Compact E-Cash

Jan Camenisch[1], Susan Hohenberger[2,⋆], and Anna Lysyanskaya[3,⋆⋆]

[1] IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland.
jca@zurich.ibm.com
[2] CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA,
srhohen@mit.edu
[3] Computer Science Department, Brown University, Providence, RI 02912, USA,
anna@cs.brown.edu

Abstract. This paper presents efficient off-line anonymous e-cash schemes where a user can withdraw a wallet containing 2^ℓ coins each of which she can spend unlinkably. Our first result is a scheme, secure under the strong RSA and the y-DDHI assumptions, where the complexity of the withdrawal and spend operations is $\mathcal{O}(\ell + k)$ and the user's wallet can be stored using $\mathcal{O}(\ell + k)$ bits, where k is a security parameter. The best previously known schemes require at least one of these complexities to be $\mathcal{O}(2^\ell \cdot k)$. In fact, compared to previous e-cash schemes, our whole wallet of 2^ℓ coins has about the same size as *one* coin in these schemes. Our scheme also offers exculpability of users, that is, the bank can prove to third parties that a user has double-spent. We then extend our scheme to our second result, the first e-cash scheme that provides traceable coins without a trusted third party. That is, once a user has double spent one of the 2^ℓ coins in her wallet, *all* her spendings of these coins can be traced. However, the price for this is that the complexity of the spending and of the withdrawal protocols becomes $\mathcal{O}(\ell \cdot k)$ and $\mathcal{O}(\ell \cdot k + k^2)$ bits, respectively, and wallets take $\mathcal{O}(\ell \cdot k)$ bits of storage. All our schemes are secure in the random oracle model.

1 Introduction

Electronic cash was invented by Chaum [22, 23], and extensively studied since, e.g., [24, 26, 37, 7, 41, 20, 21]. The main idea is that, even though the same party (a bank $\mathcal{B}$) is responsible for giving out electronic coins, and for later accepting them for deposit, the withdrawal and the spending protocols are designed in such a way that it is impossible to identify when a particular coin was spent. I.e., the withdrawal protocol does not reveal any information to the bank that would later enable it to trace how a coin was spent.

As a coin is represented by data, and it is easy to duplicate data, an electronic cash scheme requires a mechanism that prevents a user from spending the same coin twice (double-spending). There are two scenarios. In the *on-line* scenario [23–25], the bank is on-line in each transaction to ensure that

⋆ Research performed while at IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland.
⋆⋆ Supported by NSF Career Grant CNS-0347661.

no coin is spent twice, and each merchant must consult the bank before accepting a payment. In the *off-line* [26] scenario, the merchant accepts a payment autonomously, and later submits the payment to the bank; the merchant is guaranteed that such a payment will be either honored by the bank, or will lead to the identification (and therefore punishment) of the double-spender.

In this paper, we give an off-line 2^ℓ-spendable unlinkable electronic cash scheme. Namely, our scheme allows a user to withdraw a wallet with 2^ℓ coins, such that the space required to store these coins, and the complexity of the withdrawal protocol, are proportional to ℓ, rather than to 2^ℓ. We achieve this without compromising the anonymity and unlinkability properties usually required of electronic cash schemes. This problem is well-motivated: (1) communication with the bank is a bottleneck in most electronic cash schemes and needs to be minimized; (2) it is desirable to store many electronic coins compactly, as one can imagine that they may be stored on a dedicated device such as a smartcard that cannot store too much data. This problem has also proved quite elusive: no one has offered a compact e-cash solution (even for a weaker security model) since the introduction of electronic cash in the 1980s.

In addition, a good e-cash scheme should allow one to expose double-spenders to outside third parties in an undeniable fashion. I.e., assuming a PKI, if a user $\mathcal{U}$ with public key $pk_{\mathcal{U}}$ spent a coin more times than he is allowed (in our case, spent $2^\ell + 1$ coins from a wallet containing 2^ℓ coins), then this fact can be proven to anyone in a sound fashion. This property of an e-cash scheme is satisfied by numerous schemes in the literature. Our solution has this property as well.

Finally, it may often be desirable that an e-cash scheme should allow one to trace *all* coins of a cheating user. It was known that this property can be implemented using a trusted third party (TTP) [40, 10], by requiring that: (1) in each withdrawal protocol a user gives to the bank an encryption under the TTP's public key of a serial number S which will be revealed during the spending protocol; and (2) in each spending protocol, the user submits to the merchant an encryption of the user's public key under the TTP's public key. Then, should a coin with serial number S ever be double-spent, the TTP can get involved and decrypt the serial number of all of this user's coins. But the existence of such a TTP contradicts the very definition of electronic cash: to the TTP, the user is not anonymous! Therefore, another desirable and elusive property of an electronic cash scheme was traceability *without* a TTP. Our scheme achieves this property as well.

Recently, Jarecki and Shmatikov [34] also made a step in this direction. Although their work is not explicitly about electronic cash, it can be thought of in this way. Their scheme allows to withdraw and *linkably* (linkability is actually a feature for them) but anonymously spend a coin K times; but should a user wish to spend the coin $K+1$ times, his identity gets revealed. As far as electronic cash is concerned, our solution is better for two reasons: (1) their scheme does not achieve unlinkability; and (2) in their protocol, each time a user spends a coin he has to run a protocol whose communication complexity is proportional to K, rather than $\log K$, as we achieve. In 1989, Okamoto and Ohta [37] proposed an e-cash scheme with similar functionality, without achieving unlinkability or compact wallets.

Our work can also be viewed as improving on the recent traceable group signatures by Kiayias, Tsiounis, and Yung [35]. In their scheme, once a special piece of tracing information is released, it is possible to trace all group signatures issued by a particular group member; otherwise this member's signatures are guaranteed to remain anonymous. Normally, in a group signature setting, this piece of information *must* be released by a TTP, as there is no equivalent of a *double-spender* whose misbehavior may automatically lead to the release of the tracing information; however, if a limit is placed on how many signatures a group member may issue, then our e-cash scheme can be viewed as a *bounded* group signature scheme, where a group member can sign a message by incorporating it into the signature proof of a coin's validity. A group manager may allocate signing rights by acting as a bank allocating coins; and if any member exceeds their allocation, the special tracing information is revealed automatically, and all signatures produced by that group member may be traced. Our tracing algorithm is more efficient than that of Kiayias et al. [35]; in our scheme, signatures can be tracked by a serial number (that appears to be random until the user double-spends), while in theirs, *all* existing signatures must be tested, one-by-one, using the special tracing information provided by the TTP, to determine if a certain signer created it or not.

Our results. Let us summarize our results. We give a compact e-cash scheme with all the features described above in the random-oracle model, under the Strong RSA assumption in combination with the decisional Diffie-Hellman inversion (y-DDHI) [4, 31] and sum-free DDH [30] assumptions for groups with bilinear maps. The communication complexity of the spending and of the withdrawal protocol is $\mathcal{O}(\ell \cdot k)$ and $\mathcal{O}(\ell \cdot k + k^2)$ bits, respectively; it takes $\mathcal{O}(\ell \cdot k)$ bits to store all the coins. This scheme is presented in Section 4.2.

We also give a scheme where the withdrawal and the spending protocols have complexity only $\mathcal{O}(\ell + k)$, and it also takes only $\mathcal{O}(\ell + k)$ bits to store all the coins, based on the Strong RSA [33, 3] and the y-DDHI [31] assumptions in the random-oracle model. This less expensive scheme does not allow traceability, however. This scheme is presented in Section 4.1.

