Compact E-Cash

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH-8803 Rüschlikon
jca@zurich.ibm.com

Susan Hohenberger
CSAIL
Massachusetts Inst. of Technology
Cambridge, MA 02139, USA
srhohen@mit.edu

Anna Lysyanskaya
Computer Science Department
Brown University
Providence, RI 02912, USA
anna@cs.brown.edu

March 27, 2006

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. We present two alternate constructions. One construction shares the same complexities with our first result but requires a strong bilinear map assumption that is only conjectured to hold on MNT curves. The second construction works on more general types of elliptic curves, but 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 [27, 28], and extensively studied since [31, 40, 32, 11, 24, 12, 54, 39, 55, 5]. 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 [28, 29, 30], the bank is on-line in each transaction to ensure that no coin is spent twice, and each merchant must consult the bank before accepting a payment. In the *off-line* [31] 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) [54, 15], 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 [44] 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 [49] 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 [45]. 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. [45]; 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 and Decisional Diffe-Hellman Inversion (y-DDHI) [6, 37] assumptions in combination with either the External Diffie-Hellman (XDH) [42, 53, 46, 7, 3] or the Sum-Free DDH [36] assumption for groups with bilinear maps. The trade-off between these later two assumptions is that we achieve a more efficient construction using the XDH assumption, but the Sum-Free DDH assumption is conjectured to hold in a wider class of bilnear groups (e.g., those over supersingular curves).

Using the Sum-Free DDH assumption, 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. Alternatively, using the XDH assumption, we give a scheme where the withdrawal and the spending protocols have complexity $\mathcal{O}(\ell + k)$, and it takes $\mathcal{O}(\ell + k)$ bits to store all the coins. These schemes are 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 *only* the Strong RSA [41, 4] and the y-DDHI [37] assumptions in the random-oracle model. This scheme under a reduced set of assumptions does not allow traceability, however. If a user over-spends his wallet, only his public key is recovered. 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 [18, 19]. 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 [14, 18].

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 $B_i = F_t(i)$, $0 \leq i \leq 2^\ell - 1$. 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 $0 \leq i \leq 2^\ell - 1$ and public value R_i provided by the merchant.

Recently, Dodis and Yampolskiy [37] 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. (In the sequel, we denote this PRF as $F^{DY}_{(\cdot)}(\cdot)$.) Using 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 $S_i = F^{DY}_s(i)$ and $T_i = g^{sk_{\mathcal{U}}}(F^{DY}_t(i))^{R_i}$ 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 $(T_1^{R_2}/T_2^{R_1})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2} B_i^{R_1R_2}/pk_{\mathcal{U}}^{R_1} B_i^{R_1R_2})^{(R_2-R_1)^{-1}} = (pk_{\mathcal{U}}^{R_2-R_1})^{(R_2-R_1)^{-1}} = pk_{\mathcal{U}}$. 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, 16, 22] 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 $0 \leq i < 2^\ell$, 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 $e : \widehat{G}_1 \times \widetilde{G}_2 \mapsto \overline{G}$. Let $sk_{\mathcal{U}}$ be an element of $\mathbb{Z}_q$. Let $\overline{pk}_{\mathcal{U}} = e(\widehat{g}_1, \widetilde{g}_2^{sk_{\mathcal{U}}})$. Using ideas from identity-based encryption [8, 2], we know how to realize a cryptosystem that uses $\overline{pk}_{\mathcal{U}}$ as a public key, such that in order to decrypt it is sufficient to know the value $\widehat{g}_1^{sk_{\mathcal{U}}}$. So, in our scheme, the user $\mathcal{U}$ would encrypt s under $\overline{pk}_{\mathcal{U}}$ using the cryptosystem as described by Ateniese et al [2]. From the double-spending equations, the same way as before, the bank infers the value $\widehat{g}_1^{sk_{\mathcal{U}}}$. This value now allows the bank to decrypt s.

This is almost the solution, except for the following subtlety: if $\widehat{G}_1$ has a bilinear map, then the decisional Diffie-Hellman problem may be easy, and so the Dodis-Yampolskiy construction is not a PRF in this setting! We have two solutions for this problem.

First, we can hope that there are bilinear maps where DDH remains hard in $\widehat{G}_1$ (e.g., there is not an efficiently-computable isomorphism from $\widetilde{G}_2$ to $\widehat{G}_1$), apply one of these bilinear groups, and keep using the Dodis-Yampolskiy PRF. In fact, this idea is already formalized as the External Diffie-Hellman (XDH) [42, 53, 46, 7, 3] assumption, and there is growing evidence that it may hold for bilinear groups instantiated with either the Weil or Tate pairings over MNT curves [47, 3][7, Sec. 8.1], even though this is known to be false for supersingular curves [43].

As a second, less risky, solution, we instead assume Sum-Free Decisional Diffie-Hellman [36], which is believed to hold for all bilinear groups, and slightly change the construction. This is why this variant of our scheme is a factor of ℓ more expensive than the others. The details of this construction are given in Section 4.2.

One of the big open 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 $\mathsf{BKeygen}(1^k, params)$ 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, $\mathsf{UKeygen}(1^k, params)$ 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 $sk_{\mathcal{U}}$ 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 $\mathsf{Identify}(params, S, \pi_1, \pi_2)$ 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 $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$ 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 $\mathsf{Trace}(params, S, pk_{\mathcal{U}}, \Pi_G, D, n)$ 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 for the casual reader. The more interested reader will want to skip to the more elaborate formal definitions given in Section 2.1.

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 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. This informal description should be sufficient for a general understanding of our security guarantees. We now include more precise definitions.

2.1 Formal Definitions

Our goal is to give a formal definition for electronic cash. We want to obtain a definition which would make sense independently on whether a given scheme is realized in the plain model, common random string model, common parameters model, random-oracle model, or any other variety of a model.

From the informal definitions, the reader may recall that our definition of e-cash relies on the existence of a knowledge extractor $\mathcal{E}$ that extracts the serial number of all the coins that an adversarial user withdraws in a given Withdraw transaction; and the existence of the simulator $\mathcal{S}$ which simulates the malicious merchant's view of the Spend protocol without access to any users' keys or wallets.

Usually, the way that knowledge extractors and zero-knowledge simulators are defined depends on the model in which security is proven (e.g., plain model, common random string model, common parameters model, random-oracle model).

In order to obtain a definition that works equally well in *any* model, we require that a particular protocol is a proof of knowledge or a zero-knowledge proof in the model in which security of the construction is proven. For example, we require that part of the $\mathsf{Withdraw}$ protocol is a proof of knowledge of the serial numbers of the coins withdrawn by the user, or part of the Spend protocol is a zero-knowledge proof.

In whatever model we are working, however, we will need concurrently composable proofs of knowledge and concurrently composable zero-knowledge proofs.

Let us explain how to specify a knowledge extractor $\mathcal{E}_{\mathsf{Prot},l}$ for a proof of knowledge protocol Prot for language l, (resp., zero-knowledge simulator $\mathcal{S}_{\mathsf{Prot},l}$ for protocol Prot for proving membership in language l) in a (relatively) model-independent fashion. First, note that in some models, such as a public-parameter model, an extractor or simulator is allowed access to some auxiliary information not provided to a participant in the protocol. We denote denote these by *auxext* and *auxsim*, respectively.

Another resource that a knowledge extractor or zero-knowledge simulator must receive, is access to the adversary. This includes many possibilities; here are a few examples:

- (IO) Input-output interaction with the adversary. This is typical in the UC-framework.
- (BB) Black-box interaction with the adversary. This means that the adversary can be reset to a previous state.
- (NBB) Non-black-box access. This means that we are given the description of the adversary.

Each of these access models can be further augmented with the random-oracle model. For example, the IO-RO model includes input-output access to the adversary's hash function module.

Let $X\text{-}Y(\mathcal{A})$ be our access model to the adversary $\mathcal{A}$, where, for example, $X \in \{IO, BB, NBB\}$, and $Y \in \{\varepsilon, RO\}$. By $\mathsf{Alg}^{X\text{-}Y(\mathcal{A})}$ we denote that the algorithm Alg has this type of access to the adversary $\mathcal{A}$.

So, for example, if Alg is an algorithm interacting with adversary $\mathcal{A}$ in the $X\text{-}Y$ model, running on input x, then we denote it by $\mathsf{Alg}^{X\text{-}Y(\mathcal{A})}(x)$.

In the sequel, by "proof protocol," we mean a protocol between a prover and a verifier.

Thus, a knowledge extractor for proof protocol Prot for language l in the $X\text{-}Y$ model would be denoted as $\mathcal{E}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Prot},l}(params, auxext, x)$. By definition of what it means to be a knowledge extractor, for property formed $(params, auxext)$, $\mathcal{E}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Prot},l}(params, auxext, x)$ will, with high probability, in expected polynomial time, output a w such that $(x, w) \in l$ whenever the probability that the verifier accepts x when interacting with $\mathcal{A}$ in the $X\text{-}Y$ model, is non-negligible.

Similarly, a zero-knowledge simulator for proof protocol Prot for language l in the $X\text{-}Y$ model would be denoted as $\mathcal{S}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Prot},l}(params, auxsim, x)$. By definition of what it means to be a zero-knowledge simulator, $\mathcal{S}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Prot},l}(params, auxsim, x)$ will, when interacting with $\mathcal{A}$ in the $X\text{-}Y$ model, produce a view that is indistinguishable from the view $\mathcal{A}$ obtains when interacting with the prover.

Balance. Let a $\mathsf{Withdraw}$ protocol be given. Recall that the bank's input to this protocol includes the user's public key $pk_{\mathcal{U}}$. Let us break up the $\mathsf{Withdraw}$ protocol into three parts: $\mathsf{Withdraw}_b$, $\mathsf{Withdraw}_m$, and $\mathsf{Withdraw}_e$ ("b," "m," and "e" stand for "beginning," "middle," and "end," respectively). $\mathsf{Withdraw}_b$ is the part of the protocol that ends with the first message from the user to the bank. $\mathsf{Withdraw}_e$ part of the protocol is the last message from the bank to the user.

The middle of the protocol is denoted $\mathsf{Withdraw}_m$. Let us think of $\mathsf{Withdraw}_e$ as a proof protocol, where the user is the Prover, and the bank is the Verifier. The Verifier's output of $\mathsf{Withdraw}_m$ can be thought of as the bank's decision for whether or not to proceed to $\mathsf{Withdraw}_e$ and send the last message from the bank to the user.

Let m_1 denote the first message that the user sends to the bank in this protocol. Let b_1 denote the state information of the bank at the moment that m_1 was received.

The balance property requires that:

1. In whatever model the scheme is given, there exists an efficiently decidable language l_S and an extractor $\mathcal{E}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Withdraw}_m, l_S}(\mathit{params}, \mathit{auxext}, pk_{\mathcal{U}}, b_1, m_1)$ such that for all b_1 computed by the bank, and for all m_1, it extracts $w = (S_1, \ldots, S_n, \mathit{aux})$ such that $(b_1, m_1, w) \in l_S$ whenever the probability that the bank accepts in the $\mathsf{Withdraw}_m$ part of the protocol is non-negligible. We say that the extractor outputs $(b_1, m_1, w) \in l_S$ in that case.
2. (If we are in the public parameters model, assume that the values params and auxext are fixed. Assume that $pk_{\mathcal{B}}$ was generated appropriately.)

