Balancing Accountability and Privacy Using E-Cash (Extended Abstract)

Jan Camenisch[1] and Susan Hohenberger[1,⋆] and Anna Lysyanskaya[2]

[1] IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon
[2] Computer Science Department, Brown University, Providence, RI 02912, USA

Abstract. In an electronic cash (e-cash) system, a user can withdraw coins from the bank, and then spend each coin anonymously and unlinkably. For some applications, it is desirable to set a limit on the dollar amounts of anonymous transactions. For example, governments require that large transactions be reported for tax purposes. In this work, we present the first e-cash system that makes this possible without a trusted party. In our system, a user's anonymity is guaranteed so long as she does not: (1) double-spend a coin, or (2) exceed the publicly-known spending limit with any merchant. The spending limit may vary with the merchant. Violation of either condition can be detected, and can (optionally) lead to identification of the user and discovery of her other activities. While it is possible to balance accountability and privacy this way using e-cash, this is impossible to do using regular cash.
Our scheme is based on our recent compact e-cash system. It is secure under the same complexity assumptions in the random-oracle model. We inherit its efficiency: 2^ℓ coins can be stored in $O(\ell + k)$ bits and the complexity of the withdrawal and spend protocols is $O(\ell + k)$, where k is the security parameter.

1 Introduction

Electronic cash (e-cash) was invented by David Chaum [18, 19]. Its main goal is to match the untraceability properties of physical coins: the same bank is responsible for dispensing e-cash to users, and for later accepting it for deposit from merchants, and yet it cannot trace how users spent their money.

An important difference between electronic cash and physical cash, is that electronic cash is represented by data. Data is easy to duplicate, while physical coins may be made of precious metals so the cost to minting them is high. Therefore, an e-cash scheme must incorporate a way to ensure that an electronic coin (e-coin) cannot be spent more than once (double-spent). Typically [20] this is done by ensuring that, even though spending a coin once does not leak any information about a user's identity, spending it twice leaks information that leads to identification. An e-cash scheme with such a mechanism is a good illustration of how one can balance anonymity with accountability: a user can remain

⋆ Part of work performed at the Massachusetts Institute of Technology.

anonymous unless she performs a forbidden action. The system is designed in a way that prevents this type of anonymity abuse.

In this paper we consider what other actions may be forbidden, and how to realize e-cash schemes that would hold users accountable should they perform such actions. At the same time, we protect the anonymity of those users who obey the rules.

We introduce the *bounded-anonymity business model*. In this model, associated with each merchant there is a publicly-known limit to the number of e-coins that a user may anonymously give to this merchant. This limit cannot be exceeded even if the user and the merchant collude. Should any user attempt to exceed the limit with any merchant, and should this merchant attempt to submit the resulting e-coins for deposit to the bank, the user's identity will be discovered, and further penalties may be imposed.

In the real world, this corresponds to restrictions that governments set on unreported transactions. For example, in the U.S., banks are required by law to log and report all transactions over $10,000. These restrictions are set up to ensure proper taxation and to prevent money laundering and other monetary frauds. Another example application is an anonymous pass with usage limitations. For example, consider the following amusement park pass: "This pass is good to enter any Disney park up to four times, with the restriction that the Magic Kingdom can be entered at most twice." Until now, it was not known how to realize such passes anonymously.

Interestingly, in the real world it is impossible to set such restrictions on cash transactions. A merchant may be required by law to report that he received a lot of money in cash, but he may choose not to obey the law! In contrast, we show that with e-cash, it is possible to enforce the bounded-anonymity business model. The cost for achieving this is roughly double the cost of achieving regular anonymous e-cash.

There have been several previous attempts to solve this problem, but until now it remained an elusive open problem in electronic cash, as well as one of the arguments why the financial community resisted any serious deployment of e-cash, due money laundering regulations.

Some of the past efforts suggested using a trusted third party to mitigate this problem [40, 9, 31]. This TTP could trace transactions to particular users. The TTP approach is undesirable. First of all, the whole idea of electronic cash is to ensure that no one can trace e-cash transactions. Secondly, in these past solutions, the only way that a TTP can discover money laundering or other violations of the bounded-anonymity model is by tracing each transaction, which is very expensive. In a variant that reduces the trust assumption about the TTP, Kügler and Vogt [30] propose an e-cash scheme where the bank has the ability to trace coins by specially marking them during the withdrawal protocol. This tracing is auditable, i.e., a user can later find out whether or not his coins were traced (this involves an additional trusted judge). Still, this system does not allow to discover money laundering, unless it involves the marked coins, and the user must still trust the judge and bank for her anonymity. Another variant [35, 29] prevents money laundering by offering users only a limited form of anonymity. Users' coins are anonymous, but linkable, i.e., coins from the same user can be identified as such. Here it is easy to detect if a user exceeds the spending limit with some merchant. However, this weak form of anonymity is not suitable for all applications, and goes against the principle of e-cash.

Another set of papers [41, 34] addressed a related problem of allowing a user to show a credential anonymously and unlinkably to any given verifier up to k times. They give a nice solution, but it is not clear how it can be applied to *off-line* electronic cash as opposed to on-line anonymous authentication. I.e., showing an anonymous credential in their scheme more than k times allows a verifier to link the $k + 1$ *st* show to a previous transaction, but does not lead to the identification of the misbehaving user. In contrast, in our scheme, any such violation leads to identification of the user even if the verifier (merchant) colludes with the user.

Finally, Sander and Ta-Shma [37] propose to limit money laundering by dividing time into short time periods and issuing at most k coins to a user per time period (a user can deposit his unspent coins back into his account). This way, a user cannot spend more than k coins in a single transactions because he has at most k coins at any given time.