Furthermore, in the model where the bank completely trusts the merchant (this applies to, for example, a subscription service where the entity creating and verifying the coins is one and the same), we have solutions based on the same set of assumptions but *in the standard model.* Sections 4.1 and 4.2 containing our random-oracle-based schemes also explain how these security properties are obtained once the random oracle is removed.

Overview of our construction. Our schemes are based on the signature schemes with protocols due to Camenisch and Lysyanskaya [14, 15]. These schemes allow a user to efficiently obtain a signature on committed messages from the signer. They further allow the user to convince a verifier that she possesses a signature by the signer on a committed message. Both of these protocols rely on the Pedersen commitment scheme.

To explain our result, let us describe how single-use electronic cash can be obtained with CL-signatures, drawing on a variety of previously known techniques [9, 14].

Let $G = \langle g \rangle$ be a group of prime order q where the discrete logarithm problem is hard. Suppose that a user $\mathcal{U}$ has a secret key $sk_{\mathcal{U}} \in \mathbb{Z}_q$ and a public key $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$. An electronic coin is a signature under the bank $\mathcal{B}$'s public key $pk_{\mathcal{B}}$ on the set of values $(sk_{\mathcal{U}}, s, t)$, where $s, t \in \mathbb{Z}_q$ are random values. The value s is the *serial number* of the coin, while t is the *value blinding* of this coin. A protocol whereby a user obtains such a signature is called the *withdrawal protocol*.

In the *spending protocol*, the user sends the merchant a Pedersen commitment C to the values $(sk_{\mathcal{U}}, s, t)$, and computes a non-interactive proof π_1 that they have been signed by the bank. The merchant verifies π_1 and then picks a random value $R \in \mathbb{Z}_q$. Finally, the user reveals the serial number s, and the value $T = sk_{\mathcal{U}} + R \cdot t \bmod q$. Let us refer to T as a *double-spending equation* for the coin. The user must also compute a proof π_2 that the values s and T correspond to commitment C. Finally, the merchant submits (s, R, T, π_1, π_2) for payment.

Note that one double-spending equation reveals nothing about $sk_{\mathcal{U}}$ because t is random, but using two double-spending equations, we can solve for $sk_{\mathcal{U}}$. So if the same serial number s is submitted for payment twice, the secret key $sk_{\mathcal{U}}$ and therefore the identity of the double-spender $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$ can be discovered.

Now, our goal is to adapt single-use electronic cash schemes so that a coin can be used at most 2^ℓ times. The trivial solution would be to obtain 2^ℓ coins. For our purposes, however, it is unacceptable, as 2^ℓ may be quite large (e.g., 1000) and we want each protocol to be efficient.

The idea underlying our system is that the values s and t implicitly define several (pseudorandom) serial numbers S_i and blinding values B_i, respectively. In other words, we need a pseudorandom function $F_{(\cdot)}$ such that we can set $S_i = F_s(i)$, and $B_i = F_t(i)$, $0 \le i \le 2^\ell - 1$. Then the user gets 2^ℓ pseudorandom serial numbers with the corresponding double-spending equations defined by (s, t). Here, the double-spending equation for coin i is $T_i = g^{sk_{\mathcal{U}}}(B_i)^R$, where R is chosen by the merchant. This leaves us with a very specific technical problem. The challenge is to find a pseudorandom function such that, given (1) a commitment to $(sk_{\mathcal{U}}, s, t)$; (2) a commitment to i; and (3) the values S_i and T_i, the user can efficiently prove that she derived the values S_i and T_i correctly from $sk_{\mathcal{U}}$, s, and t, i.e., $S_i = F_s(i)$ and $T_i = g^{sk_{\mathcal{U}}}(F_t(i))^{R_i}$ for some $0 \le i \le 2^\ell - 1$ and public value R_i provided by the merchant.

Recently, Dodis and Yampolsky [31] proposed the following discrete-logarithm-based pseudorandom function (PRF): $F_s(x) = g^{1/(s+x+1)}$, where $s, x \in \mathbb{Z}_q$, and g is a generator of a group G of order q in which the decisional Diffie-Hellman inversion problem is hard.[4] (In the sequel, we denote this PRF as $F^{DY}_{(\cdot)}(\cdot)$.) Using

[4] Another PRF suitable for our purposes is the one due to Naor and Reingold [36]. It is based on the DDH problem, but makes our subsequent schemes less time and space efficient.

standard methods for proving statements about discrete-logarithm representations, we obtain a zero-knowledge argument system for showing that a pair of values (S_i, T_i) is of the form $S_i = F_s^{DY}(i)$ and $T_i = g^{sk_{\mathcal{U}}}(F_t^{DY}(i))^{R_i}$ corresponding to the seeds s and t signed by bank $\mathcal{B}$ and to some index $i \in [0, 2^\ell - 1]$.

Note that if S_i and T_i are computed this way, then they are elements of G rather than of $\mathbb{Z}_q$. So this leaves us with the following protocol: to withdraw a coin, a user obtains a signature on $(sk_{\mathcal{U}}, s, t)$. During the spending protocol, the user reveals S_i and the double-spending equation $T_i = g^{sk_{\mathcal{U}}}(B_i)^{R_i}$, where $sk_{\mathcal{U}}$ is the user's secret key and $pk_{\mathcal{U}} = g^{sk_{\mathcal{U}}}$ the corresponding public key. Now, with two double-spending equations $T_1 = g^{sk_{\mathcal{U}}} B_i^{R_1}$ and $T_2 = g^{sk_{\mathcal{U}}} B_i^{R_2}$ we can infer the value $(T_1^{R_2}/T_2^{R_1})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2} B_i^{R_1 R_2}/pk_{\mathcal{U}}^{R_1} B_i^{R_1 R_2})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2-R_1})^{(R_2-R_1)^{-1}} = pk_{\mathcal{U}}$. This is sufficient to detect and identify double spenders. We describe this construction in more depth in Section 4.1.

However, the above scheme does not allow the bank to identify the other spendings of the coin, i.e., to generate all the serial numbers that the user can derive from s. Let us now describe how we achieve this. For the moment, let us assume that the technique described above allows us to infer $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$. If this were the case, we could require that the user, as part of the withdrawal protocol, should verifiably encrypt [1, 11, 18] the value s *under her own* $pk_{\mathcal{U}}$, to form a ciphertext c. The record $(pk_{\mathcal{U}}, c)$ is stored by the bank. Now, suppose that at a future point, the user spends too many coins and thus her $sk_{\mathcal{U}}$ is discovered. From this, her $pk_{\mathcal{U}}$ can be inferred and the record $(pk_{\mathcal{U}}, c)$ can be located. Now that $sk_{\mathcal{U}}$ is known, c can be decrypted, the seed s discovered, the values S_i computed for all $0 \le i < 2^\ell$, and hence the database of transactions can be searched for records with these serial numbers.

Let us now redefine the way a user's keys are picked such that we can recover $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$. Suppose that G is a group with a non-degenerate bilinear map $e : G \times G \mapsto G'$. Let $sk_{\mathcal{U}}$ be an element of $\mathbb{Z}_q$. Let $pk_{\mathcal{U}} = e(g, g^{sk_{\mathcal{U}}})$. Recently, Ateniese, Fu, Green, and Hohenberger [2] exhibited a cryptosystem that uses $pk_{\mathcal{U}}$ as a public key, such that in order to decrypt it is sufficient to know the value $g^{sk_{\mathcal{U}}}$. So, in our scheme, the user $\mathcal{U}$ would encrypt s under $pk_{\mathcal{U}}$ using the cryptosystem due to Ateniese et al. From the double-spending equations, the same way as before, the bank infers the value $g^{sk_{\mathcal{U}}}$. This value now allows the bank to decrypt s.

