes potentially vulnerable to blackmailing attacks. A third solution for the membership problem was suggested by Merkle [27]. In Merkle's solution, all values $x_1, \ldots, x_k$ are put into the leaves of a tree, and a hash tree is formed over the leaves using a collision resistant function h (for more details see Section 6). The root of the hash tree is made public, and the assumption is that this data is authenticated and its integrity can be maintained. A proof that x is in the list amounts to presenting a hash chain from x to the root. The collision resistant property of h then guarantees that it is infeasible to find a membership proof for an element $x \notin L$.

When the bank adds a value to the tree it "authenticates" or "stamps" the value as being valid for payment. Stamping is very similar to signing: Anyone who has access to the authenticated data (the root of the tree) can validate a stamp (by checking the hash chain leading to the root) which is similar to the public verification predicate in signature schemes. Also, it can be achieved that it is infeasible to forge neither signatures nor stamps. The key difference, from our perspective, between signatures and stamps, is that the secret/public key pair required in a signature scheme is "replaced" with authenticated public data (the root of the tree) which is exactly what we need for solving the bank robbery attack, the blackmailing attack and achieving auditability. Although the security requirements are different our system has technically some similarities to time stamping protocols [24, 8, 3] that also made use of tree based methods.

Our protocol uses Merkle's solution in a way that also allows to incorporate full anonymity, detection of double spenders and full auditability. Other features (as non-transferability and amount limitedness) can be easily added.

3.2 On the Update of Roots and Hash Chains

Let us say for the ease of explanation that a time frame lasts a minute, an hour has two minutes, a day has two hours, a week has two days, etc. At the first minute the bank creates a minute tree with all the coins issued at that minute on its leaves. At the second minute another minute tree is formed. When the second minute ends the two minute tress are combined into an hour tree by hashing the two roots of the minute trees together. During the next hour a new hour tree is used. When the second hour ends we combine the two hour trees to a day tree, and a new day tree is formed, and so on. Altogether we have a forest. Let us say that a root is *alive* if it is a root in the forest, i.e. it is the root of one of the last hour,day,week etc tree. If our system is to cover one hundred real years, then the system has to have about $5,000,000$ roots ($100 \cdot 365 \cdot 24 \cdot 60 << 2^{26}$) and therefore 26 levels are necessary (with the bottom level covering minutes, the level above it hours etc.). In particular, at any given minute there are at most 26 live roots.

Each merchant should hold a subset of the live roots. A merchant can choose how often to update his list. If a merchant chooses to keep 20 live roots he needs to update his list every 2^6 minutes, and he can not accept coins that were issued in the last 2^6 minutes. He can choose to keep all 26 live roots (and therefore accept all issued coins) but then he needs to update his list every minute.

When a user withdraws a coin the bank sends him a hash chain from his coin to the root of the current tree, and each time the tree is combined with another tree the bank updates the chain so that it leads to the root of the combined tree. Altogether the bank sends the user 26 update messages. A payment transaction begins with the merchant sending the user the set of all live roots he knows. The user proves in a zero knowledge way that he knows a hash chain to one of the roots in the set.

We point out the following:

– The updates are independent of the actual transaction that takes place and the specific user. An update at the end of an hour should only contain the values of the roots of the last two minutes, an update at the end of a day should only contain the roots of the last two hours etc. As a result the updates can be broadcasted to every system participant.
– Each merchant can choose how often he makes the updates. The only disadvantage in making less updates is that coins that were issued within the last uncovered period can not be accepted by the merchant.
– When a user tries to spend a coin to a merchant who does not accept coins from the last k minutes, the only information the user reveals is that his coin was not withdrawn during the last k minutes.

We believe that such a system might offer a flexible and practical solution to anonymous off-line cash. We call the system off-line because if a merchant chooses to be updated only once a day he can certainly do so. The users can also spend their money without involving the bank. They suffer however from the disadvantage that at least at the beginning they need to be updated often. This can be improved if more recent roots are kept by the merchant.