   On input $(\mathit{params}, pk_{\mathcal{B}})$, the adversary $\mathcal{A}$ plays the following game: $\mathcal{A}$ executes the $\mathsf{Withdraw}$ and $\mathsf{Deposit}$ protocols with the bank as many time as it wishes. (It can simulate running the Spend protocol with itself.) Let $(b_{1,i}, m_{1,i}, w_i) \in l_S$ be the output of $\mathcal{E}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Withdraw}_m, l_S}(\mathit{params}, \mathit{auxext}, pk_i, b_{1,i}, m_{1,i})$ if the ith withdrawal protocol was successful. Recall that $w_i = (S_{i,1}, \ldots, S_{i,n}, \mathit{aux})$ is a list of n serial numbers; we say that these *belong* to pk_i. Let $A_f = \{S_{i,j} \mid 1 \le i \le f, 1 \le j \le n\}$ be the list of serial numbers after f executions of the $\mathsf{Withdraw}$ protocol. $\mathcal{A}$ wins the game if for some f, in some $\mathsf{Deposit}$ protocol, the honest bank accepts a coin with serial number $S \notin A_f$. We require that no probabilistic polynomial-time adversary succeeds in this game with non-negligible probability.

Identification of double-spenders. This property guarantees that no PPT adversary has a non-negligible probability of winning the following game:

(If we are in the public parameters model, assume that the values params and auxext are fixed; assume that $pk_{\mathcal{B}}$ was generated appropriately.)

On input $(\mathit{params}, pk_{\mathcal{B}})$, the adversary $\mathcal{A}$ plays the following game: $\mathcal{A}$ executes the $\mathsf{Withdraw}$ protocol and Spend protocol with the bank as many time as it wishes. Let $(b_{1,i}, m_{1,i}, w_i) \in l_S$ be the output of $\mathcal{E}^{X\text{-}Y(\mathcal{A})}_{\mathsf{Withdraw}_m, l_S}(\mathit{params}, \mathit{auxext}, pk_i, b_{1,i}, m_{1,i})$ if the ith withdrawal protocol was successful. Recall that $w_i = (S_{i,1}, \ldots, S_{i,n}, \mathit{aux})$ is a list of n serial numbers; recall that we say that these *belong* to pk_i. Let A_i be the list of serial numbers that belong to public key pk_i. $\mathcal{A}$ wins the game if for some f, in some Spend protocol, the bank, simulating an honest merchant, accepts a coin with serial number $S \in A_f$ twice, i.e. outputs (S, π_1) and (S, π_2), and yet $\mathsf{Identify}(\mathit{params}, S, \pi_1, \pi_2)$ output a public key pk and a proof Π_G such that $\mathsf{VerifyGuilt}(\mathit{params}, S, pk, \Pi_G)$ does not accept.

Tracing of double-spenders. Here, we play the same sort of game with the adversary as above, but we are more generous as to what constitutes the adversary's success. Suppose that for some f, in some Spend protocol, the honest merchant accepts a coin with serial number S twice, i.e. outputs (S, π_1) and (S, π_2), and computes $\mathsf{Identify}(\mathit{params}, S, \pi_1, \pi_2) \to (pk_{\mathcal{U}}, \Pi_G)$. Let $\{S_i\}$ be the set of serial numbers and $\{\Pi_i\}$ be the set of ownership proofs output by $\mathsf{Trace}(\mathit{params}, S, pk_{\mathcal{U}}, \Pi_G, D, n)$. $\mathcal{A}$ wins if (1) there exists some serial number $S' \in A_f$ such that $S' \notin \{S_i\}$, or (2) there exists a serial number $S_j \in \{S_i\}$ such that $\mathsf{VerifyOwnership}(\mathit{params}, S_j, \Pi_j, pk_{\mathcal{U}}, n)$ does not accept.

Anonymity of users. (If we are in the public parameters model, assume that the values *params* and *auxsim* are fixed; *auxsim* is a value available to a simulator but not available to regular participants in this system.)

Here, let us consider an adversary $\mathcal{A}$ that forms the bank's public key $pk_{\mathcal{B}}$ and then issues the following queries, as many as it wants, in any order:

PK of i In this query, $\mathcal{A}$ may request and receive the public key pk_i of user i, generated honestly as $(pk_i, sk_i) \leftarrow \mathsf{UKeygen}(1^k, \mathit{params})$.

Withdraw with i In this query, $\mathcal{A}$ executes the $\mathsf{Withdraw}$ protocol with user i:

$$\mathsf{Withdraw}(\mathcal{U}(pk_{\mathcal{B}}, sk_i, n), \mathcal{A}(\mathit{state}, n))$$

where *state* is $\mathcal{A}$'s state; let us denote the user's output after the j'th Withdraw query by W_j; note that W_j may be an error message, in that case we say that W_j is an *invalid* wallet.

Spend from wallet j In this query, if wallet j W_j is defined, $\mathcal{A}$ executes the Spend protocol from this wallet: $\mathsf{Spend}(\mathcal{U}(W_j), \mathcal{A}(\mathit{state}))$, where *state* is the adversary's state.

We say that $\mathcal{A}$ is *legal* if it only spends from valid wallets, and *never* asks to spend more than n coins from the same wallet W_j.

We require the existence of a simulator $\mathcal{S}^{X\text{-}Y(\cdot)}(\mathit{params}, \mathit{auxsim}, \cdot)$ such that for all $pk_{\mathcal{B}}$, no legal adversary $\mathcal{A}$ can distinguish whether he is playing Game R(eal) or Game I(deal) below with non-negligible advantage:

Game R The queries $\mathcal{A}$ issues are answered as described above.

Game I The PK and Withdraw queries $\mathcal{A}$ issues are answered as described above, but in the Spend query, instead of interacting with $\mathcal{U}(W_j)$, $\mathcal{A}$ interacts with the simulator $\mathcal{S}^{X\text{-}Y(\mathcal{A})}(\mathit{params}, \mathit{auxsim}, pk_{\mathcal{B}})$.

(The reason this captures anonymity is that the simulator does not know on behalf of which honest user he is spending the coin.)

Exculpability. Recall that the exculpability property guarantees that only users who really are guilty of double-spending a coin can ever get convicted of being a double-spender. Exculpability comes in two flavors: *weak* exculpability means that only users who double-spent *some* coin can be convicted; while *strong* exculpability means that a guilty user would only be responsible for the coins that he indeed double-spent, i.e., his guilt can be quantified and he can be punished according to guilt.

To define weak exculpability, we have the adversary $\mathcal{A}$ launch the following attack against a user **U** (assume n is fixed):

Setup The system parameters $params$ are generated and the user's keys $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$ are chosen. The adversary chooses the bank's public key $pk_{\mathcal{B}}$ in an adversarial fashion.

Queries The adversary $\mathcal{A}$ issues queries to interact with the user $\mathcal{U}$, as follows:

Withdraw wallet j The adversary plays the bank's side of the Withdraw protocol with the $\mathcal{U}$. The user outputs the wallet W_j.

Spend from wallet j The adversary plays the merchant's side of the Spend protocol where $\mathcal{U}$'s input is wallet W_j. For each wallet, the adversary is allowed to execute this query up to n times.

Success criterion In the end, the adversary $\mathcal{A}$ outputs the values $(S, \Pi)$and wins the game if VerifyOwnership$(params,\ S, \Pi, pk_{\mathcal{U}}, n)$ accepts.

We say that an electronic cash scheme guarantees weak exculpability if the probability that the adversary wins this game is negligible.

To define strong exculpability, we first insist that corresponding to any wallet W obtained by an honest user even from an adversarial bank, there exist at most n valid serial numbers. Denote the set of serial numbers corresponding to W by A_W.

The strong exculpability game is somewhat similar to the weak exculpability game: the setup and the withdraw query are the same. The only things that are different are as follows:

Queries The adversary issues queries to interact with the user $\mathcal{U}$, in addition to Withdraw above, as follows:

Spend from wallet j The adversary plays the merchant's side of the Spend protocol where $\mathcal{U}$'s input is wallet W_j. For each wallet, the adversary is allowed to execute this query as many times as he wishes. Recall that the wallet keeps state information about how many coins have already been spent, and a wallet where n coins have been spent is not well-defined. To make this query well-defined in that case, let us say that once n coins are spent from a wallet W_j, the state information is reset such that it is the same as it was before any coins were spent.

As a result of this query, the user may produce a serial number that she had already produced before. Let A_j be the set of all serial numbers that the user has produced in a Spend protocol for wallet j so far. If for any j, $|A_j| > n$, then the adversary is given the sets of serial numbers $A_{W_{j'}}$ for *all* wallets $W_{j'}$. (This is what the adversary is entitled to because of the traceability requirement.) Let A'_j be the set of serial numbers for wallet W_j that are known to the adversary. Thus $A'_j = A_j$ if no double-spending has ever occured, and $A'_j = A_{W_j}$ otherwise.

Let A_{ds} be the set of serial numbers that the user has produced more than once (i.e., double-spent) so far.

Success criterion In the end, the adversary outputs the values (S, Π) and wins the game if *one* of the following conditions is satisfied:

1. The adversary made up a bogus serial number and managed to pin it on the user; i.e., for all j, $S \notin A'_j$ and yet (a) VerifyOwnership$(params, S, \Pi, pk_{\mathcal{U}}, n)$ or (b) VerifyGuilt$(params, S,\ \Pi, pk_{\mathcal{U}}, n)$ accepts.
2. The user has spent fewer than n coins from all of his wallets, and yet the adversary managed to produce proof that the user owns or double-spent some coin in wallet j;

i.e., $|A_{j'}| < n$ for all j', $S \in A_j$ for some j, and yet (a) $\mathsf{VerifyOwnership}(params, S, \Pi, pk_{\mathcal{U}}, n)$ or (b) $\mathsf{VerifyGuilt}(params, S, \Pi, pk_{\mathcal{U}}, n)$ accepts.

3. The adversary successfully accuses the user of double-spending a coin which was *not* in fact double-spent: $S \notin A_{ds}$ and $\mathsf{VerifyGuilt}(params, S, \Pi, pk_{\mathcal{U}}, n)$ accepts.

In our model, a person *is* their secret key $sk_{\mathcal{U}}$, so any coins obtained by a party in possession of $sk_{\mathcal{U}}$ belong to $pk_{\mathcal{U}}$, and any coins double-spent with knowledge of $sk_{\mathcal{U}}$ were double-spent by $pk_{\mathcal{U}}$. For our particular construction in System Two, where one part of $sk_{\mathcal{U}}$ is revealed after double-spending, we define party $\mathcal{U}$ as those persons in possession of *both* parts of $sk_{\mathcal{U}}$.

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 Bilinear Maps

Let Bilinear_Setup be an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear mapping as $\gamma = (q, \widehat{g}_1, \widehat{h}_1, \widehat{G}_1, \widetilde{g}_2, \widetilde{h}_2, \widetilde{G}_2, \overline{G}, e)$. We follow the notation of Boneh, Lynn, and Shacham [9]:

1. $\widehat{G}_1, \widetilde{G}_2$ are both (multiplicative) groups of prime order $q = \Theta(2^k)$;
2. each element of $\widehat{G}_1$, $\widetilde{G}_2$, and $\overline{G}$ has a unique binary representation;
3. the setup algorithm provides two generators for both groups where $\widehat{G}_1 = \langle \widehat{g}_1 \rangle = \langle \widehat{h}_1 \rangle$ and $\widetilde{G}_2 = \langle \widetilde{g}_2 \rangle = \langle \widetilde{h}_2 \rangle$.
4. ψ is an efficiently computable isomorphism from $\widetilde{G}_2$ to $\widehat{G}_1$, with $\psi(\widetilde{g}_2) = \widehat{g}_1$ and $\psi(\widetilde{h}_2) = \widehat{h}_1$;
5. e is an efficiently computable bilinear map $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$ such that:
   - (Bilinear) for all $\widehat{g}_1 \in \widehat{G}_1$, $\widetilde{g}_2 \in \widetilde{G}_2$, and $a, b \in \mathbb{Z}_q$, $e(\widehat{g}_1^a, \widetilde{g}_2^b) = e(\widehat{g}_1, \widetilde{g}_2)^{ab}$;
   - (Non-degenerate) if $\widehat{g}_1$ is a generator of $\widehat{G}_1$ and $\widetilde{g}_2$ is a generator of $\widetilde{G}_2$, then $e(\widehat{g}_1, \widetilde{g}_2)$ generates $\overline{G}$.