This is almost the solution, except for the following subtlety: if G has a bilinear map, then the decisional Diffie-Hellman problem is easy, and so the Dodis-Yampolsky construction is not a PRF in this setting! Instead, we must assume sum-free decisional Diffie-Hellman [30], and slightly change the construction. This is why the variant of our scheme that allows to trace coins is a factor of ℓ more expensive than the one that does not. The details of this construction are given in Section 4.2.

One of the main remaining problems for electronic cash which this paper does not address is that of efficiently allowing for multiple denominations in a non-trivial way; i.e., without executing the spending protocol a number of times.

2 Definition of Security

Notation: if P is a protocol between A and B, then $P(A(x), B(y))$ denotes that A's input is x and B's is y.

Our electronic 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, $\mathsf{Deposit}$, $\mathsf{Identify}$, $\mathsf{VerifyGuilt}$, Trace, $\mathsf{VerifyOwnership}$. Let us give some input-output specifications for these protocols, as well as some informal intuition for what they do.

- The $\mathsf{BKeygen}(1^k, \mathit{params})$ algorithm is a key generation algorithm for the bank $\mathcal{B}$. It takes as input the security parameter 1^k and, if the scheme is in the common parameters model, it also takes as input these parameters params. This algorithm outputs the key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$. (Assume that $sk_{\mathcal{B}}$ contains the params, so we do not have to give params explicitly to the bank again.)
  Similarly, $\mathsf{UKeygen}(1^k, \mathit{params})$ is a key generation algorithm for the user $\mathcal{U}$, which outputs $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$. Since merchants are a subset of users, they may use this algorithm to obtain keys as well. (Assume that sk contains the params, so we do not have to give params explicitly to the user again.)
- In the $\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, n), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, n))$ protocol, the user $\mathcal{U}$ withdraws a *wallet* W of n coins from the bank $\mathcal{B}$. The user's output is the wallet W, or an error message. $\mathcal{B}$'s output is some information T_W which will allow the bank to trace the user should this user double-spend some coin, or an error message. The bank maintains a database D for this trace information, to which it enters the record $(pk_{\mathcal{U}}, T_W)$.
- In a $\mathsf{Spend}(\mathcal{U}(W, pk_{\mathcal{M}}), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, n))$ protocol, a user $\mathcal{U}$ gives one of the coins from his wallet W to the merchant $\mathcal{M}$. Here, the merchant obtains a serial number S of the coin, and a proof π of validity of the coin. The user's output is an updated wallet W'.
- In a $\mathsf{Deposit}(\mathcal{M}(sk_{\mathcal{M}}, S, \pi, pk_{\mathcal{B}}), \mathcal{B}(pk_{\mathcal{M}}, sk_{\mathcal{B}}))$ protocol, a merchant $\mathcal{M}$ deposits a coin (S, π) into its account held by the bank $\mathcal{B}$. Whenever an honest $\mathcal{M}$ obtained (S, π) by running the Spend protocol with any (honest or otherwise) user, there is a guarantee that this coin will be accepted by the bank. $\mathcal{B}$ adds (S, π) to to its list L of spent coins. The merchant's output is nothing or an error message.
- The $\mathsf{Identify}(\mathit{params}, S, \pi_1, \pi_2)$ algorithm allows to identify double-spenders using a serial number S and two proofs of validity of this coin, π_1 and π_2, possibly submitted by malicious merchants. This algorithm outputs a public key pk_U and a proof Π_G. If the merchants who had submitted π_1 and π_2 are *not* malicious, then Π_G is evidence that $pk_{\mathcal{U}}$ is the registered public key of a user that double-spent coin S.
- The $\mathsf{VerifyGuilt}(\mathit{params}, S, pk_{\mathcal{U}}, \Pi_G)$ algorithm allows to publicly verify proof Π_G that the user with public key $pk_{\mathcal{U}}$ is guilty of double-spending coin S.
- The $\mathsf{Trace}(\mathit{params}, S, pk_{\mathcal{U}}, \Pi_G, D, n)$ algorithm, given a public key $pk_{\mathcal{U}}$ of a double-spender, a proof Π_G of his guilt in double-spending coin S, the

database D, and a wallet size n, computes the serial numbers $S_1, \ldots, S_m$ of all of the coins issued to $\mathcal{U}$ along with proofs $\Pi_1, \ldots, \Pi_m$ of $pk_{\mathcal{U}}$'s ownership. If $\mathsf{VerifyGuilt}(params, S,\ pk_{\mathcal{U}},\ \Pi_G)$ does not accept (i.e., $pk_{\mathcal{U}}$ is honest), this algorithm does nothing.
- The $\mathsf{VerifyOwnership}(params, S, \Pi, pk_{\mathcal{U}}, n)$ algorithm allows to publicly verify the proof Π that a coin with serial number S belongs to a double-spender with public key $pk_{\mathcal{U}}$.

We will now informally define the security properties. The more elaborate formal definitions are given in the full version of this paper [13].

Correctness. If an honest user runs Withdraw with an honest bank, then neither will output an error message; if an honest user runs Spend with an honest merchant, then the merchant accepts the coin.

Balance. From the bank's point of view, what matters is that no collection of users and merchants can ever spend more coins than they withdrew. We require that there is a knowledge extractor $\mathcal{E}$ that executes u Withdraw protocols with all adversarial users and extracts un serial numbers $S_1, \ldots, S_{un}$. We require that for every adversary, the probability that an honest bank will accept (S, π) as the result of the Deposit protocol, where $S \neq S_i\ \forall 1 \leq i \leq un$, is negligible. If $S_1, \ldots, S_n$ is a set of serial numbers output by $\mathcal{E}$ when running Withdraw with public key $pk_{\mathcal{U}}$, we say that coins $S_1, \ldots, S_n$ *belong* to the user $\mathcal{U}$ with $pk_{\mathcal{U}}$.

Identification of double-spenders. Suppose $\mathcal{B}$ is honest. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are honest merchants who ran the Spend protocol with the adversary, such that $\mathcal{M}_1$'s output is (S, π_1) and $\mathcal{M}_2$'s output is (S, π_2). This property guarantees that, with high probability, $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ outputs a key $pk_{\mathcal{U}}$ and proof Π_G such that $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$ accepts.

Tracing of double-spenders. Given that a user $\mathcal{U}$ is shown guilty of double-spending coin S by a proof Π_G such that VerifyGuilt accepts, this property guarantees that $\mathsf{Trace}(params, S, pk_{\mathcal{U}}, \Pi_G, D, n)$ will output the serial numbers $S_1, \ldots, S_m$ of all coins that belong to $\mathcal{U}$ along with proofs of ownership $\Pi_1, \ldots, \Pi_m$ such that for all i, with high probability, $\mathsf{VerifyOwnership}(params, S_i, \Pi_i, pk_{\mathcal{U}}, n)$ also accepts.

Anonymity of users. From the privacy point of view, what matters to users is that the bank, even when cooperating with any collection of malicious users and merchants, cannot learn anything about a user's spendings other than what is available from side information from the environment. In order to capture this property more formally, we introduce a simulator $\mathcal{S}$. $\mathcal{S}$ has some side information not normally available to players. E.g., if in the common parameters model, $\mathcal{S}$ generated these parameters; in the random-oracle model, $\mathcal{S}$ is in control of the random oracle; in the public-key registration model $\mathcal{S}$ may hold additional information about the bank's keys, etc. We require that $\mathcal{S}$ can create simulated coins without access to any wallets, such that a simulated coin is indistinguishable from a valid one. More precisely, $\mathcal{S}$ executes the user's side of the Spend protocol without access to the user's secret or public key, or his wallet W.