3.3 On the Security of the System

A crucial property we need is that the integrity of the published root of the tree is maintained. This can be achieved by publishing the root in the New York Times, or mirroring it in many different places, as was already pointed out in the original paper of Merkle.

Besides the authenticated data the bank keeps all sorts of data structures, including, e.g., the tree itself, a data structure carrying the balance of each account in the system, and more. It is clear that the bank can change these data structures, and in fact this can be done even in the banking system of today, where a bank can technically take all the money from one's account. Such accounting problems are well understood, and they are quite successfully dealt with in the banking industry. We do not deal with them in our system.

In an off-line system a merchant needs to verify membership to L reliably, i.e. he needs the correct roots. Criminals might establish a site that claims to contain the necessary authenticated data, and try to trick merchants into accepting a forged root. There are many different ways to address this problem, and furthermore such an attack is detected by the bank as soon as a forged coin is deposited. The bank can take a variety of steps immediately like shutting down the false source of the root, raising an alert or redistributing the correct root. The merchant can also set his own policy how to verify the authenticity of the root, and thereby manage his risk.

4 System

We first describe our model:
The participants: Users, merchants, a bank and the auditor.

Infrastructure: We assume there is an authenticated way for the bank to distribute the roots of its trees of issued coins.

Time: We assume that there are consecutive time frames denoted $T_1, T_2, \ldots$. We call the basic time frame a 'minute'. Two minutes are grouped together to an hour, two hours to a day and so forth.

Computing Power: All participants are probabilistic polynomial time players.

Trust Model: The network and the distribution channels are reliable and anonymous. Users and merchants trust the bank not to steal their money [2]. The auditor does not trust the bank not to issue unreported money.

System Events: We focus on the following system events:

- A user opens an account.
- A user withdraws money at the bank.
- The bank updates and broadcasts the roots of the forest.
- A user pays a coin to a merchant.
- A merchant deposits a received coin at the bank.

[2] By introducing receipts this trust requirement can be minimized using well known techniques. In this work we focus on the auditability property.

– The bank invalidates funds that have been withdrawn before.

We have the following requirements for our system:

Unforgeability: It is infeasible for any coalition of participants in the system excluding the bank to create an amount of payments accepted by the bank that exceeds the amount of withdrawn coins.

Auditability: There is a file, accessible by the auditor, that is supposed to describe all the events that have occurred in the system. In particular all withdrawals are supposed to be reported there. A withdrawal record should at least report who withdrew the money. We say c is a coin if it can be deposited, or if it was already successfully deposited.

Definition 1. *A system is* auditable *if there is a one to one correspondence between all coins c and the withdrawal records.*

We stress that we do not require that the one to one correspondence is known to the auditor or anyone else. In fact, in a truly anonymous system it is not. All we require is that such a correspondence exists. If the system is auditable, we also say the system does not admit any unreported money.

Non-Rigidness We say a system is non–rigid if any coin that can be accepted as a valid payment by the deposit protocol (and it does not matter whether it was withdrawn in a standard or non–standard transaction) can be invalidated.

Unconditional Payer Anonymity: A payer has unconditional anonymity, if transcripts of withdrawals are statistically uncorrelated to transcripts of payments and deposits.

We stress that at withdrawal time the user has to identify himself to the bank, and the bank might record the withdrawn string z along with the identity of its owner. Yet, as transcripts of withdrawals are statistically uncorrelated to transcripts of payments and deposits, this does not give the bank any information on how or to whom a withdrawn coin is spent.