3.2 Complexity Assumptions

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

Strong RSA Assumption [4, 41]: 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 $h^e \equiv g \bmod n$. 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) [6, 37]: 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.[1]

External Diffie-Hellman Assumption (XDH) [42, 53, 46, 7, 3]: Suppose $\mathsf{Bilinear\_Setup}(1^k)$ produces the parameters for a bilinear mapping $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$. The XDH assumption states that the Decisional Diffie-Hellman (DDH) problem is hard in $\widehat{G}_1$. This implies that there does *not* exist an efficiently computable isomorphism $\psi' : \widehat{G}_1 \to \widetilde{G}_2$.

Sum-Free Decisional Diffie-Hellman Assumption (SF-DDH) [36]: Suppose that $g \in G$ is a random generator of order q. Let L be any polynomial function of $|q|$. Let $O_{\vec{a}}(\cdot)$ 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 $\vec{a} = (a_1, \ldots, a_L) \in \mathbb{Z}_q^L$. 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 $\mathcal{A}^{(\cdot)}$,

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

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

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 [52] or a composite [41, 35], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [33] or composite [21] moduli, (3) proof that a commitment opens to the product of two other committed values [20, 25, 13], (4) proof that a committed value lies in a given integer interval [26, 20, 20, 10], and also (5) proof of the disjunction or conjunction of any two of the previous [34]. 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 [23] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

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

[1]Others [6, 37] 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 $\langle e(g,g) \rangle$.

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 [38] 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 Yampolskiy [37], 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 $f_{G,q,g,s}(\cdot)$, which we simply denote as $f^{DY}_{g,s}(\cdot)$ for fixed values of (G, q, g), is defined as $f^{DY}_{g,s}(x) = g^{1/(s+x+1)}$.

This construction is secure under the y-DDHI assumption in G. As mentioned in the introduction, we could instead substitute in the Naor-Reingold PRF [48], 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 [50], 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 [18] 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 [19] and Boneh, Boyen and Shacham [7], yielding shorter signatures in practice.

3.6 Verifiable Encryption

In Section 4.2, we apply a technique by Camenisch and Damgård [16] 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 [16] for the full discussion.

3.7 Bilinear El Gamal Encryption

In particular, we apply the verifiable encryption techniques above to the following bilinear variant of El Gamal encryption [8, 2]. Assume we run $\mathsf{Bilinear\_Setup}$ on 1^k to obtain $\gamma = (q, \widehat{g}_1, \widehat{h}_1, \widehat{G}_1, \widetilde{g}_2, \widetilde{h}_2, \widetilde{G}_2, \overline{G}, e)$, where we have bilinear map $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$. Let (G, E, D) denote the standard key generation, encryption, and decryption algorithms. On input $(1^k, \gamma)$, the key generation algorithm G outputs a key pair $(\overline{pk}, \widehat{sk}) = (e(\widehat{g}_1, \widetilde{g}_2)^u, \widehat{g}_1^u)$ for a random $u \in \mathbb{Z}_q$. The main idea is that the value $\widehat{g}_1^u$ is enough to decrypt.

To encrypt a message $\overline{m} \in \overline{G}$ under $\overline{pk}$, select a random $k \in \mathbb{Z}_q$ and output the ciphertext $c = (\widetilde{g}_2^k, \overline{pk}^k \overline{m}) = (\widetilde{g}_2^k, e(\widehat{g}_1, \widetilde{g}_2)^{uk} \overline{m})$. Then, to decrypt $c = (\widetilde{c_1}, \overline{c_2})$ with the value $\widehat{g}_1^u$, simply compute $\overline{c_2}/e(\widehat{g}_1^u, \widetilde{c_1})$. This encryption scheme is known to be semantically-secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption [8, 2], i.e., given $(\widetilde{g}_2, \widetilde{g}_2^a, \widetilde{g}_2^b, \widetilde{g}_2^c, \overline{X})$ for random $a, b, c \in \mathbb{Z}_q$ and $\overline{X} \in \overline{G}$, it is hard to decide if $\overline{X} = e(\widehat{g}_1, \widetilde{g}_2)^{abc}$.

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, size of our wallets is either $\mathcal{O}(\ell + k)$ or $\mathcal{O}(\ell \cdot k)$ bits depending on the underlying assumptions. The user's anonymity is based on *computational*, not *trust* assumptions. If a user does not overspend, he cannot be identified or traced. 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 four different groups:

- $\mathbf{G} = \langle \mathbf{g} \rangle$, where n is a special RSA modulus of $2k$ bits, $\mathbf{g}$ is a quadratic residue modulo n, and $\mathbf{h} \in \mathbf{G}$.

  Common uses: Pedersen commitments and RSA-based CL signatures.

- $G = \langle g \rangle$, where g is an element of prime order $q = \Theta(2^k)$, and h is an element in G. We assume that DDH is hard in G; when this group is the range of a bilinear mapping, we will denote it as $\overline{G}$ instead.

  Common uses: coin serial numbers, security tags, and range of bilinear mapping. In particular, we let $F_{g,s}^{DY}(x) = g^{\frac{1}{x+s+1}}$ denote the PRF due to Dodis and Yampolskiy [37].

- $\widehat{G}_1 = \langle \widehat{g}_1 \rangle = \langle \widehat{h}_1 \rangle$ and $\widetilde{G}_2 = \langle \widetilde{g}_2 \rangle = \langle \widetilde{h}_2 \rangle$, where both groups have the same prime order as $\overline{G}$, and there exists a bilinear mapping $e : \widehat{G}_1 \times \widetilde{G}_2 \rightarrow \overline{G}$.

  Common uses: first and second domain of bilinear mapping.

Our first scheme will not require $\widehat{G}_1$ and $\widetilde{G}_2$. Assume that, on input 1^k, each system is initialized with the necessary common parameters, denoted ζ, as specified above. We also define $\mathrm{PedCom}(x_1, \ldots, x_n; r) = h^r \Pi_{i=1}^n g_i^{x_i}$. Sometimes for simplicity we do not explicitly include the randomess r in the input to the commitment. $h, \{g_i\}$ are assumed to be publicly known elements of the appropriate group.

4.1 System One: Compact E-Cash 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 $\mathsf{BKeygen}(1^k, \zeta)$, 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 $\mathsf{UKeygen}(1^k, \zeta)$, each user $\mathcal{U}$ generates a unique key pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = (g^u, u)$ for a random $u \in \mathbb{Z}_q$. Recall that merchants are a subset of users.

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

IDEA: User's wallet of 2^ℓ coins is $(sk_{\mathcal{U}}, s, t, \sigma, J)$, where σ is the bank's signature on $(sk_{\mathcal{U}}, s, t)$ and J is an ℓ-bit counter.

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 $\mathbf{A}' = \mathrm{PedCom}(sk_{\mathcal{U}}, s', t; r)$ 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 $\mathbf{A} = \mathbf{g}^{r'}\mathbf{A}' = \mathrm{PedCom}(sk_{\mathcal{U}}, s' + r', t; r) = \mathrm{PedCom}(sk_{\mathcal{U}}, s, t; r)$.
3. $\mathcal{U}$ and $\mathcal{B}$ run the CL protocol for obtaining $\mathcal{B}$'s signature on committed values contained in commitment $\mathbf{A}$. As a result, $\mathcal{U}$ obtains $\sigma_{\mathcal{B}}(sk_{\mathcal{U}}, s, t)$.
4. $\mathcal{U}$ saves the wallet $W = (sk_{\mathcal{U}}, s, t, \sigma_B(sk_{\mathcal{U}}, s, t), J)$, where s, t are the wallet secrets, $\sigma_{\mathcal{B}}(sk_{\mathcal{U}}, s, t)$ 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. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ be a collision-resistant hash function. (An optimized version appears in Appendix A.)
The protocol follows the outline above exactly. The ZKPOK in step 3 can be done as follows:

1. Let $\mathbf{A} = \mathrm{PedCom}(J)$; prove that $\mathbf{A}$ is a commitment to an integer in the range $[0 \ldots 2^\ell - 1]$.

Outline of Spend Protocol:

1. User computes $R = H(pk_{\mathcal{M}}||\mathit{info})$, where $\mathit{info} \in \{0,1\}^*$ is provided by the merchant.
2. User sends a coin serial number and double-spending equation:
$$S = F^{DY}_{g,s}(J) \quad , \quad T = pk_{\mathcal{U}} F^{DY}_{g,t}(J)^R.$$
3. User sends a ZKPOK Φ of $(J, sk_{\mathcal{U}}, s, t, \sigma)$ such that:
   - $0 \leq J < 2^\ell$ (standard techniques [26, 20, 20, 10])
   - $S = F^{DY}_{g,s}(J)$ (see below)
   - $T = pk_{\mathcal{U}} F^{DY}_{g,t}(J)^R$ (see below)
   - VerifySig$(pk_{\mathcal{B}}, (sk_{\mathcal{U}}, s, t), \sigma)$ =true (CL signatures [18, 19])
4. If Φ verifies, $\mathcal{M}$ accepts the coin $(S, (R, T, \Phi))$ 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.

2. Let $\mathbf{B} = \text{PedCom}(u)$, $\mathbf{C} = \text{PedCom}(s)$, $\mathbf{D} = \text{PedCom}(t)$; prove knowledge of a CL signature from $\mathcal{B}$ on the openings of $\mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ in that order,
3. Prove $S = F^{DY}_{g,s}(J) = g^{1/(J+s+1)}$ and $T = pk_{\mathcal{U}} F^{DY}_{g,t}(J)^R = g^{u+R/(J+t+1)}$.
More formally, this proof is the following proof of knowledge:

$$PK\{(\alpha, \beta, \delta, \gamma_1, \gamma_2, \gamma_3) : \mathbf{g} = (\mathbf{AC})^\alpha \mathbf{h}^{\gamma_1} \wedge S = g^\alpha \wedge \\ \mathbf{g} = (\mathbf{AD})^\beta \mathbf{h}^{\gamma_2} \;\wedge\; \mathbf{B} = \mathbf{g}^\delta \mathbf{h}^{\gamma_3} \;\wedge\; T = g^\delta (g^R)^\beta\}$$

Use the Fiat-Shamir heuristic to turn all the proofs above into one signature of knowledge on the values $(S, T, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{g}, \mathbf{h}, n, g, pk_{\mathcal{M}}, R, \mathit{info})$. Call the resulting signature Φ.

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.

IDEA: The bank accepts coin $(S, (R, T, \Phi))$ if proof Φ verifies and (S, R) never appeared together before. If S appeared before, trigger Identify algorithm.

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).

We note that if $\mathcal{B}$ was satisfied with detecting/identifying double-spenders, without worrying about proving anything to a third party, it need only store (S, R, T) in L for each coin at a considerable storage savings.

$\mathsf{Identify}(\zeta, S, \pi_1, \pi_2)$**:** 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 the proof of guilt $\Pi_G = (\pi_1, \pi_2)$ and the identity $pk = \left({T_2}^{R_1}/{T_1}^{R_2}\right)^{(R_1-R_2)^{-1}}$.

IDEA: From two coins with same serial number S, anyone can recover the user's public key.

Let us explain why this produces the public key $pk_{\mathcal{U}}$ of the double-spender. Suppose coin S belonged to some user with $pk_{\mathcal{U}} = g^u$, then each T_i is of the form $g^{u+R_i\alpha}$ 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:

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

$\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$: Parse Π_G as (π_1, π_2) and each π_i as (R_i, T_i, Φ_i). Run $\mathsf{Identify}(params,$

IDEA: Anyone can check that two coins with same S allows to compute user's public key.