Exculpability. Suppose that we have an adversary that participates any number of times in the $\mathsf{Withdraw}$ protocol with the honest user with public key $pk_{\mathcal{U}}$, and subsequently to that, in any number of legal Spend protocols with the same user. I.e., if the user withdrew u wallets of n coins each, then this user can participate in at most un Spend protocols. The adversary then outputs a coin serial number S and a purported proof Π that the user with public key $pk_{\mathcal{U}}$ is a double-spender and owns coin S. The *weak* exculpability property postulates that, for all adversaries, the probability $\mathsf{VerifyOwnership}(params,\ S,\ pk_{\mathcal{U}}, \Pi, n)$ accepts is negligible.

Furthermore, the adversary may continue to engage the user $\mathcal{U}$ in Spend protocols even if it means $\mathcal{U}$ must double-spend some coins of her choosing (in which case the state of her wallet is reset). The adversary then outputs (S, Π). The *strong* exculpability property postulates that, for all adversaries, when S is a coin serial number *not* belonging to $\mathcal{U}$, the weak exculpability property holds, and when S is a coin serial number *not* double-spent by user $\mathcal{U}$ with public key $pk_{\mathcal{U}}$, the probability that $\mathsf{VerifyGuilt}(params, S, \Pi, pk_{\mathcal{U}}, n)$ accepts is negligible.

This ends the informal description of our security definition; the descriptions in the full version of this paper [13] are more precise, but this intuition should be sufficient for understanding our subsequent security guarantees.

Strengthening the definition: the UC framework. Even though our definition of security is not in the UC framework, note that our definition would imply UC-security whenever the extractor $\mathcal{E}$ and simulator $\mathcal{S}$ are constructed appropriately. In a nutshell, an ideal electronic cash functionality would allow an honest user to withdraw and spend n coins. In this case, if the merchant and bank are controlled by the malicious environment, the simulator $\mathcal{S}$ defined above creates the merchant's and bank's view of the Spend protocol. At the same time, the balance property guarantees that the bank gets the same protection in the real world as it does in the ideal world, and the exculpability property ensures that an honest user cannot get framed in the real world, just as he cannot get framed in the ideal world.

3 Preliminaries

Our e-cash systems 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. The notation $G = \langle g \rangle$ means that g generates the group G.

3.1 Complexity Assumptions

The security of our e-cash systems is based on the following assumptions:

Strong RSA Assumption [3, 33]: 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, 31]: 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.[5]

Sum-Free Decisional Diffie-Hellman Assumption (SF-DDH) [30]: 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 $\mathcal{A}^{(\cdot)}$,

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

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

3.2 Bilinear Maps

Let $\mathsf{Bilinear\_Setup}$ be an algorithm that, on input the security parameter 1^k, outputs $\gamma = (q, g_1, h_1, G_1, g_2, h_2, G_2, e)$, where e is a non-degenerate efficiently computable bilinear map from $G_1 = \langle g_1 \rangle = \langle h_1 \rangle$ to $G_2 = \langle g_2 \rangle = \langle h_2 \rangle$, both groups of prime order $q = \Theta(2^k)$. Let $e(g_1, g_1) = g_2$ and $e(h_1, h_1) = h_2$. We assume that each group element has a unique binary representation. More formally, $e : G_1 \times G_1 \to G_2$ is a function that is: (bilinear) for all $g_1, h_1 \in G_1$, for all $a, b \in \mathbb{Z}_q$, $e(g_1^a, h_1^b) = e(g_1, h_1)^{ab}$; (non-degenerate) if g_1 is a generator of G_1, then $e(g_1, g_1)$ generates G_2; and (efficient) computing $e(\cdot, \cdot)$ is efficient for all $g_1, h_1 \in G_1$.

3.3 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 [39] or a composite [33, 29], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [27] or composite [17] moduli, (3) proof that a commitment opens to the product of two other committed values [16, 20, 8], (4) proof that a committed value lies in

[5] Others [4, 31] have used a stronger *bilinear* version of the y-DDHI assumption, where, given the same input in $\langle g \rangle^{y+1}$ it is hard to distinguish $e(g, g)^{1/x}$ from a random R in $\langle e(g, g) \rangle$.

a given integer interval [21, 16, 16, 6], and also (5) proof of the disjunction or conjunction of any two of the previous [28]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption.

When refering to the proofs above, we will follow the notation introduced by Camenisch and Stadler [19] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta \wedge (u \leq \alpha \leq v)\}$$

denotes a "*zero-knowledge Proof of Knowledge of integers α, β, and δ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta$ holds, where $u \leq \alpha \leq v$,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities of which knowledge is being proven, while all other values are known to the verifier. We apply the Fiat-Shamir heuristic [32] to turn such proofs of knowledge into signatures on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^\alpha\}(m)$.

3.4 Pseudorandom Functions

A useful building block of our e-cash systems is the pseudorandom functions recently proposed by Dodis and Yampolsky [31], which they expand to verifiable random functions using bilinear maps. Their construction is:

For every n, a function $f \in F_n$ is defined by the tuple (G, q, g, s), where G is group of order q, q is an n-bit prime, g is a generator of G, and s is a seed in $\mathbb{Z}_q$. For any input $x \in \mathbb{Z}_q$ (except for $x = -1 \bmod q$), the function $f_{P,q,g,s}(\cdot)$, which we simply denote as $f_s^{DY}(\cdot)$, is defined as $f_s^{DY}(x) = g^{1/(x+s+1)}$.

This construction is secure under the y-DDHI assumption. As mentioned in the introduction, we could instead substitute in the Naor-Reingold PRF [36], and replace the y-DDHI assumption with the more standard DDH assumption, at the cost of enlarging our wallets from $\mathcal{O}(\ell + k)$ bits to $\mathcal{O}(\ell \cdot k)$ bits.

3.5 CL Signatures

Recall the Pedersen commitment scheme [38], in which the public parameters are a group G 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 $C = \mathrm{PedCom}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$.

Camenisch and Lysyanskaya [14] came up with a secure signature scheme 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 $\mathcal{O}(\ell \log q)$. (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 the purposes of this exposition, it does not matter *how* CL signatures actually work, all that matters are the facts stated above.

Our subsequent e-cash systems will require the strong RSA assumption independently of the CL signatures. By making additional assumptions based on bilinear maps, we can use alternative schemes by Camenisch and Lysyanskaya [15] and Boneh, Boyen and Shacham [5], yielding shorter signatures in practice.

3.6 Verifiable, Bilinear El Gamal Encryption

In Section 4.2, we apply a technique by Camenisch and Damgård [11] 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 [11] for the full discussion.

In particular, we apply the verifiable encryption techniques above to a bilinear variant of El Gamal encryption due to Ateniese, Fu, Green, and Hohenberger [2]. Assume we run $\mathsf{Bilinear\_Setup}$ on 1^k to obtain $\zeta = (q, \mathrm{g}_3, \mathcal{G}_3, \mathrm{g}_2, \mathcal{G}_2, e)$, where we have bilinear map $e : \mathcal{G}_3 \times \mathcal{G}_3 \to \mathcal{G}_2$. Let (G, E, D) denote the standard key generation, encryption, and decryption algorithms. On input $(1^k, \zeta)$, the key generation algorithm G outputs a key pair $(pk, sk) = (e(\mathrm{g}_3, \mathrm{g}_3^u), \mathrm{g}_3^u) = (\mathrm{g}_2^u, \mathrm{g}_3^u)$ for a random $u \in \mathbb{Z}_q$. To encrypt a message $m \in \mathcal{G}_2$ under pk, select a random $k \in \mathbb{Z}_q$ and output the ciphertext $c = (\mathrm{g}_3^k, pk^k m) = (\mathrm{g}_3^k, \mathrm{g}_2^{uk} m)$. Then, to decrypt $c = (c_1, c_2)$, simply compute $c_2 / e(c_1, sk)$. Ateniese et al. [2] show that this encryption scheme is semantically-secure under the decisional bilinear DH assumption.