5 Tools

Definition 2. *We say a function $f : A \times B \to C$ is* one-way*, if the probability a polynomial time machine given a random $c \in C$ can find (x, r) s.t. $f(x, r) = c$ is negligible. We say a function $f : A \times B \to C$ is* collision resistant*, if the probability a polynomial time machine can find $(x, r) \neq (x', r')$ s.t. $f(x, r) = f(x', r')$ is negligible.*

Definition 3. *Let G be a domain of size p. We say a function $g : [0..p-1] \times [0..p-1] \to G$ is concealing if for any $[0..p-1]$ the distribution $g(x, [0..p-1])$ obtained by picking $r \in [0..p-1]$ at random and computing $g(x, r)$ is the uniform distribution over G.*

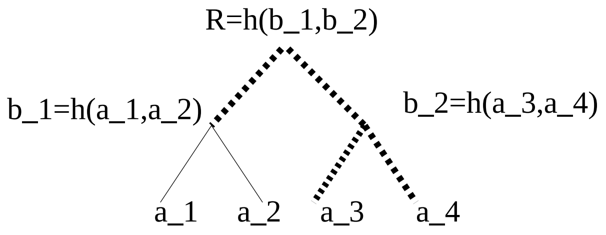

Fig. 1. *A hash chain* $((1,0);a_3;(a_4,b_1))$.

Assuming DLog is hard for certain groups of prime order G (see [9]) one-way, collision resistant and concealing functions exist and can be based on the representation problem [9]. More specifically, if G is a group of prime order p, for which $DLOG$ is hard, and g_1, g_2 are chosen at random (so almost always they are two distinct generators of G), then $g : [0..p-1] \times [0..p-1] \rightarrow G$ defined by $g(x,y) = g_1^x g_2^y$ has these properties (see [9] Proposition 7 and Corollary 8, for details).

5.1 Hash Chains and Hash Trees

Hash Chains. A hash chain of length 1 to a root R is a triplet $(i_1;x;y)$ s.t. $f^{(i_1)}(x,y) = R$, where $f^{(0)}(x,y) = h(x,y)$ and $f^{(1)}(x,y) = h(y,x)$.

A chain of length $d > 1$ to a root R is a triplet $((i_1,\ldots,i_d);x;(y_1,\ldots,y_d))$ s.t. $((i_1,\ldots,i_{d-1});f^{(i_d)}(x,y_d);(y_1,\ldots,y_{d-1}))$ is a hash chain of length $d-1$. We also say that the hash chain *starts* with the value x and leads to the root R. See Figure 1.

Hash Trees. For a given domain D, and a known hash function $h : D \times D \rightarrow D$ a hash tree (T, val) consists of a balanced binary tree T, with vertices V, together with a function $val : V \rightarrow D$ s.t. for any vertex v with two children v_1 and v_2, $val(v) = h(val(v_1), val(v_2))$. The only operation that can be performed on a hash tree is $UPDATE(leaf, w)$ where the leaf's value is changed to w and the values of the internal nodes from the leaf to the root are accordingly updated.

5.2 Non-interactive Zero Knowledge Arguments of Knowledge under the Random Oracle Assumption

We will frequently use perfect zero knowledge arguments of knowledge (ZKA) i.e., proofs that show that the prover *knows* a witness w to the predicate ϕ (i.e. $\phi(w) = True$). These proof are convincing if the prover is polynomially bounded, and the proofs *statistically* do not reveal extra information. The notion of a proof of knowledge is from [23,5]. Under the discrete log assumption any NP predicate has a perfect zero knowledge argument of knowledge ([11,12,22,21], see also [29] for zero knowledge arguments under weaker conditions and [4] for further details on ZKA's of knowledge).

We will need *non-interactive* perfect zero knowledge arguments (ZKA) of knowledge. We make the random oracle assumption [6] that has been commonly used in the design of electronic cash systems. Assuming the random oracle assumption, and using the techniques of Bellare and Rogaway [6], the zero knowledge argument of knowledge protocols can be made non-interactive (See [6] for the treatment of the similar case of zero-knowledge proofs).

6 An Anonymous Auditable Electronic Cash System

We now build an auditable, electronic cash system that allows users to have unconditional anonymity.