Our contribution. We present the first e-cash scheme in the bounded-anonymity business model. A user may withdraw, and anonymously and unlinkably spend an unlimited number of coins, so long as she does not: (1) double-spend a coin, or (2) exceed the spending limit with any merchant. Our scheme allows to efficiently detect either of these violations. We also show how to augment it so as to allow to reveal the identity of the misbehaving user. Finally, in addition to identifying a misbehaving user, one is also able to trace all of the user's previous e-coins.

Our construction takes as a starting point the e-cash system of Camenisch, Hohenberger, and Lysyanskaya (CHL) [11], which is the most efficient known. The cost of our resulting withdrawal and spend protocols is roughly double that of CHL. The size of the coin storage remains the same, but we also require the user to store a counter for each merchant with whom the user does business, which appears to be optimal. Thus we maintain CHL's asymptotic complexity: 2^ℓ coins can be stored in $O(\ell + k)$ bits and the complexity of the withdrawal and spend protocols is $O(\ell + k)$, where k is the security parameter.

2 Definition of Security

We now generalize the definition of CHL [11] to handle violations beyond double-spending. Our offline e-cash scenario consists of the three usual players: the user, the bank, and the merchant; together with the algorithms $\mathsf{BKeygen}$, $\mathsf{UKeygen}$, $\mathsf{Withdraw}$, Spend, $\mathsf{Deposit}$, $\{\mathsf{DetectViolation}^{(i)}, \mathsf{IdentifyViolator}^{(i)}, \mathsf{VerifyViolation}^{(i)}\}$, Trace, and $\mathsf{VerifyOwnership}$. Informally, the key generation algorithms $\mathsf{BKeygen}$ and $\mathsf{UKeygen}$ are for the bank and the user, respectively. A user interacts with the bank during $\mathsf{Withdraw}$ to obtain a wallet of 2^ℓ coins; the bank stores optional tracing information in database D. In Spend, a user spends one coin from his wallet with a merchant; as a result the merchant obtains the serial number S of the coin, the merchant record locator V of the coin, and a proof of validity π. In $\mathsf{Deposit}$, whenever an honest merchant accepted a coin $C = (S, V, \pi)$ from a user, there is a guarantee that the bank will accept this coin for deposit. The bank stores $C = (S, V, \pi)$ in database L. At this point, however, the bank needs to determine if C violates any of the system conditions.

For each violation i, a tuple of algorithms $\{\mathsf{DetectViolation}^{(i)}, \mathsf{IdentifyViolator}^{(i)}, \mathsf{VerifyViolation}^{(i)}\}$ is defined. Here, we have two violations.

Violation 1: Double-spending. In $\mathsf{DetectViolation}^{(1)}$, the bank tests if two coins, $C_1 = (S_1, V_1, \pi_1)$ and $C_2 = (S_2, V_2, \pi_2)$, in L have the same *serial number* $S_1 = S_2$. If so, the bank runs the $\mathsf{IdentifyViolator}^{(1)}$ algorithm on (C_1, C_2) and obtains the public key pk of the violator and a proof of guilt Π. Anyone can run $\mathsf{VerifyViolation}^{(1)}$ on (pk, S_1, V_1, Π) to be convinced that the user with public key pk double-spent the coin with serial number S_1.

Violation 2: Money-laundering. In $\mathsf{DetectViolation}^{(2)}$, the bank tests if two coins, $C_1 = (S_1, V_1, \pi_1)$ and $C_2 = (S_2, V_2, \pi_2)$, in L have the same *merchant record locator* $V_1 = V_2$. If so, the bank runs the $\mathsf{IdentifyViolator}^{(2)}$ algorithm on (C_1, C_2) and obtains the public key pk of the violator and a proof of guilt Π. Anyone can run $\mathsf{VerifyViolation}^{(2)}$ on (pk, S_1, V_1, Π) to be convinced that the user with public key pk exceeded the bounded-anonymity business limit with the coin with merchant record locator V_1.

Optionally, after any violation, the bank may also run the Trace algorithm on a valid proof of guilt Π to obtain a list of all serial numbers S_i ever spent by the cheating user, with public key pk, along with a proof of ownership Γ. Anyone can run $\mathsf{VerifyOwnership}$ on (pk, S_i, Γ) to be convinced that the user with public key pk was the owner of the coin with serial number S_i.

Security. We generalize the security definition of CHL for e-cash [11]. Their formalizations of correctness, balance, and anonymity of users remain unchanged. Roughly, *balance* guarantees that an honest bank will never have to accept for deposit more coins than users withdrew, while *anonymity of users* assures users that they remain completely anonymous unless they violate one of the known system conditions. We now informally describe three additional properties. These properties are generalizations of CHL's identification and tracing of double-spenders, and their exculpability, to apply to any specified violation, in particular those above. Let *params* be the global parameters, including the number of coins per wallet, 2^ℓ, and a (possibly unique) spending limit for each merchant. (Recall that each merchant may have a different spending limit, but that a merchant's limit will apply uniformly for all of its customers.)

Identification of violators. Suppose two coins $C_1 = (S_1, V_1, \pi_1)$ and $C_2 = (S_2, V_2, \pi_2)$ are the output of an honest merchant (or possibly merchants) running two Spend protocols with the adversary or they are two coins that an honest bank accepted for deposit. This property guarantees that, with high probability, if, for some i, the algorithm $\mathsf{DetectViolation}^{(i)}(\mathit{params},\ C_1, C_2)$ accepts, then

$\mathsf{IdentifyViolator}^{(i)}(params, C_1, C_2)$ outputs a key pk and a proof Π such that $\mathsf{VerifyViolation}^{(i)}(params, pk, S_1, V_1, \Pi)$ accepts.