4 Two Compact E-Cash Systems

We present two compact e-cash systems. In System One, an honest bank can quickly detect double-spending, identify the perpetrator, and prove his guilt to a third party from two coin deposits with the same serial number. This system allows a wallet of 2^ℓ coins to be stored in $\mathcal{O}(\ell + k)$ bits. In System Two, the bank can do everything that it could before, and in addition, the bank can compute the serial numbers for all the coins that belong to the perpetrator along with proofs of their ownership. Here, our wallets of 2^ℓ coins require $\mathcal{O}(\ell \cdot k)$ bits, which is still remarkably small. If a user does not double-spend, her coins are unlinkable. There is no trusted party.

Global parameters for both systems. Let 1^k be the security parameter and let ℓ be any value in $\mathcal{O}(\log k)$. Our subsequent schemes work most efficiently by having three different algebraic groups:

- $\mathcal{G}_1 = \langle g_1 \rangle$, where n is a special RSA modulus of $2k$ bits, g_1 is a quadratic residue modulo n, and $h_1 \in \mathcal{G}_1$.
- $\mathcal{G}_2 = \langle \mathsf{g}_2 \rangle$, where g_2 is an element of prime order $q = \Theta(2^k)$, and $\mathsf{h}_2 \in \mathcal{G}_2$.
- $\mathcal{G}_3 = \langle \mathsf{g}_3 \rangle$, where g_3 is an element of the same prime order as $\mathcal{G}_2$, and there exists a bilinear mapping $e : \mathcal{G}_3 \times \mathcal{G}_3 \to \mathcal{G}_2$.

Our first scheme will not require $\mathcal{G}_3$. Assume that, on input 1^k, each system is initialized with the necessary common parameters, denoted ζ. We also define the multiple Pedersen commitment as $\mathrm{PedCom}(x_1, \ldots, x_n; r) = h^r \Pi_{i=1}^n g_i^{x_i}$. Sometimes for simplicity we do not explicitly include the randomess r in the input to the commitment. The values $h, \{g_i\}$ are assumed to be publicly known elements of the appropriate group.

4.1 System One: Wallets of Size $\mathcal{O}(\ell + k)$ with Public Key Recovery

Our first system supports the basic algorithms (BKeygen, UKeygen, Withdraw, Spend, Deposit, Identify, VerifyGuilt). In this scheme, a wallet of size $\mathcal{O}(\ell + k)$ is sufficient to hold 2^ℓ coins. In the Identify algorithm, the bank can recover the identity of a double-spender $pk_{\mathcal{U}}$ from two deposits with the same coin serial number S. Using VerifyGuilt, the bank can prove that $pk_{\mathcal{U}}$ double-spent coin S to a third party; while all honest users are guaranteed strong exculpability.

The parties set up their keys as follows. In BKeygen$(1^k, \zeta)$, the bank $\mathcal{B}$ generates a CL signature key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ for message space M such that $\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \subseteq M$. In UKeygen$(1^k, \zeta)$, each user $\mathcal{U}$ generates a unique key pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = (\mathsf{g}_2^u, u)$ for a random $u \in \mathbb{Z}_q$. Recall that merchants are a subset of users.

Withdraw$(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, 2^\ell), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, 2^\ell))$**:** A user $\mathcal{U}$ interacts with the bank $\mathcal{B}$ as follows:

1. $\mathcal{U}$ identifies himself to the bank $\mathcal{B}$ by proving knowledge of $sk_{\mathcal{U}}$.
2. In this step, the user and bank contribute randomness to the wallet secret s; the user also selects a wallet secret t. This is done as follows: $\mathcal{U}$ selects random values $s', t \in \mathbb{Z}_q$ and sends a commitment $A' = \mathrm{PedCom}(u, s', t; r)$ to $\mathcal{B}$. $\mathcal{B}$ sends a random $r' \in \mathbb{Z}_q$. Then $\mathcal{U}$ sets $s = s' + r'$. $\mathcal{U}$ and $\mathcal{B}$ locally compute $A = g_2^{r'} A' = \mathrm{PedCom}(u, s' + r', t; r) = \mathrm{PedCom}(u, s, t; r)$.
3. $\mathcal{U}$ and $\mathcal{B}$ run the CL protocol for obtaining $\mathcal{B}$'s signature on committed values contained in commitment A. As a result, $\mathcal{U}$ obtains $\sigma_{\mathcal{B}}(u, s, t)$.
4. $\mathcal{U}$ saves the wallet $W = (sk_{\mathcal{U}}, s, t, \sigma_{\mathcal{B}}(u, s, t), J)$, where s, t are the wallet secrets, $\sigma_{\mathcal{B}}(u, s, t)$ is the bank's signature, and J is an ℓ-bit coin counter initialized to zero.
5. $\mathcal{B}$ records a debit of 2^ℓ coins for account $pk_{\mathcal{U}}$.

$\mathsf{Spend}(\mathcal{U}(W, pk_{\mathcal{M}}), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, 2^\ell))$: $\mathcal{U}$ anonymously transfers a coin to $\mathcal{M}$ as follows. (An optimized version appears in the full version of this paper [13].)

1. $\mathcal{M}$ (optionally) sends a string $\mathit{info} \in \{0,1\}^*$ containing transaction information to $\mathcal{U}$ and authenticates himself by proving knowledge of $sk_{\mathcal{M}}$.
2. $\mathcal{M}$ chooses a random $R \in \mathbb{Z}_q^*$ and sends R to $\mathcal{U}$. This is for the double-spending equation (see Section 1).
3. $\mathcal{U}$ sends to $\mathcal{M}$ the serial number of the coin $S = F_s^{DY}(J)$, and security tag $T = pk_{\mathcal{U}} F_t^{DY}(J)^R$. Now $\mathcal{U}$ must prove their validity, i.e., that S and T correspond to wallet secrets (u, s, t) signed by $\mathcal{B}$. This is done as follows:
   (a) Let $A = \mathrm{PedCom}(J)$; prove that A is a commitment to an integer in the range $[0 \ldots 2^\ell - 1]$.
   (b) Let $B = \mathrm{PedCom}(u)$, $C = \mathrm{PedCom}(s)$, $D = \mathrm{PedCom}(t)$; prove knowledge of a CL signature from $\mathcal{B}$ on the openings of B, C and D in that order,
   (c) Prove $S = F_s^{DY}(J) = \mathsf{g}_2^{1/(J+s+1)}$ and $T = pk_{\mathcal{U}} F_t^{DY}(J)^R = \mathsf{g}_2^{u+R/(J+t+1)}$. More formally, this proof is the following proof of knowledge:

$$PK\{(\alpha, \beta, \delta, \gamma_1, \ldots, \gamma_3) : g_1 = (AC)^\alpha {h_1}^{\gamma_1} \wedge S = \mathsf{g}_2^\alpha \wedge \\ g_1 = (AD)^\beta {h_1}^{\gamma_2} \ \wedge \ B = {g_1}^\delta {h_1}^{\gamma_3} \ \wedge \ T = \mathsf{g}_2^\delta (\mathsf{g}_2^R)^\beta\}$$

   Use the Fiat-Shamir heuristic to turn all the proofs above into one signature of knowledge on the values $(S, T, A, B, C, D, g_1, h_1, n, \mathsf{g}_2, pk_{\mathcal{M}}, R, \mathit{info})$. Call the resulting signature Φ.
4. If Φ verifies, $\mathcal{M}$ accepts the coin (S, π), where $\pi = (R, T, \Phi)$, and uses this information at deposit time.
5. $\mathcal{U}$ updates his counter $J = J + 1$. When $J > 2^\ell - 1$, the wallet is empty.