When Alice withdraws a coin, she chooses x and r (which she keeps secret) and sends $z = g(x, r)$ to the Bank. x should be thought of as the serial number of the coin, r is a random number and g is concealing and collision resistant. The bank adds the coin z to the public list of coins, yet only a person who knows a pre-image (x, r) of z can use z for payment, and because g is one-way only Alice can use the coin z. To make the system anonymous for Alice when she wants to spend a coin z she proves with a zero knowledge argument that she *knows* a pre-image (x, r) of some z that appears in the list of coins, without actually specifying the value z. To prevent double–spending extra standard mechanisms are added to the system.

As it turns out the most expensive part of the protocol is the zero knowledge argument of membership where Alice has to prove she knows a coin z from the public list of coins that has certain properties. A straight forward implementation of this would require a proof that is polynomially long in the number of coins. A better, and theoretically efficient, protocol for testing membership in a list was suggested by Merkle [27] using hash trees, and our system builds upon this solution. We now give a more formal description of the protocol.

6.1 The Protocol

Setup: During system setup bank and auditor (and possibly users) choose jointly the following objects. F_q is a field of size $q = poly(N)$ and N is an upper bound on the number of coins the bank can issue. G is a group of prime order p for which $DLOG$ is hard, and $|G| \geq q^3$. Further an efficient 1-1 embedding $E : F_q^3 \rightarrow [0..p-1]$ is chosen (e.g., by using binary representations of the elements in F_q and concatenating them). $g : [0..p-1] \times [0..p-1] \rightarrow G$ is a one-way, collision resistant and concealing function.

D is a large domain, $|D| \geq |G|$, and $h : D \times D \rightarrow D$ a collision resistant hash function. An efficient 1-1 embedding $F : G \rightarrow D$ is chosen. The bank keeps a hash tree T over D with N leaves. This hash tree is gradually built. There is no need to initialize the tree.

Each merchant obtains a unique identifying identity, and we assume the existence of a random oracle $\mathcal{H} : TIME \times ID \rightarrow F_q$ that maps time and an id to a random element of F_q. We assume that each merchant can do at most one transaction per time unit (this time unit can be chosen to be very short). Alternatively the merchant has to add a serial number to each transaction occurring at the same time unit and is not allowed to use the same serial number twice.

Account Opening: When Alice opens an account, Alice has to identify herself to the bank, and the bank and Alice agree on a public identity $P_A \in F_q$ that uniquely identifies Alice.

Withdrawal: Alice authenticates herself to the bank. Alice picks $u_1 \in_R F_q$, $serial \in_R F_q$ and computes $u_2 = P_A - u_1 \in F_q$ and $x = (u_1, u_2, serial) \in F_q^3$. $serial$ is the serial number of the coin and u_1, u_2 are used to encode Alice's identity. She also picks $r \in_R [0..p-1]$ and sends $z = F(g(E(x), r)) \in D$. to the bank. She gives the bank a non-interactive zero knowledge argument that she knows $u_1, u_2, serial$ and r s.t. $z = F(g(E(u_1, u_2, serial), r))$ and $u_1 + u_2 = P_A$, i.e, that the coin is well formed. The bank also makes sure that the coin z has not been withdrawn before. See Figure 2.

Fig. 2. Withdrawal

Alice Bank

$u_1 \in_R F_q$, $u_2 = P_A - u_1 \in F_q$
$serial \in_R F_q, r \in_R [0..p-1]$
$z = F(g(E(u_1, u_2, serial), r))$

$\xrightarrow{P_A, z}$

Alice gives a non-interactive
ZKA that z is well formed
$\longrightarrow$

In the zero knowledge argument Alice has to answer challenges. These challenges are determined in the non-interactive ZKA protocol using the random oracle $\mathcal{H}$.

The bank then subtracts one dollar from Alice's account, and updates one of the unused leaves in the tree T to the value z (along with the required changes to the values along the path from the leaf to the root). When the time frame ends the bank takes a snapshot of the tree T and creates a *version*. After creating the version the bank sends Alice the hash chain from z to the root of T, taken from the hash tree T. Alice checks that she was given a hash chain from z to the public root of the hash tree T.