Tracing of violators. Suppose $\mathsf{VerifyViolation}^{(i)}(params, pk, S, V, \Pi)$ accepts for some violation i derived from coins C_1, C_2. This property guarantees that, with high probability, $\mathsf{Trace}(params, pk, C_1, C_2, \Pi, D)$ outputs the serial numbers $S_1, \ldots, S_m$ of all coins belonging to the user of pk along with proofs of ownership $\Gamma_1, \ldots, \Gamma_m$ such that for all j, we have that $\mathsf{VerifyOwnership}(params, pk, S_j, \Gamma_j)$ accepts.

Exculpability. Suppose an adversary participates any number of times in the Withdraw protocol with an honest user with key pk, and subsequently to that, in any number of *non-violation* Spend protocols with the same user. The adversary then outputs an integer i, a coin serial number S, and a purported proof Γ that the user with key pk committed violation i and owns coin S. The *weak* exculpability property states that, for all adversaries, the probability that $\mathsf{VerifyOwnership}(params, pk, S, \Gamma)$ accepts is negligible.

Furthermore, the adversary may continue to engage the user in Spend protocols, forcing her to violate the system conditions. The adversary then outputs (i, S, V, Π). The *strong* exculpability property states that, for all adversaries: (1) when S is a coin serial number *not* belonging to the user of pk, weak exculpability holds, and (2) when the user of pk did *not* commit violation i, the probability that $\mathsf{VerifyViolation}^{(i)}(params, pk, S, V, \Pi)$ accepts is negligible.

The formal definitions follow in a straight-forward manner by applying the above intuition to the CHL definitions [11].

3 Technical Preliminaries

Our e-cash system use a variety of known protocols as building blocks, which we now briefly review. Many of these protocols can be shown secure under several different complexity assumptions, a flexibility that will extend to our e-cash systems. **Notation:** we write $G = \langle g \rangle$ to denote that g generates the group G.

3.1 Bilinear Maps

Let Bilinear_Setup be an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear mapping as $\gamma = (q, g_1, h_1, \mathbb{G}_1, g_2, h_2, \mathbb{G}_2, \mathbb{G}_T, e)$ [6]. Each group $\mathbb{G}_1 = \langle g_1 \rangle = \langle h_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle = \langle h_2 \rangle$, and $\mathbb{G}_T$ are of prime order $q = \Theta(2^k)$. The efficiently computable mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is both: (*Bilinear*) for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_q$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$; and (*Non-degenerate*) if g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, then $e(g_1, g_2)$ generates $\mathbb{G}_T$.

3.2 Complexity Assumptions

The security of our scheme relies on the same assumptions as CHL, which are:

Strong RSA Assumption [3, 27]: Given an RSA modulus n and a random element $g \in \mathbb{Z}_n^*$, it is hard to compute $h \in \mathbb{Z}_n^*$ and integer $e > 1$ such that $h^e \equiv g \mod n$. The modulus n is of a special form pq, where $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes.

y-Decisional Diffie-Hellman Inversion Assumption (y-DDHI) [4, 25]: Given a random generator $g \in G$, where G has prime order q, the values $(g, g^x, \ldots, g^{(x^y)})$ for a random $x \in \mathbb{Z}_q$, and a value $R \in G$, it is hard to decide if $R = g^{1/x}$ or not.

External Diffie-Hellman Assumption (XDH) [28, 39, 32, 5, 2]: Suppose $\mathsf{Bilinear\_Setup}(1^k)$ produces the parameters for a bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The XDH assumption states that the Decisional Diffie-Hellman (DDH) problem is hard in $\mathbb{G}_1$. This implies that there does *not* exist an efficiently computable isomorphism $\psi' : \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Sum-Free Decisional Diffie-Hellman Assumption (SF-DDH) [24]: Suppose that $g \in G$ is a random generator of order q. Let L be any polynomial function of $|q|$. Let $O_{\boldsymbol{a}}(\cdot)$ be an oracle that, on input a subset $I \subseteq \{1, \ldots, L\}$, outputs the value $g_1^{\beta_I}$ where $\beta_I = \prod_{i \in I} a_i$ for some $\boldsymbol{a} = (a_1, \ldots, a_L) \in \mathbb{Z}_q^L$. Further, let R be a predicate such that $R(J, I_1, \ldots, I_t) = 1$ if and only if $J \subseteq \{1, \ldots, L\}$ is DDH-independent from the I_i's; that is, when $v(I_i)$ is the L-length vector with a one in position j if and only if $j \in I_i$ and zero otherwise, then there are no three sets I_a, I_b, I_c such that $v(J) + v(I_a) = v(I_b) + v(I_c)$ (where addition is bitwise over the integers). Then, for all probabilistic polynomial time adversaries $\mathcal{A}^{(\cdot)}$,

$$Pr[\boldsymbol{a} = (a_1, \ldots, a_L) \leftarrow \mathbb{Z}_q^L; (J, \alpha) \leftarrow \mathcal{A}^{O_{\boldsymbol{a}}}(1^{|q|}); y_0 = g^{\prod_{i \in J} a_i}; y_1 \leftarrow G;$$
$$b \leftarrow \{0, 1\}; b' \leftarrow \mathcal{A}^{O_{\boldsymbol{a}}}(1^{|q|}, y_b, \alpha) \;:\; b = b' \wedge R(J, Q) = 1] < 1/2 + 1/\mathrm{poly}(|q|),$$

where Q is the set of queries that $\mathcal{A}$ made to $O_{\boldsymbol{a}}(\cdot)$.