$\mathsf{Deposit}(\mathcal{M}(sk_{\mathcal{M}}, S, \pi, pk_{\mathcal{B}}), \mathcal{B}(pk_{\mathcal{M}}, sk_{\mathcal{B}}))$: A merchant $\mathcal{M}$ sends to bank $\mathcal{B}$ a coin $(S, \pi = (R, T, \Phi))$. If Φ verifies and R is fresh (i.e., the pair (S, R) is not already in the list L of spent coins), then $\mathcal{B}$ accepts the coin for deposit, adds (S, π) to the list L of spent coins, and credits $pk_{\mathcal{M}}$'s account; otherwise, $\mathcal{B}$ sends $\mathcal{M}$ an error message.

Note that in this deposit protocol, $\mathcal{M}$ must convince $\mathcal{B}$ that it behaved honestly in accepting some coin (S, π). As a result, our construction requires the Fiat-Shamir heuristic for turning a proof of knowledge into a signature. If $\mathcal{M}$ and $\mathcal{B}$ were the same entity, and the Withdraw and Spend protocols were interactive, then the bank $\mathcal{B}$ would not need to verify the validity of the coin that the merchant wishes to deposit, and as a result, we could dispense with the Fiat-Shamir heuristic and thus achieve balance and anonymity in the plain model (i.e., not just in the random oracle model).

$\mathsf{Identify}(\zeta, S, \pi_1, \pi_2)$: Suppose $(R_1, T_1) \in \pi_1$ and $(R_2, T_2) \in \pi_2$ are two entries in the bank's database L of spent coins for serial number S. Then output $\Pi_G = (\pi_1, \pi_2)$ and $pk = \left(T_2^{R_1} / T_1^{R_2}\right)^{(R_1 - R_2)^{-1}}$.

Let us explain why this produces the public key $pk_{\mathcal{U}}$ of the double-spender. Suppose coin S belonged to some user with $pk_{\mathcal{U}} = \mathrm{g}_2^u$, then each T_i is of the form $\mathrm{g}_2^{u+R_i\alpha}$ for the same values u and α. (Either this is true or an adversary has been successful in forging a coin, which we subsequently show happens with only negligible probability.) As the bank only accepts coins with fresh values of R (i.e., $R_1 \neq R_2$), it allows to compute:

$$(\frac{T_2^{R_1}}{T_1^{R_2}})^{(R_1-R_2)^{-1}} = (\frac{\mathrm{g}_2^{uR_1+R_1R_2\alpha}}{\mathrm{g}_2^{uR_2+R_1R_2\alpha}})^{(R_1-R_2)^{-1}} = \mathrm{g}_2^{\frac{u(R_1-R_2)}{(R_1-R_2)}} = \mathrm{g}_2^u = pk_{\mathcal{U}}.$$

$\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$: Parse Π_G as (π_1, π_2) and each π_i as (R_i, T_i, Φ_i). Run $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ and compare the first part of its output to the public key $pk_{\mathcal{U}}$ given as input. Check that the values match. Next, verify each Φ_i with respect to (S, R_i, T_i). If all checks pass, accept; otherwise, reject.

Efficiency Discussion of System One. The dominant computational cost in these protocols are the single and multi base exponentations. In a good implementation, a multi-base exponentation is essentially as fast as an ordinary exponentation. While we do not provide the full details of the Withdraw protocol, it can easily be derived from the known protocols to obtain a CL-signature on a committed signature [14, 12]. Depending on how the proof of knowledge protocol is implemented, Withdraw requires only three moves of communication.

The details of (an optimized version of) the Spend protocol are given in the full version [13]. One can verify that a user must compute seven multi-base exponentiations to build the commitments and eleven more for the proof. The merchant and bank need to do eleven 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.

Theorem 1. *System One supports the algorithms* (BKeygen*,* UKeygen*,* Withdraw*,* Spend*,* Deposit*,* Identify) *and guarantees balance, identification of double-spenders, anonymity of users, and strong exculpability under the Strong RSA and y-DDHI assumptions in the random oracle model.*

Proof of Theorem 1 appears in the full version of this paper [13].

4.2 System Two: Wallets of Size $\mathcal{O}(\ell \cdot k)$ with Traceable Coins

We now extend System One to allow coin tracing. Suppose for the moment that the Identify algorithm recovered $sk_{\mathcal{U}}$ rather than $pk_{\mathcal{U}}$ for a double-spender. We change the withdrawal protocol so that the user $\mathcal{U}$ must also provide the bank $\mathcal{B}$ with a verifiable encryption of her wallet secret s (used to generate the coin serial numbers) under *their own public key* $pk_{\mathcal{U}}$. This way, if $\mathcal{U}$ double-spends, $\mathcal{B}$ can compute $sk_{\mathcal{U}}$, recover her secret s, and output the serial numbers $S_i = f_s^{DY}(i)$, for $i = 0$ to $2^\ell - 1$, of all coins belonging to $\mathcal{U}$. (Observe that recovering $sk_{\mathcal{U}}$ in the above scheme allows the bank to trace *all* coins from *all* wallets for $\mathcal{U}$ and not just from the wallet from which the coin was double-spent!) If $\mathcal{U}$ does not double-spend a coin, however, her anonymity is computationally guaranteed.

The verifiable encryption techniques of Camenisch and Damgård, as described in Section 3.6, can be applied to any semantically-secure encryption scheme for our withdrawal protocol. Thus, all we need is a coin construction which allows one to recover a double-spender's $sk_{\mathcal{U}}$. Luckily, the bilinear El Gamal scheme recently proposed by Ateniese et al. (see Section 3.6) allows for public keys of the form $pk_{\mathcal{U}} = e(\mathrm{g}_3, \mathrm{g}_3^u) = \mathrm{g}_2^u$, for $u \in \mathbb{Z}_q$, where knowing $sk_{\mathcal{U}} = \mathrm{g}_3^u$ is sufficient for decryption, given the mapping $e : \mathcal{G}_3 \times \mathcal{G}_3 \rightarrow \mathcal{G}_2$. By setting our tag $T_i = \mathrm{g}_3^u (f_t^{DY}(i))^R$ in $\mathcal{G}_3$ (where R is a random value chosen by the merchant), we can now recover $sk_{\mathcal{U}} = \mathrm{g}_3^u$ from a coin spent twice.

One complication with setting T in $\mathcal{G}_3$ is that the Dodis-Yampolsky [31] construction is no longer a PRF when DDH is easy, as it is in $\mathcal{G}_3$. This breaks the anonymity of our coins. Thus, we need a new PRF.

PRF based on the Sum-Free DDH Assumption. Dodis [30] gave a definition of a *sum-free* encoding, and proved that if V is any sum-free encoding, and $\langle \mathrm{g}_3 \rangle$ is a group of order q, then $f_{(\cdot)}^V$, defined as follows, is a PRF: the seed for this PRF consists of values $t_i \in \mathbb{Z}_q$, for $0 \leq i \leq 3\ell$; let $\boldsymbol{t} = (t_0, \ldots, t_{3\ell})$; the function $f_{\boldsymbol{t}}^V$ is defined as $f_{\boldsymbol{t}}^V(x) = \mathrm{g}_3^{t_0 \prod_{V(x)_i = 1} t_i}$. This holds under the sum-free DDH assumption (also introduced by Dodis [30]); note that it seems reasonable to make such an assumption even of groups where DDH is easy.