Updates: Each minute a new minute tree is generated, and a version of it is taken at the end of the minute. When two minute versions exist, they are combined together to an 'hour' tree, by hashing the two roots together. Similarly, if two hour trees exist, they are combined together to a day tree and so forth.

At the end of each hour,day, week etc. a broadcast message is sent to all users who withdrew a coin during that time period. The hour update, e.g., contains the values of the two minute roots that were hashed together to give the hour tree root. Merchants can follow their own updating policy.

Payment: Alice wants to make a payment with a coin $z = F(g(E(u_1, u_2, serial), r))$ to a merchant M with identity m_{id}. The protocol starts with M sending Alice the set $ROOTS$ of live roots he knows. A root is alive if it is the root of the tree of the last minute, hour,day etc. Alice then sends the merchant $serial, time$ and the value $v = u_1 + cu_2$, where $1 \neq c = \mathcal{H}(time, m_{id})$. She then proves to the merchant with a non-interactive ZKA that she knows u_1, u_2, r, R and a hash chain $((i_1, \ldots, i_d); w; (y_1, \ldots, y_d))$ to R s.t.

- $R \in ROOTS$,
- $w = F(g(E(u_1, u_2, serial), r))$ and
- $v = u_1 + cu_2$.

The merchant verifies the correctness of the non-interactive ZKA. See Figure 3.

Fig. 3. Payment

Alice		Merchant
	$\xleftarrow{ROOTS}$	
$c = \mathcal{H}(time, m_{id}) \in F_q$		
$v = u_1 + cu_2 \in F_q$	$\xrightarrow{time,v,serial}$	
	Alice gives non-interactive ZKA that she knows a valid hash chain with the above $c, v, serial, ROOTS$ $\longrightarrow$	

Deposit: The merchant transfers the payment transcript to the bank. The bank checks that the merchant with identity m_{id} has not earlier deposited a payment transcript with this particular parameter $time$. The bank verifies that the challenges are correct (i.e., they are $\mathcal{H}(time, m_{id})$), that the set $ROOTS$ in the payment transcript consists of valid roots, and that the non-interactive ZKA is correct. The bank then checks whether $serial$ has already been spent before. If not the bank credits the merchant's account and records $serial \in F_q$ as being spent along with the values $c \in F_q$ and $v(= u_1 + cu_2) \in F_q$.

If $serial$ has been spent before the bank knows two different linear equations $v_1 = u_1 + c_1u_2$ and $v_2 = u_1 + c_2u_2$. The bank solves the equations to obtain u_1 and u_2, and $P = u_1 + u_2$. The bank then finds the user with the public identity P.

Invalidation of funds: The bank removes the coins that should be invalidated from the forest and recomputes the corresponding roots and the hash chains for each leaf in the forest. The bank distributes the updated snapshot of the forest and sends the updated hash chains for each of the withdrawn coins in the forest to the user who withdrew it.

6.2 Correctness

Theorem 1. *Under the random oracle assumption, if DLOG is hard, and the assumptions we made in the description of our model in Section 4 the electronic cash system is statistically anonymous, unforgeable, auditable, non–rigid and detects double spenders. The system is also efficient, and the time required from all players at withdrawal, payment and deposit time is polynomial in the* log *of the total number of coins N (i.e. the depth of the tree) and the security parameter of the system. Each user receives $logN$ messages per withdrawal.*

Proof. *Unforgeability*: We prove that any spent coin must appear as a leaf in T. To spend a coin Charlie needs to know a hash chain $((i_1, \ldots, i_d); w; (y_1, \ldots, y_d))$ to a root R of a valid tree T. Because h is collision resistant, and all the players are polynomial, they can not find two different chains, with the same $(i_1, \ldots, i_d)$ leading to the same value. Hence, it follows that the hash chain Charlie knows is exactly the path in T going down from the root according to $(i_1, \ldots, i_d)$. In particular w is a leaf of T.