3.3 Key Building Blocks

Known Discrete-Logarithm-Based, Zero-Knowledge Proofs. In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [38] or a composite [27, 23], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [21] or composite [15] moduli, (3) proof that a commitment opens to the product of two other committed values [14, 16, 8], (4) proof that a committed value lies in a given integer interval [17, 14, 7], and also (5) proof of the disjunction or conjunction of any two of the previous [22]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption. We can apply the Fiat-Shamir heuristic [26] to turn such proofs of knowledge into *signature proofs of knowledge* on some message m.

DY Pseudorandom Function (PRF). Let $G = \langle g \rangle$ be a group of prime order q. Let s be a random element of $\mathbb{Z}_q$. Dodis and Yampolskiy [25] recently proposed a pseudorandom function $f_{g,s}^{DY}(x) = g^{1/(s+x)}$ for inputs $x \in \mathbb{Z}_q^*$. This construction is secure under the y-DDHI. [3]

Pedersen Commitments. Pedersen proposed a perfectly-hiding, computationally-binding commitment scheme [36] based on the discrete logarithm assumption, in which the public parameters are a group of prime order q, and generators $(g_0, \ldots, g_m)$. In order to commit to the values $(v_1, \ldots, v_m) \in {\mathbb{Z}_q}^m$, pick a random $r \in \mathbb{Z}_q$ and set $C = \mathrm{PedCom}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$. Fujisaki and Okamoto [27] showed how to expand this scheme to composite order groups.

CL Signatures. Camenisch and Lysyanskaya [12] came up with a secure signature scheme based on the Strong RSA assumption with two protocols: (1) An efficient protocol between a user and a signer with keys (pk_S, sk_S). The common input consists of pk_S and C, a Pedersen commitment. The user's secret input is the set of values $(v_1, \ldots, v_\ell, r)$ such that $C = \mathrm{PedCom}(v_1, \ldots, v_\ell; r)$. As a result of the protocol, the user obtains a signature $\sigma_{pk_S}(v_1, \ldots, v_\ell)$ on his committed values, while the signer does not learn anything about them. The signature has size $O(\ell \cdot \log q)$. (2) An efficient proof of knowledge of a signature protocol between a user and a verifier. The common inputs are pk_S and a commitment C. The user's private inputs are the values $(v_1, \ldots, v_\ell, r)$, and $\sigma_{pk_S}(v_1, \ldots, v_\ell)$ such that $C = \mathrm{PedCom}(v_1, \ldots, v_\ell; r)$. These signatures are secure under the strong RSA assumption. For our current purposes, it does not matter *how* CL signatures actually work, all that matters are the facts stated above.

Verifiable Encryption. We use a technique by Camenisch and Damgård [10] for turning any semantically-secure encryption scheme into a verifiable encryption scheme. A verifiable encryption scheme is a two-party protocol between a prover and encryptor $\mathcal{P}$ and a verifier and receiver $\mathcal{V}$. Roughly, their common inputs are a public encryption key pk and a commitment A. As a result of the protocol, $\mathcal{V}$ either rejects or obtains the encryption c of the opening of A. The protocol ensures that $\mathcal{V}$ accepts an incorrect encryption only with negligible probability and that $\mathcal{V}$ learns nothing meaningful about the opening of A. Together with the corresponding secret key sk, transcript c contains enough information to recover the opening of A efficiently. We hide some details here and refer to Camenisch and Damgård [10] for the full discussion.

Bilinear Elgamal Encryption. In particular, we apply the verifiable encryption techniques above to a bilinear variant of the Elgamal cryptosystem [6, 1], which is semantically secure under an assumption implied by either y-DDHI or Sum-Free DDH. What we will need is a cryptosystem where g^x is sufficient for decryption and then the public key is $f(g^x)$ for some function f.

Assume we run $\mathsf{Bilinear\_Setup}$ on 1^k to obtain $\gamma = (q, g_1, h_1, \mathbb{G}_1, g_2, h_2, \mathbb{G}_2, \mathbb{G}_T, e)$, where we have bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. In bilinear Elgamal [1],

[3] It is possible to eliminate the y-DDHI assumption from our e-cash system by replacing the DY PRF with a DDH-based PRF such as the one due to Naor and Reingold [33]. However, this approach would enlarge our wallets from $\mathcal{O}(\ell + k)$ bits to $\mathcal{O}(\ell \cdot k)$ bits. Thus, we present only the most optimal building blocks.

a public-secret keypair is of the form $(e(g_1^u, g_2), g_1^u)$ for a random $u \in \mathbb{Z}_q$. Thus, we can think of $f(\cdot) := e(\cdot, g_2)$ where the value g_1^u is enough to decrypt.

4 Compact E-Cash in the Bounded-Anonymity Model

Overview of our construction. As in the CHL compact e-cash scheme, a user withdraws a wallet of 2^ℓ coins from the bank and spends them one by one. Also, as in the CHL scheme, we use a pseudorandom function $F_{(\cdot)}(\cdot)$ whose range is some group G of large prime order q.

At a high level, a user forms a wallet of $2^\ell = N$ coins by picking five values, (x, s, t, v, w) from an appropriate domain to be explained later, and running an appropriate secure protocol with the Bank to obtain the Bank's signature σ on these values.

Suppose that the user wants to spend coin number i by buying goods from merchant M. Suppose that only up to K transactions with this merchant may be anonymous. Let's say that this is the user's j-th transaction with M, $j \leq K$. Associated with the i-th coin in the wallet is its serial number $S = F_s(i)$. Associated with the j-th transaction with the merchant M is the merchant's record locator $V = F_v(M, j)$.