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 [18, 17]. 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 Appendix A. 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 4.1 *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.
Balance. *Part 1.* Let $\mathcal{A}$ be an adversary who executes f Withdraw protocols with an extractor $\mathcal{E} = \mathcal{E}^{BB\text{-}\varepsilon(\mathcal{A})}_{\mathsf{Withdraw}_{m,l_S}}$ acting as the bank (with knowledge of sk_B). Suppose that the proof of knowledge

in step 3 (for the opening of commitment $\mathbf{A}$) is done interactively, and let $\mathcal{E}' = \mathcal{E}_{pok}^{BB\text{-}\varepsilon(\mathcal{A})}$ be the extractor for that proof of knowledge. (We know that such an extractor exists due to previous results; see Section 3.3.) Then, for each Withdraw protocol, $\mathcal{E}$ acts exactly as an honest bank would, except during step 3 where $\mathcal{E}$ runs the code of $\mathcal{E}'$ to extract the values (u, s, t). From the value s, $\mathcal{E}$ can compute $w = (S_1, \ldots, S_n, u, s, t, r)$ such that $S_i = F_{g,s}^{DY}(i)$, for $i = 1$ to n, and $\mathbf{A} = \mathrm{PedCom}(u, s, t; r)$. At the end of each Withdraw execution, $\mathcal{E}$ outputs $(state_{\mathcal{E}}, \mathbf{A}, w) \in l_S$, where $\mathbf{A}$ is a commitment, purportedly equivalent to $\mathrm{PedCom}(u, s, t)$, that $\mathcal{A}$ sent during step 3 of Withdraw and the language l_S is the set of triples $(\cdot, \mathbf{A}, w)$, where the first element may be anything, but the second two must conform to the specification for w immediately above. Let $A_f = \{S_{i,j} | 1 \le i \le f, 1 \le j \le n\}$ be the list of serial numbers after f executions of the Withdraw protocol.

If the proof of knowledge in the Withdraw protocol is done non-interactively, then both $\mathcal{E}$ and $\mathcal{E}'$ must be given control over the random oracle in addition to their black-box access to $\mathcal{A}$.

Part 2. Due to the soundness of the underlying proof of knowledge protocols, we know that A_f contains all *valid* serial numbers that $\mathcal{A}$ can produce, except with negligible probability. Thus, if $\mathcal{A}$ is to succeed at its game, it must convince an honest $\mathcal{B}$ to accept a serial number for which it cannot generate an honest proof of validity with some non-negligible probability.

Now suppose $\mathcal{A}$ convinces an honest $\mathcal{B}$ to accept the invalid coin (S, π) during Deposit, where $S \notin A_f$ and $\pi = (R, T, \Phi)$. Then $\mathcal{A}$ must have concocted a *false* proof as part of the signature proof Φ that: (1) $\mathbf{A}$ opens to an integer in $[1, \ldots, 2^\ell]$ or (2) $\mathcal{A}$ knows a signature from $\mathcal{B}$ on the opening of $\mathbf{B}, \mathbf{C}, \mathbf{D}$ or (3) that S (and T) are well formed as in Γ. Case (1) happens with negligible probability $\nu_1(k)$ under the Strong RSA assumption [26, 20, 20, 10]. Case (2) happens with negligible probability $\nu_2(k)$ under the assumption that the CL signatures are secure (which requires the Strong RSA assumption [18] or an additional bilinear map assumption called LRSW [19]). Case (3) happens with negligible probability $\nu_3(k)$ under the discrete logarithm assumption [52] (this assumption will later be subsumed by the y-DDHI assumption in the theorem statement). Thus, the $\mathcal{A}$ succeeds in this case with negligible probability $\nu_1(k) + \nu_2(k) + \nu_3(k)$. Thereby, $\mathcal{A}$'s total success probability in both parts is also negligible.

Removing the random oracle. If Withdraw is interactive, then random oracles are not used in part 1. It is enough to give the extractor $\mathcal{E}'$ (and thus $\mathcal{E}$ as well) black-box access to $\mathcal{A}$.

A (programmable) random oracle is used, however, to build the signature proofs in part 2. This is necessary if a merchant is to prove to a bank during Deposit that it has not colluded with a user to concoct a fake coin. In the special case that the same entity is running the Withdraw and Spend protocols (and thus, there is no need for a Deposit protocol), we remove the need for random oracles all together by making each proof in the Spend interactive. Black-box access to the adversary is still required for our efficient proofs.

Identification of double-spenders: Let us first point out a case in which this property does not apply. Observe that whenever the bank issues a CL signature on wallet secrets s, s' such that $|s - s'| < 2^\ell$, the recipients of these signatures can both generate valid spending proofs for some coin $S = f_{g,s}^{DY}(1) = f_{g,s'}^{DY}(s - s' + 1) = g^{1/(s+1)}$. Thus, there will be two valid entries in the bank's database as (S, π_1) and (S, π_2). This may look suspicious, but since double-spending has *not* occurred, this property does not apply. That said, we point out that the domain from which s is sampled is large enough that this is not a problem. During step 2 of the Withdraw protocol, both the bank and user contribute to the randomness used to select $s \in \mathbb{Z}_q$. Thus, so long as one of them is honest, s will be 2^ℓ-far from any other s' with probability $2^{\ell+1}/q$.

We now return to our main proof. As defined in balance, let $\mathcal{E} = \mathcal{E}^{BB\text{-}\varepsilon(\mathcal{A})}_{\mathsf{Withdraw}_{m},l_S}$ be the extractor that interacts with the adversary $\mathcal{A}$ during the Spend protocols to extract a set of valid serial numbers $A_f = \{S_{i,j} | 1 \le i \le f, 1 \le j \le n\}$. Now, suppose the honest merchant $\mathcal{M}$ (simulated by the bank) accepts two coins (S, π_1) and (S, π_2) for some $S \in A_f$. (We already saw in balance that the adversary cannot get an honest merchant to accept $S \notin A_f$ with non-negligible probability.)

Parse each π_i as (R_i, T_i, Φ_i). Since an honest $\mathcal{M}$ chooses R at random during the Spend protocol, we know that $R_1 \neq R_2$ with high probability. If we show that, with high probability, each $T_i = pk_{\mathcal{U}} f^{DY}_{g,t}(J+1)^R$ for the same values of $pk_{\mathcal{U}}$, J and t, then the success of $\mathsf{Identify}(\zeta, S, \pi_1, \pi_2)$ in recovering $pk_{\mathcal{U}} = g^u$ follows directly from the correctness of the algorithm.

Since S is valid, we have $S = f^{DY}_{g,s}(J+1)$ for some $0 \le J < 2^\ell$ and $s \in \mathbb{Z}_q$ such that $\mathcal{A}$ "knows" $\sigma_B(u, s, t)$ for some u, t, which incidentally $\mathcal{E}$ extracted. (Note that either $\mathcal{A}$ is proving knowledge of the same σ_B in both transactions or it is actually spending two different coins in which case $\mathcal{A}$ is not guilty of double-spending, and thus this algorithm does nothing.) Further since $\mathcal{M}$ (or the random oracle) chose R_1, R_2, this uniquely fixes $T_1 = g^{u+R_1/(J+t+1)}, T_2 = g^{u+R_2/(J+t+1)}$ as the only valid security tags to accompany serial number S in these two transactions. To deviate from these tags during Spend, $\mathcal{A}$ must fake the proof of validity Φ in step 3 of Spend which we already saw (in balance) happens with only negligible probability.

Thus, Identify can output the public key of the double-spender $pk_{\mathcal{U}}$ along with proof that he double-spent coin S which is simply $\Pi_G = (\pi_1, \pi_2)$. Since running $\mathsf{VerifyGuilt}(params, S, pk_{\mathcal{U}}, \Pi_G)$, for our System One construction, is tantamount to re-running identify and comparing the outputs, we see that VerifyGuilt will always accept on any honest output of Identify.

Anonymity of users: We capture anonymity by describing the simulator S for our construction as described in the formal definition. As presented in System One, we will assume we are using the efficient discrete-logarithm based proof protocols from Sections 3.3 (standard proof protocols) and 3.5 (CL signature protocols).

Let the adversary $\mathcal{A}$, representing a colluding bank and merchant, create and publish a public key $pk_{\mathcal{B}}$. Next, $\mathcal{A}$ may request the public key pk_i of any user i. The adversary can engaged in the Withdraw protocol with any user i as many times as he likes. Finally, $\mathcal{A}$ will be asked to engage in a *legal* number of Spend protocols with some real user j (with a real wallet W_j) or a simulator $\mathcal{S}$ (without W_j or even knowledge of j).

The simulator $\mathcal{S} = \mathcal{S}^{IO\text{-}RO(\mathcal{A})}(params, auxsim, n)$ is given as input the global parameters *params*, some additional information *auxsim* possibly required by other simulators called as subroutines, and the total number of coins in a valid wallet n. This simulator $\mathcal{S}$ is given control of the random oracle as well as input-output access to the adversary $\mathcal{A}$.

Our simulator $\mathcal{S}$ executes each new Spend request by $\mathcal{A}$ as follows:

1. $\mathcal{S}$ receives an optional transaction string $info \in \{0,1\}^*$.
2. $\mathcal{S}$ gathers the appropriate transaction information, such as *info*, $pk_{\mathcal{B}}$, current time, etc., and fixes its output for the random oracle to be an arbitrary value $R \in \mathbb{Z}_q^*$.
3. Next $\mathcal{S}$ chooses random values $u, s, t \in \mathbb{Z}_q$ and a random $J \in [0, \dots, n-1]$, and computes $S = f^{DY}_{g,s}(J+1)$ and $T = g^u f^{DY}_{g,t}(J+1)^R$.
4. $\mathcal{S}$ sends $\mathcal{A}$ the coin (S, π), where $\pi = (R, T, \Phi)$, and Φ is the following simulated signature proof Φ:

(a) $\mathbf{A} = \text{PedCom}(J+1)$ and a (real) proof that $\mathbf{A}$ is a commitment to an integer in the range $[1, \ldots, n]$,

(b) $\mathbf{B} = \text{PedCom}(u)$, $\mathbf{C} = \text{PedCom}(s)$, $\mathbf{D} = \text{PedCom}(t)$ and a *simulated* proof of knowledge of a CL signature from $\mathcal{B}$ on the openings of $\mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ in that order. ($\mathcal{S}$ invokes the appropriate CL simulator [18, 19] for this step, which requires control of the random oracle.)

(c) and a (real) proof Γ that $S = g^{1/(J+s+1)}$ and $T = g^{u+R/(J+t+1)}$.

Observe above that all the difficulty of $\mathcal{S}$'s job is handled by the simulator for a proof of knowledge of a CL signature.

We now explain why the output of $\mathcal{S}$ is computationally indistinguishable from the output of a real user. We claim that during the Withdraw protocol $\mathcal{A}$ did not learn anything meaningful about the set of secrets (u, s, t) that it signed due to the security of the CL signatures. In fact, these values can be information theoretically-hidden from $\mathcal{A}$ by requesting a signature on (u, s, t, r) for a random r that is otherwise discarded [18, 19].

Thus, the values s and t chosen by $\mathcal{S}$ are indistinguishable from those chosen by real users. Recall that a coin consists of the tuple $(S, (R, T, \Phi))$, where R is chosen by $\mathcal{A}$ or the random oracle (thus, indistinguishable between the schemes). Due to the security of the DY PRF [37], the coin parts $S = f^{DY}_{g,s}(J+1)$ and $T = g^u f^{DY}_{g,t}(J+a)^R$ are computationally indistinguishable from random elements in $\widetilde{G}_2$. And therefore, equally likely to have been generated by any user with $pk_i = g^{u_i}$ and with any coin counter value $J \in [0, \ldots, n-1]$. Observe that T looks random due to the fact that $f^{DY}_{g,t}(J+1)$ is computed in a cyclic group of prime order, and is thus a random generator in G raised to an arbitrary, non-zero power. Finally, recall that our simulated Φ consisted of a group of (perfectly hiding) Pederson commitments, two true proofs, and one simulated proof. The simulated proof is the only difference between $\mathcal{S}$'s Φ and that of a real user. However, we ran the CL signature proof simulator to generate the simulated proof, and thus, by the security of the CL signatures, $\mathcal{A}$ distinguishes between Game R (with a real user) and Game I (with $\mathcal{S}$) with only negligible probability based on the Strong RSA [18] or a bilinear map based assumption called LRSW [19].