Double Spending: If Charlie spends a coin z then this coin must have been withdrawn before, as was shown in the proof of the unforgeability property, let's say by Alice. When Alice withdrew z she proved she knew $u_1, u_2, serial$ and r s.t. $z = F(g(E(u_1, u_2, serial), r))$ and $u_1 + u_2 = P_A$. When Charlie spends the coin he proves he knows a hash chain that starts at z, and $u_1', u_2', serial', r'$ s.t. $F(g(E(u_1, u_2', serial'), r')) = z$. As g is collision resistant and E and F are 1-1, it follows that $u_1' = u_1, u_2' = u_2$ and $serial' = serial$. In particular, any time a coin z is spent, the serial number that is reported is always the same (the serial number chosen at withdrawal time). In particular double spent coins can be easily detected by observing the serial number.

When a coin is spent a challenge to $u_1' + cu_2'$ is answered and answered correctly (we have a non-interactive zero knowledge argument of correctness). Furthermore, as we have seen above the u_1', u_2' are always the same and they are identical to the u_1, u_2 chosen at withdrawal time. By the random oracle assumption, the assumption that each merchant has a distinct ID and that he can only do one transaction per time unit, no challenge is repeated twice. Hence if the same coin is spent twice, we see the answers to different challenges and we can easily solve the system of two equations to discover u_1 and u_2 and consequently $u_1 + u_2 = P$ which is the identity of the double spender.

Anonymity: The non-interactive zero knowledge arguments do not reveal any information in an information theoretic sense about Alice's inputs beyond the fact that the coin is well formed (resp. the validity of the claim that is proved). We would like to show that the information at withdrawal time (i.e. P and z) is statistically independent of the information at deposit time ($c, v = u_1 + cu_2, serial$ and $ROOTS$). We first observe that even given $u_1, u_2, c, serial$ and $ROOTS$ the value $z = F(g(E(u_1, u_2, serial), r))$ is uniform, because g is concealing. We are left with P at withdrawal time and $c, v = u_1 + cu_2, ROOTS$ at deposit time. Since $u_2 = P - u_1$ we have $v = cP + (1-c)u_1$. As u_1 is uniformly distributed, so is v. Also, c is a function of time and merchant id (which are public data) and is independent of the coin.

The only possible information leakage can result from the subset $ROOTS$ the user uses. However, $ROOTS$ covers all the time periods acceptable by the merchant. Hence the deposit does not reveal more information then merely the fact that some user paid a merchant M a coin that was proved to be valid according to the update policy of the merchant. Thus, the system is anonymous. Merchants who try to provide users with a manipulated list $ROOTS$ can be detected by the user (e.g., if he knows a valid root not in the list that should be there) and will definitely be caught by the bank (or the auditor) when they check the payment transcripts.

Auditability: We have already shown that if a polynomial time player can spend a coin c then it must have appeared as a leaf in the tree. As all the leaves in the tree are different, this shows that a one to one correspondence between usable coins and leaves in the tree (and therefore withdrawals) exists.

Non–Rigidness: We have seen that a coin z is only usable for payment if z appears as a leaf in the tree. Furthermore, the bank and the auditor know who withdrew z. To invalidate z all the bank has to do is replace z with a NULL, and update the tree, the hash chains and the roots as described in the system event "invalidation of funds".

Efficiency:

Withdrawal, Payment, Deposit: The dominating complexity stems from the zero knowledge arguments. Each non-interactive ZKA of an NP statement takes resources polynomial in the length of the NP statement. The way we described the protocol, the user claims knowledge of a valid root, and a hash chain leading to that root. Thus, the NP statement is of length $O((\log(N)+|ROOTS|)\log(|D|)$, (A root among all possible roots is a claim of length $O(|ROOTS|\log(|D|)$, and a hash chain is a sequence of $O(\log(N))$ elements each of length $O(\log(|D|)))$. Thus, altogether the statement size is $O(\log^2 N)$.