The first idea is that in the Spend protocol, the user should give to the merchant the values (S, V), together with a (non-interactive zero-knowledge) proof that these values are computed as a function of (s, i, v, M, j), where $1 \leq i \leq N$, $1 \leq j \leq K$, and (s, v) correspond to a wallet signed by the Bank. Note that S and V are pseudorandom, and therefore computationally leak no information; and the proof leaks no information because it is zero-knowledge.

Suppose that a user spends more than N coins. Then he must have used some serial number more than once, since there are only N possible values S of the form $F_s(i)$ where $1 \leq i \leq N$. (This is the CHL observation.) Similarly, suppose that a user made more than K transactions with M. Then he must have used some merchant record locator more than once, since for a fixed M, there are only K different values $V = F_v(M, j)$, $1 \leq j \leq K$. Therefore it is easy to see that double-spending and violations of the bounded-anonymity business model can be detected.

Now we need to explain how to make sure that using any S or V more than once leads to identification. Remember that besides s and v, the wallet also contains x, t and w. The value $x \in \mathbb{Z}_q$ is such that g^x is a value that can be publicly linked to the user's identity. (Where g is a generator of the group G.) For example, for some computable function f, $f(g^x)$ can be the user's public key. Suppose that as part of the transaction the merchant contributes a random value $r \neq 0$, and the user reveals $T = g^x F_t(i)^r$ and $W = g^x F_w(M, j)^r$, together with a proof that T and W are computed appropriately as a function of (r, x, t, i, w, M, j) corresponding to the very same wallet and the same i and j. Again, T and W are pseudorandom and therefore do not leak any information.

If a user uses the same serial number $S = F_s(i)$ twice, and q is appropriately large, then with high probability in two different transactions she will receive different r's, call them r_1 and r_2, and so will have to respond with $T_1 = g^x F_t(i)^{r_1}$, $T_2 = g^x F_t(i)^{r_2}$. It is easy to see that the value g^x can then be computed as follows: $g^x = T_1/(T_1/T_2)^{r_1/(r_1-r_2)}$. This was discovered by CHL building on the original ideas of offline e-cash [20].

We show that it is also the case that if the user uses the same merchant's record locator number V twice, then g^x can be found in exactly the same fashion. Suppose that in the two transactions the merchant used the same r. In that case, the Bank can simply refuse to deposit this e-coin (since it is the same merchant, he is responsible for his own lack of appropriate randomization). So suppose that the merchant used two different r's, r_1 and r_2, giving rise to W_1 and W_2. It is easy to see that $g^x = W_1/(W_1/W_2)^{r_1/(r_1-r_2)}$.

Thus, a double-spending or a violation of the bounded-business model leads to identification. The only remaining question is how this can be adapted to trace other transactions of the same user. Note that g^x is not necessarily a public value, it may also be the case that only $f(g^x)$ is public, while knowledge of g^x gives one the ability to decrypt a ciphertext which was formed by verifiably encrypting s (for example, Boneh and Franklin's cryptosystem [6] has the property that g^x is sufficient for decryption). When withdrawing a wallet, the user must give such a ciphertext to the bank. In turn, knowledge of s allows to discover serial numbers of all coins from this wallet and see how they were spent.

Finally, note that the values (x, v, w) should be tied to a user's identity and not to a particular wallet. This way, even if a user tries to spend too much money with a particular merchant from different wallets, it will still lead to detection and identification.

4.1 Our Protocols

Recall our building blocks from Section 3: the Dodis-Yampolskiy pseudo-random function [25], i.e., $f^{DY}_{(g,s)}(x) = g^{1/(s+x)}$, where g is the generator of a suitable group; CL-signatures [12] and the related protocols to issue signatures and prove knowledge of signatures; and the Bilinear Elgamal cryptosystem [6, 1] used with the Camenisch-Damgård [10] verifiable encryption techniques.

Notation: Let $F_{(g,s)}(x) = f^{DY}_{(g,s)}(x)$, and when H is a hash function whose range is an appropriate group, let $G^H_s(M, x) = f^{DY}_{(H(M),s)}(x)$.

We are now describing the protocols of our system: Setup, $\mathsf{Withdraw}$, Spend, and $\mathsf{Deposit}$ (including the protocols in response to violations).

Setup*:* Let k be the security parameter. The common system parameters are the bilinear map parameters $\mathsf{Bilinear\_Setup}(1^k) \rightarrow (q, g_1, \mathbb{G}_1, g_2, h_2, \mathbb{G}_2, \mathbb{G}_T, e)$, a wallet size ℓ, and two hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_T$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$. The bank generates CL signing keys $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ as before.

Each user generates a key pair of the form $sk_{\mathcal{U}} = (x, v, w)$ and $pk_{\mathcal{U}} = (e(g_1, h_2)^x, e(g_1, h_2)^v,\ e(g_1, h_2)^w)$, where x, v, and w are chosen randomly from $\mathbb{Z}_q$. Each user also generates a signing keypair for any secure signature scheme. Each merchant publishes a unique identity string $id_{\mathcal{M}}$. Also, an upper-bound $N_{\mathcal{M}}$ for the number of coins each user can spend with merchant $id_{\mathcal{M}}$ is fixed.

Withdraw: A user $\mathcal{U}$ withdraws 2^ℓ coins from the bank $\mathcal{B}$ as follows. The user and the bank engage in an interactive protocol, and if neither report an error, then at the end:

1. $\mathcal{U}$ obtains (s, t, σ), where s, t are random values in $\mathbb{Z}_q$, and σ is the bank's signature on $(sk_{\mathcal{U}}, s, t)$, i.e., (x, v, w, s, t).
2. $\mathcal{B}$ obtains a verifiable encryption of s under $e(g_1, h_2)^x$, i.e., the first element from the user's public key $pk_{\mathcal{U}}$, together with the user's signature on this encryption.
3. $\mathcal{B}$ does not learn anything about $sk_{\mathcal{U}}$, s, or t.