Removing the random oracle. As in the balance discussion, a (programmable) random oracle will typically be necessary to our proofs here, so that S can fake the coin's *signature* proof of validity. (It needs to be a signature proof, in practice, because the merchant will need to have the bank verify it before deposit.) However, in the special case that the bank and the merchant are a single entity, the random oracle can be securely removed at the cost of make the Spend protocol interactive, changing each proof protocol so that $\mathcal{A}$ must commit to his challenge in advance, and giving the simulator black-box access to $\mathcal{A}$. Further, if random oracles could be removed from the proof of the CL signatures, then that result would carry over to our scheme as well.

Strong Exculpability: In our construction, it is easy to see that corresponding to a wallet W, even one obtained as a result of interacting with a malicious bank $\mathcal{B}$, there are exactly n serial numbers that an honest user $\mathcal{U}$ can produce.

Parts (1a) and (2a) of the strong exculpability definition (and indeed, all of the weak exculpability one) require that an adversary should not be able to produce a serial number S and a proof Π such that VerifyOwnership($params$, $S, \Pi, pk_{\mathcal{U}}, n$) accepts with non-negligible probability. In our System One presentation, the algorithm VerifyOwnership was not defined. Let us now define the algorithm such that it rejects on all inputs. Then System One trivially satisfies weak exculpability and part of the strong exculpability requirement.

Now, we move on to the part of exculpability requirement that concerns VerifyGuilt. This is also fairly trivial. Recall that in System One the proof of guilt is $\Pi = (\pi_1, \pi_2)$ such that the coins (S, π_1) and (S, π_2) were accepted by honest merchants. (That is, the (non-interactive) coins (S, π_1) and (S, π_2) are re-verified as part of the VerifyGuilt algorithm.) And furthermore, part of any valid π_i involves proving knowledge of the user's secret key $sk_{\mathcal{U}} = u$ (see step 3(c) of Spend). Thus, either (1) $\mathcal{A}$ is successful at producing a false Π which means that he is also successful at forging the underlying proof of knowledge, which happens with only negligible probability [52]; or (2) if (S, π_1) and (S, π_2) are both valid coins because they are registered to different users (see discussion at the start of the identification of double-spenders proof) then VerifyGuilt will reject, because the Identify algorithm will not recover $pk_{\mathcal{U}}$. $\square$

A Simpler E-Cash System. Although we settled on System One for presentation purposes, in our original design we considered a simplified version called System Zero. System Zero required that a wallet simply consist of a PRF seed s, the bank's signature on s, and a counter J: $(s, \sigma_{\mathcal{B}}(s), J)$. To spend a coin, the user produces the serial number $f^{DY}_{g,s}(J)$ along with a proof of its validity. This has the benefit that a user would not need to have a public key to join the system. The drawback, however, is that there is no method for recovering the identity of a double-spender. For some applications, this level of security might be appropriate. For example, suppose each coin represents an anonymous vote in a company election. If any shareholder attempts to over-vote, i.e., cast more votes than they are allocated, this fraud will be detected and those votes can be voided.

4.2 System Two: Compact E-Cash with Full Tracing

We now extend System One to allow coin tracing. Let us begin by sketching how this can be accomplished. 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 *her 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 $S_i = f^{DY}_{g,s}(i)$, for $i = 0$ to $2^\ell - 1$, of all coins belonging to $\mathcal{U}$.

(Notice that recovering $sk_{\mathcal{U}}$ would actually allow the bank to decrypt *all* ciphertexts corresponding to different wallets for user $\mathcal{U}$, and thus, trace *all* coins withdrawn from account $pk_{\mathcal{U}}$. We will return to this point later.)

Now that we understand the high-level construction, we will go into more details as to how we realize it. Recall that a user's keys in the previous system were of the form $pk_{\mathcal{U}} = g^a$ and $sk_{\mathcal{U}} = a$, and that the Identify algorithm recovered g^a when a user double-spent. Suppose there was a semantically-secure cryptosystem where the value g^a was sufficient to decrypt and the new public key was a one-way function of g^a. Given such a cryptosystem, we can use the verifiable encryption techniques of Camenisch and Damgård, as described in Section 3.6, which be applied to any semantically-secure encryption scheme. This will allow user $\mathcal{U}$ to leave a verifiable encryption of her wallet secret s under her own $pk_{\mathcal{U}}$ with the bank $\mathcal{B}$ during the withdrawal protocol. No one but $\mathcal{U}$ knows the corresponding $sk_{\mathcal{U}}$. However, by using the same coin structure as System One, double-spending allows $\mathcal{B}$ to recover the value $\widehat{sk_{\mathcal{U}}} = \widehat{g}_1^u$. Thus, $\mathcal{B}$ can decrypt the ciphertexts and trace the cheating user's coins. All we need is a cryptosystem with keypairs of this new form.

Luckily, the bilinear El Gamal scheme (see Section 3.7) does exactly this. It supports public keys of the form $\overline{pk}_{\mathcal{U}} = e(\widehat{g}_1, \widetilde{g}_2)^u$, for $u \in \mathbb{Z}_q$, where knowing $\widehat{sk}_{\mathcal{U}} = \widehat{g}_1^u$ is sufficient for decryption, given a bilinear mapping $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$.

This would be the entire solution, except one technical difficulty arises. The Dodis-Yampolskiy PRF [37], used in the double-spending equation, is only known to be a PRF in groups where DDH is hard. However, to recover the secret key $\widehat{g}_1^u$ for the bilinear El Gamal cryptosystem, we need to set the double-spending equation in the group $\widehat{G}_1$, where there exists a mapping $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$, and thus DDH may be easy. Obviously, this is a problem, because the anonymity of the coins relies on this function being pseudorandom.

We propose two different solutions under two different assumptions to overcome this technical hurdle. Our final result is summarized (later in Theorem 4.4) as:

> 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 **either the XDH (using Solution #1) or the Sum-Free DDH (using Solution #1)** assumptions in the random oracle model.

Let us now describe these two solutions, and then provide a detailed construction using the second one.

Solution #1: Use the DY PRF, make XDH Assumption. As described in Section 2, the XDH Assumption assumes that there are bilinear mappings $e : \widehat{G}_1 \times \widetilde{G}_2 \to \overline{G}$ where DDH is hard in $\widehat{G}_1$. Boneh, Boyen, and Shacham [7] showed that when $\widehat{G}_1$ and $\widetilde{G}_2$ are distinct groups with an efficient isomorphism $\psi : \widetilde{G}_2 \to \widehat{G}_1$, but not $\psi' : \widehat{G}_1 \to \widetilde{G}_2$, then, in the generic group model, the standard Decision Diffie-Hellman (DDH) problem is hard in $\widehat{G}_1$. It has been conjectured that this property may hold when the bilinear groups are instantiated using either the Weil or Tate pairing over MNT curves [47][7, Sec. 8.1]. (For supersingular curves, such a conjecture is known to be false [43].)

Suppose that the above conjecture holds for MNT curves, implying that DDH is hard in $\widehat{G}_1$. Then our construction above just works. We may continue to use the DY PRF [37] to create our coins, as we did in System One. Where we used G, a generic group of prime order where DDH was hard, in System One, we can now replace it with $\overline{G}$, the range of the bilinear mapping, for all items *except* the double-spending equation $\widehat{T}_i = \widehat{g}_1^u f^{DY}_{\widehat{g}_1,t}(i)^R$ which is now set in $\widehat{G}_1$. The user's key pairs are now of the form $\overline{pk}_{\mathcal{U}} = e(\widehat{g}_1, \widetilde{g}_2)^u$ and $\widehat{sk}_{\mathcal{U}} = \widehat{g}_1^u$ for the bilinear El Gamal cryptosystem. (We observe that the bilinear El Gamal cryptosystem works in bilinear groups regardless of the XDH Assumption.)

Due to the obvious similarities with System One, the above comments are sufficient to describe the entire system.

Solution #2: Use a New PRF, make Sum-Free DDH Assumption. Solution #1 has the benefit of being very simple and efficient. However, the XDH Assumption may turn out to be false for MNT curves, or, for other reasons, it may be more desirable to implement System Two using the supersingular curves. For either of these cases, we now propose an alternative design where we do not simultaneously set the double-spending equation in $\widehat{G}_1$ (where DDH is now easy) *and* use the DY PRF (where DDH must be hard).

We overcome this difficulty by using a new PRF that works even in DDH-easy groups. (We'll only use this new PRF for the double-spending equation T and continue to use the DY PRF elsewhere.) We are aware of one candidate PRF for a DDH-easy group due to Dodis [36]. Unfortunately, we could not adapt his construction for use in System Two without performing expensive proofs of knowledge. For efficiency reasons, we instead propose a new PRF construction specifically to solve our problem.

4.2.1 A New PRF for Groups where DDH may be Easy

Let us begin by recalling the definition of a *sum-free* encoding as given by Dodis [36].

Definition 4.2 (Sum-Free) *We say that an encoding $V : \{0,1\}^n \to \{0,1\}^m$ is* sum-free *if for all distinct elements a, b, c, d in the set of encodings $\{V(s)\}_{s\in\{0,1\}^n}$, it holds that $a + b \neq c + d$, where a, b, c, d are viewed as m-bit 0/1-vectors and $+$ is bitwise addition over the integers.*

Dodis [36] proved that if V is any sum-free encoding, and $\langle g\rangle$ is a group of order q, then $f^V_{g,(\cdot)}$, 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 $\vec{t} = (t_0, \ldots, t_{3\ell})$; the function $f^V_{g,\vec{t}}$ is defined as

$$f^V_{g,\vec{t}}(x) = g^{t_0 \prod_{V(x)_i=1} t_i}.$$

This holds under the Sum-Free DDH assumption (also introduced by Dodis [36]); 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.

Lemma 4.3 *The encoding $V(x) = x \circ x^2$ described above is sum-free.*

Proof. Observe that $V : \{0,1\}^\ell \to \{0,1\}^{3\ell}$. Suppose that A, B, C, D are elements of $\{V(s)\}_{s\in\{0,1\}^\ell}$. We can parse each element as $A = a \circ a^2$, $B = b \circ b^2$, etc.

We show that if $A + B = C + D$, then without loss of generality $a = c$ and $b = d$, implying that A, B, C, D are not distinct. If $A + B = C + D$, it follows that

$$\begin{aligned} a + b &= c + d && (1) \\ a^2 + b^2 &= c^2 + d^2 && (2) \end{aligned}$$

For the moment, let $+$ and $-$ be addition and subtraction, respectively, over the integers. We can rearrange equation two as $a^2 - d^2 = c^2 - b^2$. By factoring this new equation, we have $(a+d)(a-d) = (c + b)(c - b)$. We can rearrange equation one as $a - d = c - b$. We can then substitute this into the previous equation to obtain $(a + d)(a - d) = (c + b)(a - d)$. By canceling $(a - d)$ from both sides, we arrive at $a + d = c + b$. Subtracting equation one, we have $d - b = b - d$, which implies $2d = 2b$, and $d = b$ directly follows. The same logic shows that $c = a$.

Lastly, it suffices to observe that this analysis is the same when $+$ and $-$ are bitwise addition and subtraction, respectively. □

4.2.2 System Two with Solution #2

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 $SG(1^k, \zeta) \to (vk_{\mathcal{U}}, ssk_{\mathcal{U}})$ and the bilinear El Gamal key generation algorithm $G(1^k, \zeta) \to (\overline{ek_{\mathcal{U}}}, \widehat{dk_{\mathcal{U}}}) = (e(\widehat{g}_1^u, \widetilde{g}_2), \widehat{g}_1^u)$, and outputs $pk_{\mathcal{U}} = (\overline{ek_{\mathcal{U}}}, vk_{\mathcal{U}})$ and $sk_{\mathcal{U}} = (\widehat{dk_{\mathcal{U}}}, ssk_{\mathcal{U}})$. The bank's keys are as before.