Updates and Invalidation: Whenever a minute, hour, day, year etc. ends, the bank has to broadcast two root values to all the users who withdrew money during that period. E.g., at the end of the year the bank has to update all customers who withdrew money during that year. Minute updates are done frequently but involve only few users, year updates happen only once a year but involve many users. Each user will ultimately receive $\log N$ update messages each containing two root values. We point out that the bank can avoid notifying the users about year (and all infrequent) updates by simply publishing the two roots in the New York Times or a similar medium. It should also be noted that once a broadcasting technology becomes widely accessible, the complexity of updates is only a constant.

Invalidation takes $O(N)$ applications of the hash function to update the tree, and messages should be sent to all users. Unlike regular updates this update involves the whole tree (with $O(N)$ data items) and can not be done by broadcast. However, invalidation should not happen too often as mentioned above, and is mainly used to deal with extreme cases where the bank surrendered to blackmailing.

7 Some Comments

Other protocols for testing membership in a list exist. In particular, the protocol suggested by Benaloh and de Mare [7] using one-way accumulators, has the property that it produces hash chains of length 1. The later protocol, however, suffers from the disadvantage that the coalition of parties which generates the one–way accumulator h knows a trapdoor for h. Employing the protocol in our payment system will reduce the length of the zero knowledge proofs to a constant independent of the number of withdrawn coins. However a person who knows the trapdoor, can completely bypass the system's security and pay with counterfeited money. Thus, if an honest agency constructs h and then destroys the trapdoor the system is safe. But at least during system setup the payment system is still vulnerable to the bank robbery attack as the trapdoor may be compromised. We consider this a serious disadvantage [3].

We leave open the question of designing a practically efficient system that is both anonymous and auditable. We believe that by designing an appropriate accumulator–like hash function and the efficient corresponding zero knowledge proofs this should be possible. A first step in this direction was taken in [35], where efficient accumulators without trapdoor are constructed.

Finally, we ask if there are conceptually different ways to design anonymous payment systems which have the auditability property, and how existing anonymous electronic payment systems can be transformed into auditable ones with minimal performance overhead.

Acknowledgments

We would like to thank Moni Naor and Moti Yung for stimulating conversations and Mike Just for a having pointed out a reference to us.

[3] Benaloh and de Mare leave open whether one-way accumulators can be built without using a trapdoor. Nyberg [30] builds such a system, with an accumulated hash of length $N \log(N)$, where N is the number of items to be hashed. The large hash length makes this protocol completely irrelevant to our case.

References

1. FATF-VII report on money laundering typologies. Financial Crimes Enforcement Network Publications, August 1996. http://www.treas.gov/fincen/pubs.html.
2. Cyberpayments and money laundering. RAND, 1998. http://www.rand.org/publications/MR/MR965/MR965.pdf/.
3. D. Bayer, S. Haber, and W. S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In R. M. Capocelli et al., editor, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. SV , New York, 1992.
4. Bellare, Jakobsson, and Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 1997.
5. M. Bellare and O. Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer-Verlag, 1993, 16–20 August 1992.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A pardigm for designing efficient protocols. In Victoria Ashby, editor, *1st ACM Conference on Computer and Communications Security*, Fairfax, Virginia, November 1993. ACM Press. also appeared as IBM RC 19619 (87000) 6/22/94.
7. J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. *Lecture Notes in Computer Science*, 765:274–??, 1994.
8. J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Sciences. August, 1991.
9. S. Brands. An efficient off-line electronic cash system based on the representation problem. In *246*. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, December 31 1993. AA (Department of Algorithmics and Architecture), CS-R9323, URL=ftp://ftp.cwi.nl/pub/CWIreports/AA/CS-R9323.ps.Z.
10. S. Brands. Untraceable off-line cash in wallet with observers. In Douglas R. Stinson, editor, *Crypto 93*, volume 773 of *LNCS*, pages 302–318. SV, 1993.
11. Brassard and Crepeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 1986.
12. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.
13. E. Brickell, P. Gemmell, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 457–466. Sandia National Labs, January 1995.
14. Report by the Committee on Payment, Settlement Systems, and the Group of Computer Experts of the central banks of the Group of Ten countries. Security of electronic money. Publication of the Bank for International Settlements, Basle, August 1996. http://www.bis.org/publ/index.htm.
15. J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. *Lecture Notes in Computer Science*, 1146:33–, 1996.
16. J. Camenisch, J. Piveteau, and M. Stadler. Fair blind signatures. In L. Guilloy and J-J. Quisquater, editors, *EuroCrypt 95*, LNCS, pages 209–219. SV, 1995.