Step one can be efficiently realized using the Camenisch-Lysyanskaya signatures and the related protocols [13]. Step two can be realized by applying the Camenisch-Damgård [10] verifiable encryption techniques to the Bilinear Elgamal cryptosystem [6, 1]. Step three follows from the other two. All these steps are essentially the same as in the CHL e-cash scheme, the exception being the secret key signed which now also includes v and w besides x.

Spend: A user $\mathcal{U}$ spends one coin with a merchant $\mathcal{M}$ with a spending limit of $N_{\mathcal{M}}$ coins as follows. As in CHL, the user keeps a private counter i from 1 to 2^ℓ for the number of coins spent in her wallet. Additionally, the user now also keeps a counter $j_{\mathcal{M}}$ for each merchant $\mathcal{M}$ representing the number of coins she has spent with that merchant.

1. $\mathcal{U}$ checks that she is under her spending limit with merchant $\mathcal{M}$; that is, that $j_{\mathcal{M}} < N_{\mathcal{M}}$. If not, she aborts.
2. $\mathcal{M}$ sends random $r_1, r_2 \in \mathbb{Z}_q^*$ to $\mathcal{U}$.
3. $\mathcal{U}$ sends $\mathcal{M}$ the i-th coin in her wallet on her $j_{\mathcal{M}}$-th transaction with $\mathcal{M}$. Recall that $sk_{\mathcal{U}} = (x, v, w)$. This coin consists of a serial number S and a wallet check T, where

$$S = F_{(e(g_1,h_2),s)}(i) = e(g_1, h_2)^{1/(s+i)}\ ,\ T = g_1^x(F_{(g_1,t)}(i))^{r_1} = g_1^{x+r_1/(t+i)}$$

and two money laundering check values V and W, where

$$V = G_v^{H_1}(id_{\mathcal{M}}, j_{\mathcal{M}}) = H_1(id_{\mathcal{M}})^{1/(v+j_{\mathcal{M}})}\ ,$$
$$W = g_1^x(G_w^{H_2}(id_{\mathcal{M}}, j_{\mathcal{M}}))^{r_2} = g_1^x H_2(id_{\mathcal{M}})^{r_2/(w+j_{\mathcal{M}})}$$

and a zero-knowledge, proof of knowledge (ZKPOK) π of $(i, j_{\mathcal{M}}, sk_{\mathcal{U}} = (x, v, w), s, t, \sigma)$ such that
   (a) $1 \le i \le 2^\ell$;
   (b) $1 \le j_{\mathcal{M}} \le N_{\mathcal{M}}$;
   (c) $S = F_{(e(g_1,h_2),s)}(i)$, i.e., $S = e(g_1, h_2)^{1/(s+i)}$;
   (d) $T = g_1^x(F_{(g_1,t)}(i))^{r_1}$, i.e., $T = g_1^{x+r_1/(t+i)}$;
   (e) $V = G_v^{H_1}(id_{\mathcal{M}}, j_{\mathcal{M}})$, i.e., $V = H_1(id_{\mathcal{M}})^{1/(v+j_{\mathcal{M}})}$;
   (f) $W = g_1^x(G_w^{H_2}(id_{\mathcal{M}}, j_{\mathcal{M}}))^{r_2}$, i.e., $W = g_1^x H_2(id_{\mathcal{M}})^{r_2/(w+j_{\mathcal{M}})}$; and
   (g) VerifySig$(pk_{\mathcal{B}}, (sk_{\mathcal{U}} = (x, v, w), s, t), \sigma)$=true.

The proof π can be made non-interactive using the Fiat-Shamir heuristic [26].

4. If π verifies and the value V_j was never seen by $\mathcal{M}$ before, then $\mathcal{M}$ accepts and saves the coin $(r_1, r_2, S, T, V, W, \pi)$. If the value V_j was previously seen before in a coin $(r'_1, r'_2, S', T', V, W', \pi')$, then $\mathcal{M}$ runs $Open(W', W, r'_2, r_2)$. Let us define the $Open(\cdot,\cdot,\cdot,\cdot)$ algorithm as:

$$Open(A, B, C, D) := \frac{A}{(A/B)^{C/(C-D)}} \ .$$

   If $\mathcal{M}$ executed the Spend protocols honestly (i.e., chose fresh random values at the start of each protocol), then with high probability $r_2 \neq r'_2$, and $Open(W', W, r'_2, r_2) = g_1^x$. Thus, the merchant can identify the user by computing $e(g_1^x, h_2)$, which is part of $\mathcal{U}$'s public key. This allows an honest merchant to protect itself from customers who try to overspend with it. (If the merchant is dishonest, the bank will catch the overspending at deposit time.)

Steps 3(a,c,d) are the same as in the CHL scheme whereas Steps 3(b,e,f) are new, and Step 3(g) needs to be adapted properly. Consequently, Steps 3(a) and 3(b) can be done efficiently using standard techniques [17, 14, 7]. Steps 3(c) to 3(f) can be done efficiently using techniques of Camenisch, Hohenberger, and Lysyanskaya [11]. Step 3(g) can be done efficiently using the Camenisch and Lysyanskaya signatures [13].

Deposit*:* A merchant $\mathcal{M}$ deposits a coin with bank $\mathcal{B}$ by submitting the coin $(r_1, r_2, S, T, V, W, \pi)$. The bank checks the proof π; it if does not verify, the bank rejects immediately. Now, the bank must make two additional checks.