We continue to set the CL signatures, for the bank, in the group $\mathbf{G}$ with composite modulus n, although they could moved to the bilinear groups [19].

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

IDEA:

- Bank obtains a verifiable encryption of s under the user's public key $pk_{\mathcal{U}}$ for the bilinear El Gamal cryptosystem.
- User's wallet of 2^ℓ coins is $(sk_{\mathcal{U}}, s, t_0, \ldots, t_{3\ell}, \sigma, J)$, where σ is the bank's signature on $(sk_{\mathcal{U}}, s, t_0, \ldots, t_{3\ell})$ and J is an ℓ-bit counter.

1. $\mathcal{U}$ identifies himself to the bank $\mathcal{B}$ by proving knowledge of $sk_{\mathcal{U}} = (\widehat{dk_{\mathcal{U}}}, ssk_{\mathcal{U}})$.
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 $\vec{t} = (t_0, \ldots, t_{3\ell})$, where $t_i \in \mathbb{Z}_q$ for all i. As before, this is done as follows: $\mathcal{U}$ chooses random values s' and $\vec{t}$, and sends the commitment $\mathbf{A}' = \mathrm{PedCom}(u, s', \vec{t})$ to $\mathcal{B}$, obtains a random r', sets $s = s' + r'$, and then both $\mathcal{U}$ and $\mathcal{B}$ locally set $\mathbf{A} = \mathbf{g}^{r'}\mathbf{A}' = \mathrm{PedCom}(u, s, \vec{t})$.
3. $\mathcal{U}$ forms a verifiable encryption Q of the value s under his own key $\overline{ek_{\mathcal{U}}} = e(\widehat{g}_1^u, \widetilde{g}_2)$. (This encryption can be proved correct relative to commitment $\mathbf{A}$.) 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 $\mathbf{A}$ 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 $W = (sk_{\mathcal{U}}, s, \vec{t}, \sigma_B(u, s, \vec{t}), J)$, 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.

Recall the details of the serial number of the coin $\overline{S} = f^{DY}_{\overline{g},s}(J) = \overline{g}^{1/(J+s+1)}$, the security tag $\widehat{T} = \widehat{dk_{\mathcal{U}}} f^{V}_{\widehat{g}_1,\vec{t}}(J)^R = \widehat{g}_1^{\,u+Rt_0 \prod_{\{i:V(J)_i=1\}} t_i}$. The signature proof Φ of their validity consists of:

(a) $\mathbf{A}_0 = \mathrm{PedCom}(J)$ and a proof that $\mathbf{A}$ is a commitment to an integer in the range $[0, \ldots, 2^\ell - 1]$; a commitment $\mathbf{A}_1 = \mathrm{PedCom}(J^2)$ and a proof that it is a commitment to the square of the opening of $\mathbf{A}_0$; and finally, a commitment $\mathbf{A}_2 = \mathrm{PedCom}(V(J)) = \mathrm{PedCom}(J \circ J^2)$ and a proof that it was formed correctly.

Outline of Spend Protocol:

1. User computes $R = H(pk_{\mathcal{M}}||info)$, where $info \in \{0,1\}^*$ is provided by the merchant.
2. User sends a coin serial number and double-spending equation:
$$\overline{S} = F^{DY}_{\overline{g},s}(J) \quad , \quad \widehat{T} = \widehat{pk}_{\mathcal{U}} F^{DY}_{\widehat{g}_1,\vec{t}}(J)^R.$$
3. User sends a ZKPOK Φ of $(J, sk_{\mathcal{U}}, s, t_0, \ldots, t_{3\ell}, \sigma)$ such that:
   - $0 \leq J < 2^\ell$ (standard techniques [26, 20, 20, 10])
   - $\overline{S} = F^{DY}_{\overline{g},s}(J)$ (see below)
   - $\widehat{T} = \widehat{pk_{\mathcal{U}}} F^{V}_{\widehat{g}_1,\vec{t}}(J)^R$ (see below)
   - VerifySig$(pk_{\mathcal{B}}, (sk_{\mathcal{U}}, s, t_0, \ldots, t_{3\ell}), \sigma)$ =true (CL signatures [18, 19])
4. If Φ verifies, $\mathcal{M}$ accepts the coin $(\overline{S}, (R, \widehat{T}, \Phi))$ 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.

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

(c) proof that $\mathbf{A}_2$ and $\{\mathbf{B_i}\}$ are consistent; $PK\{(\gamma) : \mathbf{A}_2/\mathbf{g}\prod_{i=1}^{3\ell} \mathbf{B}_i^{2^{i-1}} = \mathbf{h}^\gamma\}$,

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

(e) the following commitments that will help in proving that $\widehat{T}$ was formed correctly: $\mathbf{F}_0 = \mathbf{E}_0$, $\mathbf{F}_i = \mathrm{PedCom}(\prod_{\{j \leq i : V(J)_j = 1\}} t_j)$ for $i = 1$ to 3ℓ,

(f) and a proof that $\overline{S} = f^{DY}_{\overline{g},s}(J)$ and $\widehat{T} = \widehat{g}_1^u(f^V_{\widehat{g}_1,\vec{t}}(J))^R$. (We provide more details of this step in Appendix A.2.) Proving the statement about $\overline{S}$ is done as in System One; proving the statement about $\widehat{T}$ can be done as follows: Prove, for every $1 \leq i \leq 3\ell$ that $\mathbf{F}_i$ was formed correctly, corresponding to the committed value t_i and the value (bit) contained in the commitment $\mathbf{B}_i$. That is to say:
$$\begin{aligned} PK\{(\alpha, \beta, \delta) \ \ : \ \ & \mathbf{F}_i = \mathrm{PedCom}(\alpha) \ \wedge \ \mathbf{F}_{i-1} = \mathrm{PedCom}(\beta) \ \wedge \\ & \mathbf{E}_i = \mathrm{PedCom}(\delta) \ \wedge \ \Big(\big(\mathbf{B}_i = \mathrm{PedCom}(0) \ \wedge \ \alpha = \beta\big) \ \vee \\ & \big(\mathbf{B}_i = \mathrm{PedCom}(1) \ \wedge \ \alpha = \beta\delta\big)\Big)\} \end{aligned}$$
Note that, if all $\mathbf{F}_i$'s are formed correctly, then $\mathbf{F}_{3\ell}$ is a commitment to the discrete logarithm of the value $f^V_{\widehat{g}_1,\vec{t}}(J)$ which is $t_0 \prod_{i=1}^{3\ell} t_i^{(V(J))_i}$. So we can prove the validity of tag $\widehat{T}$ as follows:
$$PK\{(\alpha, \beta) \ : \ \widehat{T} = \widehat{g}_1^{\alpha+\beta R} \wedge \mathbf{C} = \mathrm{PedCom}(\alpha) \wedge \mathbf{F}_{3\ell} = \mathrm{PedCom}(\beta)\}$$

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 $(\overline{S}, R, \widehat{T})$ 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.

Identify$(\zeta, \overline{S}, \pi_1, \pi_2)$**:** The algorithm behaves exactly as before. The proof of guilt Π_G can essentially

IDEA: From two coins with serial number $\overline{S}$, anyone can recover the user's *secret* key $sk_{\mathcal{U}}$.

just be the user's secret key $\widehat{g}_1^u$. If a user does not over-spend her wallets, then her secret key will not be exposed *based on anything in this system.* We acknowledge that an honest user may have her secret key stolen through adversarial actions outside this system (such as a stolen smartcard), and thus, for *policy* reasons the proof of guilt should perhaps also include the transaction logs in L allowing to compute $\widehat{g}_1^u$.

Trace$(\zeta, \overline{S}, pk_{\mathcal{U}}, \Pi_G, D, 2^\ell)$**:** Parse Π_G as $(\widehat{dk}, \pi_1, \pi_2)$ and $pk_{\mathcal{U}}$ as $(\overline{ek}_{\mathcal{U}}, vk_{\mathcal{U}})$. The bank checks that

IDEA: Bank can use $sk_{\mathcal{U}}$ to decrypt and recover s for each wallet registered to $\mathcal{U}$, then compute all serial numbers for all coins in that wallet.

$e(\widehat{dk}, \widetilde{g}_2) = \overline{ek}_{\mathcal{U}}$; 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 $\widehat{dk}$ to recover the value s; and (2) then for $i = 0$ to $2^\ell - 1$, outputs a serial number $\overline{S}_i = f^{DY}_{\overline{g},s}(i)$ and a proof of ownership $\Pi_i = (Q, \sigma, \widehat{dk}, i)$.

VerifyOwnership$(\zeta, \overline{S}, \Pi, pk_{\mathcal{U}}, 2^\ell)$**:** Parse Π as $(Q, \sigma, \widehat{dk}, i)$. Check that σ is $pk_{\mathcal{U}}$'s signature on Q and

IDEA: Anyone can check that the user with $pk_{\mathcal{U}}$ is the owner of a coin with serial number $\overline{S}$.

that i is in the range $[0, \ldots, 2^\ell - 1]$. Next, verify that $\widehat{dk}$ is $pk_{\mathcal{U}}$'s decryption key by checking that $e(\widehat{dk}, \widetilde{g}_2) = \overline{ek}_{\mathcal{U}}$. Finally, run the verifiable decryption algorithm on Q with $\widehat{dk}$ to recover s' and verify that $\overline{S} = f^{DY}_{\overline{g},s'}(i)$. 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.

Details of the proof in step (1f) of the Spend protocol are in Appendix A.2. 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 4.4 *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 either the XDH (using Solution #1) or Sum-Free DDH (using Solution #2) assumptions in the random oracle model.*

Proof sketch.
Balance: *Part 1.* The extractor $\mathcal{E} = \mathcal{E}^{BB\text{-}\varepsilon(\mathcal{A})}_{\mathsf{Withdraw}_m, l_S}$ proceeds exactly as in System One, the only difference being that this time he will recover more messages, e.g., $(u, s, t_0, \ldots, t_{4\ell})$.

Part 2. As before, we know that $\mathcal{A}$ cannot produce a truly valid serial number $\overline{S}$ (i.e., one for which it need not fake a proof of validity) which is not in the set output by the extractor A_f except with negligible probability. So it remains to analyze $\mathcal{A}$ success in faking proofs of validity for $\overline{S} \notin A_f$. The analysis is similar to before, except that the signature proof $\pi = (R, \widehat{T}, \Phi)$ is more complicated. If $\mathcal{A}$ succeeds in convincing $\mathcal{B}$ to accept $(\overline{S}, \pi)$, then he must have concocted a *false* proof as part of Φ that: (1) $\mathbf{A}$ opens to an integer in $[1, \ldots, 2^\ell]$ or (2) some $\mathbf{B}_i$ opens to either 0 or 1 or (3) $\mathbf{A}$ and $\{\mathbf{B}_i\}$ are consistent or (4) $\mathcal{A}$ knows a signature from $\mathcal{B}$ on the opening of $\mathbf{C}, \mathbf{D}, \{\mathbf{E}_i\}$ or (5) that $\overline{S}$ (and $\widehat{T}$) are well formed as in Γ. Case (1) happens with negligible probability $\nu_1(k)$ under the Strong RSA assumption [26, 20, 20, 10]. Cases (2), (3), and (5) happen with negligible probability $\nu_2(k)$ under the discrete logarithm assumption [52]. Case (4) happens with negligible probability $\nu_3(k)$ under the assumption that the CL signatures are secure (which requires either the Strong RSA assumption [18] or an additional bilinear map assumption called LRSW [19]). Thus, $\mathcal{A}$'s total success probability is also negligible.

Identification of double-spenders: The identification of double-spenders property of System Two follows directly from the proof of this property for System One.