17. J. Camenisch, J. Piveteau, and M. Stadler. An efficient fair payment system. In Clifford Neuman, editor, *3rd ACM Conference on Computer and Communications Security*, pages 88–94, New Delhi, India, March 1996. ACM Press.
18. G. Davida, Y. Frankel, Y. Tsiounis, and Moti Yung. Anonymity control in E-cash systems. In Rafael Hirschfeld, editor, *Financial Cryptography: First International Conference, FC '97*, volume 1318 of *Lecture Notes in Computer Science*, pages 1–16, Anguilla, British West Indies, 24–28 February 1997. Springer-Verlag.
19. E. Fujisaki and T. Okamoto. Practical escrow cash system. *Lecture Notes in Computer Science*, 1189:33–??, 1997.
20. General Accounting Office (GAO). Private banking: Raul Salinas, Citibank, and alleged money laundering. GAO/OSI-99-1, December 1998. http://www.gao.gov/monthly.list/dec98/dec9811.htm.
21. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.
22. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In A. M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1987, 11–15 August 1986.
23. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.
24. Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.
25. M. Jakobsson and J. Muller. Improved magic ink signatures using hints. In *proceedings of Financial Cryptography '99. Forthcoming volume in Lecture Notes in Computer Science*, 1999.
26. M. Jakobsson and M. Yung. Revokable and versatile electronic mony. In Clifford Neuman, editor, *3rd ACM Conference on Computer and Communications Security*, pages 76–87, New Delhi, India, March 1996. ACM Press.
27. R. Merkle. Protocols for public key cryptosystems. In IEEE, editor, *Proceedings of the 1980 Symposium on Security and Privacy, April 14–16, 1980 Oakland, California*, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1980. IEEE Computer Society Press.
28. D. M'Raihi. Cost-effective payment schemes with privacy regulation. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 266–275, Kyongju, Korea, 3–7 November 1996. Springer-Verlag.
29. M. Naor, R. Ostrovsky, Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11, 1998.
30. K. Nyberg. Fast accumulated hashing. In Dieter Grollman, editor, *Fast Software Encryption: Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 83–87, Cambridge, UK, 21–23 February 1996. Springer-Verlag.
31. Report of the working party on electronic money of the Group of Ten countries. Electronic money - consumer protection, law enforcement, supervisory and cross border issues. Publication of the Bank for International Settlements, Basle, April 1997. http://www.bis.org/publ/index.htm.
32. Basle Committee on Banking Supervision. Core principles for effective banking supervision. Publication of the Bank for International Settlements, Basle, September 1997. http://www.bis.org/publ/index.htm.

33. Basle Committee on Banking Supervision. Risk management for electronic banking and electronic money activities. Publication of the Bank for International Settlements, Basle, March 1998. http://www.bis.org/publ/index.htm.
34. H. Peterson and G. Poupard. Efficient scalable fair cash with off-line extortion prevention. *Lecture Notes in Computer Science*, 1334:463–??, 1997.
35. T. Sander. Efficient accumulators without trapdoor. Manuscript, 1999.
36. T. Sander and A. Ta-Shma. Flow control: A new approach for anonymity control in electronic cash systems. In *proceedings of Financial Cryptography '99. Forthcoming volume in Lecture Notes in Computer Science*, 1999.
37. Daniel R. Simon. Anonymous communication and anonymous cash. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 61–73. Springer-Verlag, 18–22 August 1996.
38. S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, October 1992.