First, $\mathcal{B}$ checks that the spender of the coin has not overspent her wallet; that is, the bank searches for any previously accepted coin with the same serial number S. Suppose such a coin $(r'_1, r'_2, S, T', V', W', \pi')$ is found. If $r_1 = r'_1$, $\mathcal{B}$ refuses to accept the coin. Otherwise, $\mathcal{B}$ accepts the coin from the merchant, but now must punish the user who double spent.

1. $\mathcal{B}$ executes $Open(T', T, r'_1, r_1) = g_1^x$.
2. $\mathcal{B}$ identifies user as person with public key containing $e(g_1^x, h_2)$.
3. $\mathcal{B}$ uses g_1^x to decrypt the encryption of s left with the bank during the withdraw protocol. Next, $\mathcal{B}$ uses s to compute the serial numbers $S_j = F_{(e(g_1,h_2),s)}(j)$ for each coin $j = 1$ to 2^ℓ of all coins in the user's wallet. (In fact, the bank can use g_1^x to decrypt the secret of *all* the user's wallets and trace those transactions in the same way.)

Second, $\mathcal{B}$ checks that the spender of the coin has not exceeded her spending limit with merchant $\mathcal{M}$. That is, the bank searches for any previously accepted coin with the same money-laundering check value V_j. Suppose such a coin $(r'_1, r'_2, S', T', V, W', \pi')$ is found. The bank immediately refuses to accept the deposit and punishes the merchant. The bank now must also determine if the spender is to blame. If $r_2 = r'_2$, $\mathcal{B}$ punishes the merchant alone. Otherwise, $\mathcal{B}$ must also punish the user who attempted to money launder.

1. $\mathcal{B}$ executes $Open(W', W, r_2', r_2) = g_1^x$.
2. $\mathcal{B}$ identifies user as person with public key containing $e(g_1^x, h_2)$.
3. $\mathcal{B}$ uses g_1^x to decrypt the encryption of s left with the bank during the withdraw protocol. Next, $\mathcal{B}$ uses s to compute the serial numbers $S_j = F_{(e(g_1,h_2),s)}(j)$ for each coin $j = 1$ to 2^ℓ of all coins in the user's wallet. (In fact, the bank can use g_1^x to decrypt the secret of *all* the user's wallets and trace those transactions in the same way.)

If all checks pass, $\mathcal{B}$ accepts the coin for deposit in $\mathcal{M}$'s account.

The deposit protocol is again very similar to the deposit protocol of the CHL scheme, i.e., instead of only checking for double spending, the bank now also checks for money laundery. Thus, if the user was honest, the bank needs now to perform two database lookup's instead of one before.

For completeness, we point out explicitly how the violation-related protocols work. Let $C_1 = (r_1, r_2, S, T, V, W, \pi)$ and $C_2 = (r_1', r_2', S', T', V', W', \pi')$ be one existing and one newly deposited coin. Detecting double-spending or money-laundering involves checking $S_1 = S_2$ or $V_1 = V_2$, respectively. The identification algorithm runs *Open* on the appropriate inputs, and the resulting proof of guilt is $\Pi = (C_1, C_2)$. Verifying the violation entails successfully checking the validity of the coins, detecting the claimed violation, running *Open* to obtain g_1^x, and checking its relation to pk. (Recall that knowledge of x, not just g_1^x, is required to create a valid coin. Thus the leakage of one violation cannot be used to spend the user's coins or fake another violation.) The trace algorithm involves recovering s, from the encryption E signed by the user during Withdraw, and computing all serial numbers. The proof of ownership $\Gamma = (E, \sigma, g_1^x)$, where σ is the user's signature on E. Verifying ownership for some serial number S involves verifying the signature σ, checking that $e(g_1^x, h_2) = pk$, decrypting E to recover s, computing all serial numbers S_i, and testing if, for any i, $S = S_i$.

Theorem 1. *In the bounded-anonymity business model, our scheme achieves correctness, balance, anonymity of users, identification of violators, tracing of violators, and strong exculpability under the Strong RSA, y-DDHI, and either the XDH or Sum-Free DDH assumptions in the random oracle model.*

Due to space limitations, we refer to the full version of this paper for the proof of Theorem 1. We briefly provide some informal intuition.

Balance. For each wallet, s deterministically defines exactly 2^ℓ values that can be valid serial numbers for coins. To overspend a wallet, a user must either use one serial number twice, in which case she is identifiable, or she must forge a CL signature or fake a proof of validity.

Anonymity of users. A coin is comprised of four values (S, T, V, W), which are pseudorandom and thus leak no information about the user, together with a non-interactive, zero-knowledge proof of validity, which since it is zero-knowledge also leaks nothing. The only abnormality here is that, when computing V and W, the base used for the PRF is the hash of the merchant's identity (as opposed to the fixed bases used to compute S and T). Treating hash H as a random oracle, we see that given $G_v^H(id_{\mathcal{M}}, j)$, the output of $G_v^H(\cdot,\cdot)$ on any other input, in particular $G_v^H(id'_{\mathcal{M}}, j)$ for $id_{\mathcal{M}} \neq id'_{\mathcal{M}}$, is indistinguishable from random. Specifically, if an adversary given $G_v^H(id_{\mathcal{M}}, j) = f^{DY}_{(H(id_{\mathcal{M}}),v)}(j) = H(id_{\mathcal{M}})^{1/(v+j)}$ can distinguish $H(id'_{\mathcal{M}})^{1/(v+j)}$ from random for some random, fixed $H(id_{\mathcal{M}})$ and $H(id'_{\mathcal{M}})$, then it is solving DDH.