For our purposes, we need the encoding V to have nice algebraic properties. We define an encoding $V : \{0,1\}^\ell \mapsto \{0,1\}^{3\ell}$ as $V(x) = x \circ x^2$, where $\circ$ denotes concatenation, and multiplication is over the integers. In the full version of this paper, we recall the sum-free definition and prove that this encoding is sum-free.

Our second system supports all algorithms mentioned in Section 2: ($\mathsf{BKeygen}$, $\mathsf{UKeygen}$, $\mathsf{Withdraw}$, Spend, $\mathsf{Deposit}$, $\mathsf{Identify}$, $\mathsf{VerifyGuilt}$, Trace, $\mathsf{VerifyOwnership}$). We assume a standard signature scheme $(SG, Sign, SVf)$. Then, $\mathsf{UKeygen}(1^k, \zeta)$ runs $SG(1^k, \zeta) \rightarrow (vk_{\mathcal{U}}, ssk_{\mathcal{U}})$ and the bilinear El Gamal key generation algorithm $G(1^k, \zeta) \rightarrow (ek_{\mathcal{U}}, dk_{\mathcal{U}}) = (e(\mathrm{g}_3, \mathrm{g}_3^u), \mathrm{g}_3^u) = (\mathrm{g}_2^u, \mathrm{g}_3^u)$, and outputs $pk_{\mathcal{U}} = (ek_{\mathcal{U}}, vk_{\mathcal{U}})$ and $sk_{\mathcal{U}} = (dk_{\mathcal{U}}, ssk_{\mathcal{U}})$. The bank's keys are as before.

$\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_{\mathcal{U}}, 2^\ell), \mathcal{B}(pk_{\mathcal{U}}, sk_{\mathcal{B}}, 2^\ell))$**:** A user $\mathcal{U}$ interacts with the bank $\mathcal{B}$ as follows:

1. $\mathcal{U}$ identifies himself to the bank $\mathcal{B}$ by proving knowledge of $sk_{\mathcal{U}} = (dk_{\mathcal{U}}, ssk_{\mathcal{U}})$.
2. As in System One, in this step $\mathcal{U}$ and $\mathcal{B}$ contribute randomness to the wallet secret s, and the user selects wallet secrets $\boldsymbol{t} = (t_0, \ldots, t_{3\ell})$, where $t_i \in \mathbb{Z}_q$ for all i. As before, this is done as follows: $\mathcal{U}$ chooses random values s' and $\boldsymbol{t}$, and sends the commitment $A' = \mathrm{PedCom}(u, s', \boldsymbol{t})$ to $\mathcal{B}$, obtains a random r', sets $s = s' + r'$, and then both $\mathcal{U}$ and $\mathcal{B}$ locally set $A = g_2^{r'} A' = \mathrm{PedCom}(u, s, \boldsymbol{t})$.
3. $\mathcal{U}$ forms a verifiable encryption Q of the value s under his own key $ek_{\mathcal{U}} = \mathrm{g}_3^u$. (This encryption can be proved correct relative to commitment A.) Q is signed by $\mathcal{U}$. $\mathcal{B}$ verifies the correctness of Q and the signature σ on Q. $\mathcal{U}$

obtains a CL signature from $\mathcal{B}$ on the values committed in A via the protocol for getting a signature on a set of committed values.

4. $\mathcal{B}$ debits 2^ℓ from account $pk_\mathcal{U}$, records the entry $(pk_\mathcal{U}, Q, \sigma)$ in his database D, and issues $\mathcal{U}$ a CL signature on Y.
5. $\mathcal{U}$ saves the wallet $W = (sk_\mathcal{U}, s, \boldsymbol{t}, \sigma_B(u, s, \boldsymbol{t}), J)$, where J is an ℓ-bit counter set to zero.

$\mathsf{Spend}(\mathcal{U}(W, pk_\mathcal{M}), \mathcal{M}(sk_\mathcal{M}, pk_\mathcal{B}, 2^\ell))$**:** The only change from System One is in the calculation of the security tag T and the subsequent proof Φ. Assume $info \in \{0,1\}^*$ and $R \in \mathbb{Z}_q^*$ are obtained as before.

1. $\mathcal{U}$ sends the serial number of the coin $S = f_s^{DY}(J) = \mathrm{g}_2^{1/(J+s+1)}$, the security tag $T = pk_\mathcal{U} f_{\boldsymbol{t}}^V(J)^R = \mathrm{g}_3^{u+Rt_0\prod_{\{i:V(J)_i=1\}} t_i}$, and a proof Φ of their validity to $\mathcal{M}$. The signature proof Φ consists of:

(a) $A_0 = \mathrm{PedCom}(J)$ and a proof that A is a commitment to an integer in the range $[0, \ldots, 2^\ell - 1]$; a commitment $A_1 = \mathrm{PedCom}(J^2)$ and a proof that it is a commitment to the square of the opening of A_0; and finally, a commitment $A_2 = \mathrm{PedCom}(V(J)) = \mathrm{PedCom}(J \circ J^2)$ and a proof that it was formed correctly.

(b) $B_i = \mathrm{PedCom}(V(J)_i)$ for $i = 1$ to 3ℓ (commitments to the bits of $V(J)$) and proof that each B_i opens to either 0 or 1; that is, $PK\{(\gamma_1, \gamma_2) : B_i/g_1 = {h_1}^{\gamma_1} \vee B_i = {h_1}^{\gamma_2}\}$,

(c) proof that A_2 and $\{B_i\}$ are consistent; $PK\{(\gamma) : A_2/g_1 \prod_{i=1}^{3\ell} B_i^{2^{i-1}} = {h_1}^\gamma\}$,

(d) commitments to $\mathcal{U}$'s secret key u, and wallet secrets s and $\boldsymbol{t}$: $C = \mathrm{PedCom}(u)$, $D = \mathrm{PedCom}(s)$, $E_i = \mathrm{PedCom}(t_i)$ for $i = 0$ to 3ℓ, and proof of knowledge of a CL signature on the openings of C, D, and all E_i's in that order,

(e) the following commitments that will help in proving that T was formed correctly: $F_0 = E_0$, $F_i = \mathrm{PedCom}(\prod_{\{j \le i : V(J)_j = 1\}} t_j)$ for $i = 1$ to 3ℓ,

(f) and a proof that $S = f_s^{DY}(J)$ and $T = \mathrm{g}_3^u(f_{\boldsymbol{t}}^V(J))^R$. Proving the statement about S is done as in System One; proving the statement about T can be done as follows: Prove, for every $1 \le i \le 3\ell$ that F_i was formed correctly, corresponding to the committed value t_i and the value contained in the commitment B_i. That is to say:

$$
\begin{aligned}
PK\{(\alpha, \beta, \delta) : F_i = \mathrm{PedCom}(\alpha) \ \wedge \ & F_{i-1} = \mathrm{PedCom}(\beta) \ \wedge \\
& E_i = \mathrm{PedCom}(\delta) \ \wedge \ \Big(\big(B_i = \mathrm{PedCom}(0) \ \wedge \ \alpha = \beta\big) \ \vee \\
& \big(B_i = \mathrm{PedCom}(1) \ \wedge \ \alpha = \beta\delta\big)\Big)\}
\end{aligned}
$$

Note that, if all F_i's are formed correctly, then $F_{3\ell}$ is a commitment to the discrete logarithm of the value $f_{\boldsymbol{t}}^V(J) = t_0 \prod_{i=1}^{3\ell} t_i^{(V(J))_i}$. So we can prove the validity of tag T as follows:

$$PK\{(\alpha, \beta) \ : \ T = \mathrm{g}_3^{\alpha+\beta R} \wedge C = \mathrm{PedCom}(\alpha) \wedge F_{3\ell} = \mathrm{PedCom}(\beta)\}$$

2. $\mathcal{M}$ and $\mathcal{U}$ proceed exactly as before.