Tracing of double-spenders: First, the adversary has only negligible chance of winning by the existence of a serial number $\overline{S}' \in A_f$ not output by Trace. This is because, in each Withdraw protocol, the secret s that $\mathcal{E}$ extracts (to generate $\overline{S}_i = f^{DY}_{\overline{g},s}(i)$, for $1 \leq i \leq n$, for A_f) is also verifiably encrypted under the user's public key $\overline{ek_{\mathcal{U}}} = e(\widehat{g}_1, \widetilde{g}_2)^u$ and given to the bank. Due to the semantic-security of the bilinear El Gamal scheme [2] and the soundness of the verifiable encryption protocol of Camenisch and Damgård [16], we know that the bank can decrypt to obtain s once part of the user's secret key $\widehat{g}_1^u$ is known. For System Two, it is easy to see that $\widehat{g}_1^u$ can be recovered from double-spent coins given the identification of double-spenders property above. (Recall that in System Two, one computes $\widehat{g}_1^u$ on the way to finding $ek_{\mathcal{U}}$.) Thus, with high probability, the Trace algorithm will obtain s, for each wallet, and output all serial numbers in A_f.

Secondly, the adversary has only a negligible chance of winning by the existence of a serial number $\overline{S}_j$ and corresponding proof Π_j output by Trace$(params, \overline{S}, pk_{\mathcal{U}}, \Pi_G, D, n)$ such that the algorithm VerifyOwnership$(params, \overline{S}_j, \Pi_j, pk_{\mathcal{U}}, n)$ does not accept. This is because verifying the ownership of a coin $\overline{S}_j$ in our construction is tantamount to re-running the (deterministic) Trace algorithm and checking its output.

Anonymity of users: The proof of anonymity for System Two follows that of System One, with the three exceptions being that:

1. One must argue that $\mathcal{A}$ does not learn any information about the users during the verifiable encryption protocol added to $\mathsf{Withdraw}$ that will later help it distinguish. This follows from the semantic-security of the bilinear El Gamal scheme [2] and the verifiable encryption technique of Camenisch and Damgård [16] which rely on the DBDH (which is later subsumed by the Sum-Free DDH assumption in the theorem statement) and the Strong RSA assumptions.
2. The coin part $\widehat{T}$ is now indistinguishable from a random element in $\widehat{G}_1$ due to the pseudorandomness of our new PRF $f^V_{(\widehat{g}_1,\cdot)}(\cdot)$ secure under the Sum-Free DDH assumption.
3. And finally, the simulated signature proof Φ is more complicated, it remains that the only simulated part of Φ is performed for $\mathcal{S}$ (with IO, random oracle access) by the CL signature proof simulator under the Strong RSA [18] and the LRSW [19] assumptions.

Strong Exculpability: It is easy to see that corresponding to each wallet in our construction, there are exactly n valid serial numbers.

First, we show parts (1b), (2b) and (3) of the strong exculpability requirement, namely, that no adversary can produce a serial number $\overline{S}$ and a proof Π such that $\mathsf{VerifyGuilt}(params, \overline{S}, pk_{\mathcal{U}}, \Pi)$ accepts with non-negligible probability, and yet it is not the case that the coin with serial number $\overline{S}$ was spent more than once by user $\mathcal{U}$.The same reasoning for proving eculpability of System One applies to this part of the proof of exculpability for System Two – that is, either (1) the adversary is successful in faking a proof of knowledge of u or (2) two users have wallet secrets s and s' that are 2^ℓ-close. Both cases occur with negligible probability.

Next, we show part (1a) that no adversary can produce a serial number $\overline{S}$ that does not belong to the set of serial numbers of the $\mathcal{U}$ known to $\mathcal{A}$, and a proof Π such that $\mathsf{VerifyOwnership}(params,$ $\overline{S}$, Π, $pk_{\mathcal{U}}$, $n)$ accepts with non-negligible probability. Recall that a proof for $\mathsf{VerifyOwnership}$ in System Two consists of $(Q, \sigma, \widehat{dk}, i)$, where Q is a verifiable encryption of a wallet secret s, σ is the user's signature on Q, $\widehat{dk}$ is the user's decryption key, and i is an integer in $[1, \ldots, 2^\ell]$. There is only one $\widehat{dk}$ corresponding to $\overline{ek_{\mathcal{U}}} \in pk_{\mathcal{U}}$ (which is easy to test as $e(\widehat{dk}, \widetilde{g}_2) = \overline{ek_{\mathcal{U}}}$). Decryption of Q, to say s, given $\widehat{dk}$ is deterministic, as is checking that $\overline{S} = f^{DY}_{e(\widehat{g}_1,\widetilde{g}_2),s}(i)$ for i in the right range. Thus, to forge a proof of ownership against $pk_{\mathcal{U}}$ for a serial number $\overline{S}$, the adversary needs to produce a seed s and number J such that $\overline{S} = F^{DY}_{e(\widehat{g}_1,\widetilde{g}_2),s}(J+1)$, and a ciphertext Q that is an encryption of s signed by $\mathcal{U}$'s signing key. It is easy to see that the ability to do so violates either the security of the signature scheme used for signing Q or the security of the PRF used to generate $\overline{S}$, or the security of the encryption scheme used to generate the ciphertext Q.

Finally, let us show part (2a) that, if no double-spending has occured, it is impossible to prove that the user double-spent a coin whose serial number is known to the adversary. This also easily follows from the security of the PRF used to generate $\overline{S}$, and the security of the encryption scheme used to generate the ciphertext Q: the only way that $\mathsf{VerifyOwnership}$ can accept the serial number $\overline{S}$ is if it is given the seed s and a number J such that $\overline{S} = F^{DY}_{e(\widehat{g}_1,\widetilde{g}_2),s}(J+1)$, which contradicts the security of the encryption and of the PRF.

$\square$

Acknowledgments. We are grateful to Yevgeniy Dodis for helpful comments. We also thank Giuseppe Ateniese, Breno de Medeiros, and Hovav Shacham for bringing the XDH assumption to our attention.

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.

Susan Hohenberger's work was performed while at IBM Research, Zurich Research Laboratory, CH-8803, Rüschlikon.

Anna Lysyanskaya was supported by NSF Career Grant CNS-0347661.

References

[1] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, April 2000.

[2] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *the 12th Annual Network and Distributed System Security Symposium*, pages 29–43, 2005.

[3] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-Resistant Storage. Johns Hopkins University, Computer Science Department Technical Report # TR-SP-BGMM-050705. `http://spar.isi.jhu.edu/~mgreen/correlation.pdf`, 2005.

[4] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 480–494. Springer Verlag, 1997.

[5] Mihir Bellare and Adriana Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In Moti Yung, editor, *Advances in Cryptology — CRYPTO '02*, volume 2442 of *LNCS*, pages 162–177. Springer Verlag, 2002.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT '04*, volume 3027 of *LNCS*, pages 54–73. Springer, 2004.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures using strong Diffie-Hellman. In *Advances in Cryptology – CRYPTO '04*, volume 3152 of LNCS, pages 41–55, 2004.

[8] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer Verlag, 2001.

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT '01*, volume 2248 of LNCS, pages 514–532, 2001.

[10] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT '00*, volume 1807 of *LNCS*, pages 431–444. Springer Verlag, 2000.

[11] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, April 1993.

[12] Stefan Brands. Untraceable off-line cash in wallets with observers. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *LNCS*, pages 302–318, 1993.

[13] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 318–333. Springer Verlag, 1997.

[14] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[15] Ernie Brickell, Peter Gemmel, and David Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proceedings of the Sixth Annual ACM-SIAMs*, pages 457–466. Association for Computing Machinery, January 1995.

[16] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT '00*, volume 1976 of *LNCS*, pages 331–345. Springer Verlag, 2000.

[17] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks '04*, volume 3352 of LNCS, pages 120–133, 2004.

[18] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks '02*, volume 2576 of *LNCS*, pages 268–289. Springer Verlag, 2002.

[19] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 56–72. Springer Verlag, 2004.

[20] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 107–122. Springer Verlag, 1999.

[21] Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 413–430. Springer Verlag, 1999.

[22] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO '03*, volume 2729 of *LNCS*, pages 126–144, 2003.

[23] Jan Camenisch and M. Stadler. Efficient group signature schemes for large groups. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1296 of LNCS, pages 410–424, 1997.

[24] Jan L. Camenisch, Jean-Marc Piveteau, and Markus A. Stadler. Blind signatures based on the discrete logaritm problem. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *LNCS*, pages 428–432. Springer Verlag Berlin, 1994.

[25] Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.

[26] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come – easy go divisible cash. In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *LNCS*, pages 561–575. Springer Verlag, 1998.

[27] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology — CRYPTO '82*, pages 199–203. Plenum Press, 1982.

[28] David Chaum. Blind signature systems. In David Chaum, editor, *Advances in Cryptology — CRYPTO '83*, page 153. Plenum Press, 1983.

[29] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[30] David Chaum. Online cash checks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *LNCS*, pages 289–3293. Springer Verlag, 1989.

[31] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *LNCS*, pages 319–327. Springer Verlag, 1988.

[32] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *LNCS*, pages 390–407. Springer-Verlag, 1992.

[33] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.

[34] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *LNCS*, pages 174–187. Springer Verlag, 1994.

[35] Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT '02*, volume 2501 of *LNCS*, pages 125–142. Springer, 2002.

[36] Yevgeniy Dodis. Efficient construction of (distributed) verifiable random functions. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *LNCS*, pages 1–17. Springer Verlag, 2002.

[37] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography*, volume 3386 of LNCS, pages 416–431, 2005.

[38] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86*, volume 263 of LNCS, pages 186–194, 1986.

[39] Yair Frankel, Yiannis Tsiounis, and Moti Yung. "Indirect discourse proofs:" Achieving efficient fair off-line e-cash. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *LNCS*, pages 286–300. Springer Verlag, 1996.

[40] Matthew Franklin and Moti Yung. Towards provably secure efficient electronic cash. Technical Report TR CUSC-018-92, Columbia University, Dept. of Computer Science, April 1992. Also in: Proceedings of ICALP 93, Lund, Sweden, July 1993, volume 700 of LNCS, Springer Verlag.

[41] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 16–30. Springer Verlag, 1997.

[42] Steven D. Galbraith. Supersingular curves in cryptography. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT '01*, volume 2248 of LNCS, pages 495–513, 2001.

[43] Steven D. Galbraith and Victor Rotger. Easy Decision-Diffie-Hellman Groups. *LMS Journal of Computation and Mathematics*, 7:201–218, 2004.

[44] Stanislaw Jarecki and Vitaly Shmatikov. Hancuffing big brother: an abuse-resilient transaction escrow scheme. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT '04*, volume 3027 of *LNCS*, pages 590–608. Springer, 2004.

[45] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of LNCS, pages 571–589. Springer, 2004.

[46] Noel McCullagh and Paulo S. L. M. Barreto. A new two-party identity-based authenticated key agreement. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA*, volume 3376 of LNCS, pages 262–274, 2004.

[47] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–1243, 2001.

[48] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51, Number 2:231–262, 2004.

[49] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *LNCS*, pages 481–496. Springer-Verlag, 1989.

[50] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer Verlag, 1991.

[51] Michael O. Rabin and Jeffery O. Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39:239–256, 1986.

[52] Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[53] Mike 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.

[54] Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair blind signatures. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *LNCS*, pages 209–219. Springer Verlag, 1995.

[55] Yiannis S. Tsiounis. *Efficient Electonic Cash: New Notions and Techniques.* PhD thesis, Northeastern University, Boston, Massachusetts, 1997.

A Full Protocols to Spend Coins

To provide the full protocols to spend a coin, we need to provide some details about the signature scheme the bank uses.

Let QR_n denote the set of quadratic residues modulo n. Let $\mathbf{Z}, \mathbf{U}, \mathbf{V}$ be elements of QR_n that are part of the public key of the bank.