Exculpability. First, an honest user cannot be proven guilty of a crime he didn't commit, because the proof of guilt includes the user's *secret* value g_1^x. If a user is honest, only he knows this value. Second, even a cheating user cannot be proven guilty of a crime he didn't commit— e.g., double-spending one coin does not enable a false proof of money-laundering twenty coins —because: (1) guilt is publicly verifiable from the coins themselves, and (2) knowledge of x is required to create coins. The value g_1^x, which is leaked by a violation, is not enough to spend a coin from that user's wallet.

4.2 Scaling Back the Punishment for System Violators

When tracing is deemed too harsh a punishment or simply to make the system more efficient when tracing is not needed, two other options are available:

Option (1): violation is detected and user's identity is revealed. This system operates as the above except that during the Withdraw protocol the user does not give the bank verifiable encryptions of her wallet secret s. Then later during the Deposit protocol, the bank may still detect the violation and identify the user, but will not be able to compute the serial numbers of other transactions involving this user.

Option (2): violation is detected. This system operates as the Option (1) system, except that during Spend, the user does not provide the merchant with either values T or W. Then later during the Deposit protocol, the bank may still detect a violation, but will not be able to run *Open* and identify the user.

4.3 Efficiency Considerations

We give the detailed protocols in the full version of the paper (they are rather similar to the detailed ones of the CHL scheme [11] and require slightly less than double the work of the participants). As indication of the protocols efficiency let us state some numbers here. One can construct Spend such that a user must compute fourteen multi-base exponentiations to build the commitments and twenty more for the proof. The merchant and bank need to do twenty multi-base exponentiations to check that the coin is valid. The protocols require two rounds of communication between the user and the merchant and one round between the bank and the merchant. If one takes Option (2) above, then it is thirteen multi-base exponentiations to build the commitments and eighteen more for the proof. Verification by bank and merchant takes eighteen multi-base exponentiations.

Acknowledgments

Part of Jan Camenisch's work reported in this paper is supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and by the IST Project PRIME. The PRIME projects receives research funding from the European Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Part of Susan Hohenberger's work is supported by an NDSEG Fellowship.

Anna Lysyanskaya is supported by NSF Career grant CNS 0347661.

References

1. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *NDSS*, p. 29–43, 2005.
2. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-Resistant Storage. Johns Hopkins University, CS Technical Report # TR-SP-BGMM-050705. `http://spar.isi.jhu.edu/~mgreen/correlation.pdf`, 2005.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, volume 1233, p. 480–494, 1997.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, p. 54–73, 2004.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures using strong Diffie-Hellman. In *CRYPTO*, volume 3152 of LNCS, p. 41–55, 2004.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, p. 213–229, 2001.
7. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, volume 1807 of *LNCS*, p. 431–444, 2000.
8. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, volume 1233 of *LNCS*, p. 318–333, 1997.
9. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *SIAM*, p. 457–466, 1995.
10. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In T. Okamoto, editor, *ASIACRYPT '00*, volume 1976 of *LNCS*, p. 331–345, 2000.
11. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, volume 3494 of LNCS, p. 302–321, 2005.
12. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, p. 268–289, 2002.
13. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, volume 3152 of *LNCS*, p. 56–72, 2004.
14. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, volume 1592, p. 107–122, 1999.
15. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, volume 1666 of *LNCS*, p. 413–430, 1999.

16. J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998.
17. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *EUROCRYPT '98*, volume 1403 of *LNCS*, p. 561–575, 1998.
18. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, p. 199–203. Plenum Press, 1982.
19. D. Chaum. Blind signature systems. In *CRYPTO '83*, p. 153–156. Plenum, 1983.
20. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '90*, volume 403 of *LNCS*, p. 319–327, 1990.
21. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, volume 740 of *LNCS*, p. 89–105, 1993.
22. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, p. 174–187, 1994.
23. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, 2002.
24. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, volume 2567 of *LNCS*, p. 1–17, 2003.
25. Y. Dodis and A. Yampolsky. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography*, volume 3386 of LNCS, p. 416–431, 2005.
26. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of LNCS, p. 186–194, 1986.
27. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, volume 1294 of *LNCS*, p. 16–30, 1997.
28. S. D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of LNCS, p. 495–513, 2001.
29. S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT*, volume 3027 of *LNCS*, p. 590–608, 2004.
30. D. Kügler and H. Vogt. Fair tracing without trustees. In *Financial Cryptography '01*, volume 2339 of LNCS, p. 136–148, 2001.
31. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT*, volume 3027 of LNCS, p. 571–589, 2004.
32. N. McCullagh and P. S. L. M. Barreto. A new two-party identity-based authenticated key agreement. In *CT-RSA*, volume 3376 of LNCS, p. 262–274, 2004.
33. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51, Number 2:231–262, 2004.
34. L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS 2005*, number 3531 in LNCS, p. 318–333. Springer Verlag, 2005.
35. T. Okamoto and K. Ohta. Disposable zero-knowledge authentications and their applications to untraceable elec. cash. In *CRYPTO*, volume 435, p. 481–496, 1990.
36. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '92*, volume 576 of *LNCS*, p. 129–140, 1992.
37. T. Sander and A. Ta-Shma. Flow control: a new approach for anonymity control in electronic cash systems. In *FC*, volume 1648 of *LNCS*, p. 46–61, 1999.
38. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
39. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Available at http://eprint.iacr.org/2002/164, 2002.
40. M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT '95*, volume 921 of *LNCS*, p. 209–219, 1995.
41. I. Teranishi, J. Furukawa, and K. Sako. k-times anonymous authentication (extended abstract). In *Asiacrypt 2004*, volume 3329 of *LNCS*, p. 308–322, 2004.