As in System One, using the Fiat-Shamir heuristic we turn these multiple proofs of knowledge into one signature, secure in the random-oracle model.

The Deposit protocol and VerifyGuilt algorithm follow the same outline as System One. During deposit, the bank may store only (S, R, T) in database L to obtain all desired functionality – except the ability to convince a third party of anything, such as a double-spender's identity or which coins belong to him. In the Identify protocol, the proof of guilt Π_G will additionally include the part of the user's secret key recovered as g_3^u.

$\mathsf{Trace}(\zeta, S, pk_{\mathcal{U}}, \Pi_G, D, 2^\ell)$**:** Parse Π_G as (dk, π_1, π_2) and $pk_{\mathcal{U}}$ as $(ek_{\mathcal{U}}, vk_{\mathcal{U}})$. The bank checks that $e(\mathrm{g}_3, dk) = ek_{\mathcal{U}}$; if not, it aborts. Otherwise the bank searches its database D, generated during the withdrawal protocol, for verifiable encryptions tagged with the public key $pk_{\mathcal{U}}$. For each matching entry $(pk_{\mathcal{U}}, Q, \sigma)$, $\mathcal{B}$ does the following: (1) runs the Camenisch-Damgård decryption algorithm on Q with dk to recover the value s; and (2) then for $i = 0$ to $2^\ell - 1$, outputs a serial number $S_i = f_s^{DY}(i)$ and a proof of ownership $\Pi_i = (Q, \sigma, dk, i)$.

$\mathsf{VerifyOwnership}(\zeta, S, \Pi, pk_{\mathcal{U}}, 2^\ell)$**:** Parse Π as (Q, σ, dk, i). Check that σ is $pk_{\mathcal{U}}$'s signature on Q and that i is in the range $[0, \ldots, 2^\ell - 1]$. Next, verify that dk is $pk_{\mathcal{U}}$'s decryption key by checking that $e(\mathrm{g}_3, dk) = ek_{\mathcal{U}}$. Finally, run the verifiable decryption algorithm on Q with dk to recover s' and verify that $S = f_{s'}^{DY}(i)$. If all checks pass, the algorithm accepts, otherwise, it rejects.

Efficiency Discussion of System Two. In Withdraw, the number of communication rounds does not change from System One, but one of the multi-base exponentiations will involve 3ℓ bases and hence its computation will take longer. Let us discuss the computational load of the verifiable encryption. For a cheating probability of at most 2^{-k}, the user must additionally compute k exps and $2k$ encryptions with the bilinear El Gamal scheme. To verify, the bank also must perform k exps but only k encryptions. Upon recovery of the double-spender's secret key, the bank needs to perform at most k decryptions and k exponentiations. Furthermore, the bank needs to compute all the 2^ℓ serial numbers each of which takes one exponentiation.

In Spend, the user must compute a total of $7 + 9\ell$ and $17 + 21\ell$ multi-base exponentiations for the commitments and the signature proof, respectively. The merchant and the bank also need to perform $17+21\ell$ multi-base exponentiations. For each of these, there is one multi-base exponentiation with 3ℓ exponents while all the others involve two to four bases.

Theorem 2. *System Two supports the algorithms* (BKeygen, UKeygen, Withdraw, Spend, Deposit, Identify, VerifyGuilt, Trace, VerifyOwnership) *and guarantees balance, identification of double-spenders, tracing of double-spenders, anonymity of users, and weak and strong exculpability under the Strong RSA, y-DDHI, and SF-DDH assumptions in the random oracle model.*

Proof of Theorem 2 appears in the full version of this paper [13]. Random oracles can be removed as discussed in System One.

Acknowledgments. We are grateful to Yevgeniy Dodis for helpful comments.

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.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, 2000.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *NDSS '05*, pp. 29–43, 2005. Full version available at http://eprint.iacr.org/2005/028.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, vol. 1233, pp. 480–494, 1997.
4. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT 2004*, vol. 3027 of *LNCS*, pp. 54–73, 2004.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures using strong Diffie-Hellman. In *CRYPTO '04*, vol. 3152 of LNCS, pp. 41–55, 2004.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, vol. 1807 of *LNCS*, pp. 431–444, 2000.
7. S. Brands. Electronic cash systems based on the representation problem in groups of prime order. In *Preproceedings of CRYPTO '93*, pp. 26.1–26.15, 1993.
8. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 318–333, 1997.
9. S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy.* PhD thesis, Eindhoven Inst. of Tech. The Netherlands, 1999.
10. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *SIAM*, pp. 457–466, 1995.
11. 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*, vol. 1976 of *LNCS*, pp. 331–345, 2000.
12. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04*, vol. 3352 of LNCS, pp. 120–133, 2004.
13. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash, 2005. Full version available at http://eprint.iacr.org/2005/060.
14. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, vol. 2576 of *LNCS*, pp. 268–289, 2003.
15. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, vol. 3152 of *LNCS*, pp. 56–72, 2004.

16. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, vol. 1592, pp. 107–122, 1999.
17. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, vol. 1666 of *LNCS*, pp. 413–430, 1999.
18. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO '03*, vol. 2729 of *LNCS*, pp. 126–144, 2003.
19. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of LNCS, pp. 410–424, 1997.
20. J. L. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998.
21. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 561–575, 1998.
22. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pp. 199–203. Plenum Press, 1982.
23. D. Chaum. Blind signature systems. In *CRYPTO '83*, pp. 153–156. Plenum, 1983.
24. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
25. D. Chaum. Online cash checks. In *EUROCRYPT '89*, vol. 434, pp. 289–293, 1989.
26. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '90*, vol. 403 of *LNCS*, pp. 319–327, 1990.
27. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, vol. 740 of *LNCS*, pp. 89–105, 1993.
28. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, vol. 839 of *LNCS*, pp. 174–187, 1994.
29. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, vol. 2501 of *LNCS*, 2002.
30. Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Public Key Cryptography*, vol. 2567 of *LNCS*, pp. 1–17, 2003.
31. Y. Dodis and A. Yampolsky. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography '05*, vol. 3386 of LNCS, pp. 416–431, 2005.
32. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, vol. 263 of LNCS, pp. 186–194, 1986.
33. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30, 1997.
34. S. Jarecki and V. Shmatikov. Handcuffing big brother: an abuse-resilient transaction escrow scheme. In *EUROCRYPT*, vol. 3027 of *LNCS*, pp. 590–608, 2004.
35. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT '04*, vol. 3027 of LNCS, pp. 571–589, 2004.
36. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51, Number 2:231–262, 2004.
37. T. Okamoto and K. Ohta. Disposable zero-knowledge authentications and their applications to untraceable elec. cash. In *CRYPTO*, vol. 435, pp. 481–496, 1990.
38. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '92*, vol. 576 of *LNCS*, pp. 129–140, 1992.
39. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
40. M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 209–219, 1995.
41. Y. S. Tsiounis. *Efficient Electonic Cash: New Notions and Techniques.* PhD thesis, Northeastern University, Boston, Massachusetts, 1997.