Let ℓ_n denote the number of bits of the bank's RSA modulus n and let $\ell_\varnothing$, e.g., $\ell_\varnothing$ be a security parameter controlling the statistically zero knowledge property of the proof protocol as well as the statistically hiding property of the commitment schemes we use.

A signature of the bank on the seed (message) s consists of the values $(\mathbf{Q}, e, v)$, where $e \in \{2^{\ell_e} - 2^{\ell'_e}, 2^{\ell_e} + 2^{\ell'_e}\}$ is a random prime, $v \in \{0,1\}^{\ell_n+\ell_\varnothing}$is a random integer, and $\mathbf{Q} \in \langle \mathbf{U} \rangle \subset QR_n$, such that the following holds

$$\mathbf{Z} \equiv \mathbf{Q}^e \mathbf{V}^v \mathbf{U}^s \pmod{n} \ .$$

We note that the bank does not learn s when issuing this signature. Instead, the bank and the user run an efficient two-party protocol where the output of the user will be $(\mathbf{Q}, e, v)$ [18].

In case the bank signs a block of messages, say (u, s, t), at once, we replace $\mathbf{V}$ by $(\mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3})$ in the public. The signature still consists of values $(\mathbf{Q}, e, v)$ such that

$$\mathbf{Z} \equiv \mathbf{Q}^e {\mathbf{V_1}}^u {\mathbf{V_2}}^s {\mathbf{V_3}}^t \mathbf{U}^s \pmod{n} \ .$$

To make our proof more efficient by roughly a factor of two, we will change the range of J, the coin counter. Our goal is to have a range that contains an odd number of elements. This is not strictly necessary but gives us a more efficient proof when J lies in an odd interval. Thus, we need that $J_0 - J_1 \le J \le J_0 + J_1$ holds for some J_0 and J_1. For instance, setting $J_1 = 2^{\ell-1}$ and $J_0 = 2^\ell$ we have $2^{\ell-1} \le J \le 3 \cdot 2^{\ell-1}$ and thus can spend $2^\ell + 1$ coins. Note that now J can no longer be 0 and a therefore the serial number S can be computed as $S = g^{1/(J+s)}$.

A.1 System One: Full Protocol

We now describe how the user spends a coin in System One in more detail. Recall that the user has obtained from the bank a signature $(\mathbf{Q}, e, v)$ on the "messages" u, s, and t.

Let J be the number of the coin. By construction we have $J_0 - J_1 \leq J \leq J_0 + J_1$ and hence $0 \leq J_1^2 - (J - J_0)^2 = J_1^2 - J_0^2 + 2J_0J - J^2$ for all J. Thus, to show that a coin number J lies in the required interval, we only need to show that for $J_1^2 - (J - J_0)^2$ is a positive number. Now, due to Lagrange, each positive number can be represented by four squares.

We are now ready to describe how a user spends one of her coins. we merged all the single proofs of Φ into one and get some efficiency improvements, e.g., the commitments $\mathbf{C}$ and $\mathbf{D}$ are not needed. Also, we choose J differently and thus compute S and T in a slightly different manner.

1. $\mathcal{M}$ (optionally) sends a string *info* $\in \{0,1\}^*$ containing transaction information to $\mathcal{U}$ and/or identifies himself by proving knowledge of $sk_{\mathcal{M}}$ (We leave this step out.).
2. $\mathcal{M}$ chooses a random $R \in \mathbb{Z}_q$ such that $R \neq 0$ and sends R to $\mathcal{U}$ (or it is a result of a hash function).
3. If $J > J_0 + J_1$, the user aborts as she has already sent all coins.
4. The user computes the serial number $S = g^{1/(J+s)}$ and a (now fixed) security tag $T = pk_{\mathcal{U}} g^{R/(J+t)}$
5. The user computes values w_1, w_2, w_3, and w_4 such that $\sum_i w_i = J_1^2 - (J - J_0)^2$ (e.g., using an algorithm by Rabin and Shallit [51]).
6. The user chooses random values r_A,r_B, r_1, r_2, r_3, and r_4 $r \in_R \{0,1\}^{\ell_n + \ell_\varnothing}$, and computes the the commitments $\mathbf{A} = \mathbf{g}^J\mathbf{h}^{r_A}$, $\mathbf{B} = \mathbf{g}^u\mathbf{h}^{r_B}$ and $\mathbf{W_i} = \mathbf{g}^{w_i}\mathbf{h}^{r_i}$ for $i = 1, \ldots, 4$, The user chooses $r \in_R \{0,1\}^{\ell_n + \ell_\varnothing}$ and computes $\mathbf{Q}' := \mathbf{Q}\mathbf{U}^r$. Note that $(\mathbf{Q}', e, v + r)$ is also a valid signature on the message s but that $\mathbf{Q}$ and $\mathbf{Q}'$ are statistically independent.
7. The user computes the following signature proof of knowledge:

$$\begin{aligned}\Psi = SPK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \rho_1, \ldots, \rho_7) : \\ &\mathbf{Z} = \pm\mathbf{Q}'^{\varepsilon}\mathbf{V_1}^{\mu}\mathbf{V_2}^{\sigma}\mathbf{V_3}^{\tau}\mathbf{U}^{\zeta} \ \wedge \ \mathbf{A} = \pm\mathbf{g}^{\gamma}\mathbf{h}^{\rho_A} \ \wedge \ g = S^{\sigma}S^{\gamma} \ \wedge \\ &\mathbf{W_i} = \pm\mathbf{g}^{\nu_i}\mathbf{h}^{\rho_i} \ \wedge \ \mathbf{g}^{J_1^2 - J_0^2} = \pm(\mathbf{A}\mathbf{g}^{-2J_0})^{\gamma}\mathbf{W_1}^{\nu_1}\mathbf{W_2}^{\nu_2}\mathbf{W_3}^{\nu_3}\mathbf{W_4}^{\nu_4}\mathbf{h}^{\rho_5} \ \wedge \\ &\mathbf{B} = \mathbf{g}^{\mu}\mathbf{h}^{\rho_6} \ \wedge \ 1 = \mathbf{B}^{\sigma}\mathbf{B}^{\tau}(1/\mathbf{g})^{\alpha}\mathbf{h}^{\rho_7} \ \wedge \ g^R = T^{\sigma}T^{\tau}(g^R)^{\alpha} \ \wedge \\ &\sigma \in \{0,1\}^{\ell_m + \ell_\varnothing + \ell_{\mathcal{H}} + 2} \ \wedge \\ (\varepsilon - 2^{\ell_e}) \in \{0,1\}^{\ell'_e + \ell_\varnothing + \ell_{\mathcal{H}} + 1}\}&(\mathbf{Z}, \mathbf{V_1}, \mathbf{V_2}, \mathbf{V_3}, \mathbf{U}, \mathbf{g}, \mathbf{h}, n, g, S, T, \mathbf{A}, \mathbf{B}, \mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{W_4}, \mathit{info})\end{aligned}$$

8. If Φ verifies, $\mathcal{M}$ accepts the coin (R, S, T, Φ) and subsequently submits (R, S, T, Φ) to the bank $\mathcal{B}$. If R is not already in L, then $\mathcal{B}$ accepts the coin and stores the values (R, S, T) in a database of spent coins L. If S was already in L, then $\mathcal{B}$ initiates the Identify protocol.
9. $\mathcal{U}$ updates his counter $J = J + 1$ as before. When $J > 2^{\ell} - 1$, the wallet is empty.

We see that, to spend a coin, a user needs to compute 7 multi-base exponentiations to build the commitments and 11 multi-base exponentiations to carry out the proof. The merchant and bank need to do 11 multi-base exponentiations to spend a coin.

A.2 System Two: Partial Protocol

The Spend protocol in System Two is more involved than that of System One. From the description in Section 4.2, it may not be clear how to implement step (1f) of the Spend protocol. We now describe how to construct these signature proofs in more detail (although further optimizations are also possible).

Recall that in step (1f) a user $\mathcal{U}$ must prove to a merchant $\mathcal{M}$ that the serial number $\overline{S}$ and the security tag $\widehat{T}$ are *valid* elements of a coin. In particular, $\mathcal{U}$ is tasked with showing that $\overline{S} = f^{DY}_{\overline{g},s}(J)$ and $\widehat{T} = \widehat{g}_1^u(f^V_{\widehat{g}_1,\vec{t}}(J))^R$. The essence of this proof it to show that $\overline{S}$ and $\widehat{T}$ are using the same J value.

More formally, we denote $\Gamma = \Gamma_0 \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_{3\ell}$ as the signature proof of knowledge where Γ_0 is the following signature proof of knowledge:

$$\begin{aligned}\Gamma_0 = SPK\{(\alpha,\beta,\gamma,\delta_1,\delta_2,\delta_3) : \mathbf{g} = (\mathbf{AD})^{\alpha}\mathbf{h}^{\delta_1} \wedge \overline{S} = \overline{g}^{\alpha} \wedge \mathbf{C} = \mathbf{g}^{\beta}\mathbf{h}^{\delta_2} \wedge \mathbf{F_{3\ell}} = \mathbf{g}^{\gamma}\mathbf{h}^{\delta_3} \wedge \\ \widehat{T} = \widehat{g}_1^{\beta}(\widehat{g}_1^{R})^{\gamma}\}(\overline{S},\widehat{T},\mathbf{A},\{\mathbf{B_i}\},\mathbf{C},\mathbf{D},\{\mathbf{E_i}\},\{\mathbf{F_i}\},\mathbf{g},\mathbf{h},n,\widehat{g}_1,\widetilde{g}_2,\overline{g},pk_{\mathcal{M}},R,\mathit{info})\end{aligned}$$

After this point in the proof we can consider the values $(u, s, \vec{t}, J)$ as fixed (although unknown to $\mathcal{M}$) and value R as public, what remains to be shown is that $\mathbf{F_{3\ell}} = \mathbf{g}^{\gamma}\mathbf{h}^{\delta_3}$ where $f^V_{\widehat{g}_1,\vec{t}} = \widehat{g}_1^{\gamma}$ and the $\mathcal{U}$ knows γ, δ_3. To show this, we successively build up to $\mathbf{F_{3\ell}}$ by showing that for each intermediate commitment $\mathbf{F_i}$, for $i = 1$ to 3ℓ, either: (1) the ith bit of $V(J)$ is zero and thus $\mathbf{F_i}$ and $\mathbf{F_{i-1}}$ open to the same value, or (2) the ith bit of $V(J)$ is one and thus $\mathbf{F_i}$ opens to the opening of $\mathbf{F_i}$ times t_i (where t_i is the corresponding secret in $\vec{t}$). The user already proved that each $\mathbf{B_i}$ commits to the ith bit of $V(J)$ in steps (1b) and (1c).

Then for $i = 1$ to 3ℓ, we have the signature proof of knowledge Γ_i:

$$\begin{aligned}\Gamma_i = SPK\{(\alpha_i,\beta_i,\phi_i,\{\gamma_i^{(n)}\}) : \Big(\mathbf{F_i} = \mathbf{F_{i-1}}\mathbf{h}^{\gamma^{(2)}} \wedge \mathbf{B_i} = \mathbf{h}^{\gamma_i^{(3)}}\Big) \vee \\ \Big(\mathbf{F_i} = \mathbf{F_{i-1}}^{\alpha_i}\mathbf{h}^{\gamma_i^{(4)}} \wedge \mathbf{E_i} = \mathbf{g}^{\alpha_i}\mathbf{h}^{\gamma_i^{(5)}} \wedge \mathbf{B_i} = \mathbf{g}\mathbf{h}^{\gamma_i^{(6)}}\Big)\} \\ (\overline{S},\widehat{T},\mathbf{A},\{\mathbf{B_i}\},\mathbf{C},\mathbf{D},\{\mathbf{E_i}\},\{\mathbf{F_i}\},\mathbf{g},\mathbf{h},n,\widehat{g}_1,\widetilde{g}_2,\overline{g},pk_{\mathcal{M}},R,\mathit{info}).\end{aligned}$$