# Compact E-Cash and Simulatable VRFs Revisited

Mira Belenkiy[1], Melissa Chase[2], Markulf Kohlweiss[3], and Anna Lysyanskaya[4]

[1] Microsoft, mibelenk@microsoft.com
[2] Microsoft Research, melissac@microsoft.com
[3] KU Leuven, ESAT-COSIC / IBBT, markulf.kohlweiss@esat.kuleuven.be
[4] Brown University, anna@cs.brown.edu

**Abstract.** Efficient non-interactive zero-knowledge proofs are a powerful tool for solving many cryptographic problems. We apply the recent Groth-Sahai (GS) proof system for pairing product equations (Eurocrypt 2008) to two related cryptographic problems: compact e-cash (Eurocrypt 2005) and simulatable verifiable random functions (CRYPTO 2007). We present the first efficient compact e-cash scheme that does not rely on a random oracle. To this end we construct efficient GS proofs for signature possession, pseudo randomness and set membership. The GS proofs for pseudorandom functions give rise to a much cleaner and substantially faster construction of simulatable verifiable random functions (sVRF) under a weaker number theoretic assumption. We obtain the first efficient fully simulatable sVRF with a polynomial sized output domain (in the security parameter).

## 1 Introduction

Since their invention [BFM88] non-interactive zero-knowledge proofs played an important role in obtaining feasibility results for many interesting cryptographic primitives [BG90,GO92,Sah99], such as the first chosen ciphertext secure public key encryption scheme [BFM88,RS92,DDN91]. The inefficiency of these constructions often motivated independent practical instantiations that were arguably conceptually less elegant, but much more efficient ([CS98] for chosen ciphertext security).

We revisit two important cryptographic results of pairing-based cryptography, compact e-cash [CHL05] and simulatable verifiable random functions [CL07], that have very elegant constructions based on non-interactive zero-knowledge proof systems, but less elegant practical instantiations. Our results combine the best of both worlds, a clean design and an efficient implementation.

*Compact e-cash.* Electronic cash (e-cash) was introduced by David Chaum [Cha83] as an electronic analogue of physical money and has been a subject of ongoing cryptographic research since then [CFN90,FY92,CP93,Bra93a,CPS94,Bra93b,SPC95,FTY96,Tsi97]. The participants in an e-cash system are users who withdraw and spend e-cash; a bank that creates e-cash and accepts it for deposit, and merchants who offer goods and services in exchange for e-cash, and then deposit the e-cash to the bank. The main security requirements are (1) anonymity: even if the bank and the merchant and all the remaining users collude with each other, they still cannot distinguish Alice's purchases from Bob's; (2) unforgeability: even if all the users and all the merchants collude against the bank, they still cannot deposit more money than they withdrew.

Unfortunately, it is easy to see that, as described above, e-cash is useless. The problem is that here money is represented by data, and it is possible to copy data. Unforgeability will guarantee that the bank will only honor at most one of copy of a given coin for deposit and will reject the others. Anonymity will guarantee that there is no recourse against such a cheating Alice. So one of the merchants will be cheated. There are two known remedies against this double-spending behavior. The first remedy is on-line e-cash [Cha83], where the bank is asked to vet a coin before the spend protocol can terminate successfully. The second remedy is off-line e-cash, introduced by Chaum, Fiat and Naor [CFN90]. The additional requirement of an offline e-cash system is (informally) that no coin can be double-spent without revealing the identity of the perpetrator.

A further development in the literature on e-cash was compact e-cash [CHL05]. In compact e-cash, the user withdraws $N$ coins in a withdrawal protocol whose complexity is $O(\log N)$ rather than $O(N)$. Similarly, the resulting wallet requires storage size $(\log N)$ rather than $)(N)$. The main idea is as follows:

Available from the IACR Cryptology ePrint Archive as Report 2009/107.

in the withdrawal protocol, a user obtains the Bank's signature on $(x, s, t)$, where $s$ and $t$ are random seeds of a pseudorandom function (PRF) $F_{(\cdot)}(\cdot)$ and $x$ is the user's identifier. In the spend protocol, a serial number of the $i$th coin is computed as $S = F_s(i)$, and a double spending equation is computed as $T = x + RF_t(i)$, where $R$ is a random challenge by the merchant. The coin itself consists of $(S, T, R, \pi)$, where $\pi$ is a non-interactive zero-knowledge proof of knowledge of the following values: $x$, $s$, $t$, $i$, $\sigma$ where $\sigma$ is the Bank's signature on $(x, s, t)$, $1 \leq i \leq N$, $S = F_s(i)$ and $T = x + RF_t(i) \bmod q$. If $g$ is a generator of a group $G$ of order $q$, and $G$ is the range of the PRF $F_{(\cdot)}(\cdot)$, then the double-spending equation can instead be computed as $T = g^x F_t(i)^R$. It is easy to see that two double-spending equations for the same $t$, $i$ but different $R$'s allow us to compute $g^x$. It was shown that this approach yields a compact e-cash scheme [CHL05]. Later, this was extended to so-called e-tokens [CHK$^+$06] that allow up to $k$ anonymous transactions per time period (for example, this would correspond to subscriptions to interactive game sites or anonymous sensor reports).

Thus, we see that compact e-cash and variants such as e-tokens can be obtained from a signature scheme, a pseudorandom function, and a non-interactive zero-knowledge (NIZK) proof system for the appropriate language. However, until now no efficient instantiations of the NIZK proofs could be given, and all practical instantiations of compact e-cash had to derive the non-interactive proofs from interactive proofs via the Fiat-Shamir heuristic [FS87] which is known not to yield provably secure constructions [GK03]. It seemed that, perhaps, random oracle based techniques were necessary to achieve such schemes efficiently. We show here that this is not the case.

*Challenges and Techniques.* Until the recent proof system of Groth and Sahai [GS07], there were no efficient NIZK proof systems for languages most heavily used in cryptographic constructions (such as languages of true statements about discrete logarithm representations and bilinear pairings). However, constructing an efficient provably-secure compact e-cash scheme is not simply a matter of replacing the Fiat-Shamir based NIZK proofs with the Groth-Sahai system. There are several issues that arise when we attempt to apply the Groth-Sahai proofs. First, recall that the Groth-Sahai system only works for proofs of particular types of statements. Thus, we must find a PRF and a signature scheme where verification can be phrased in terms of such statements. In the case of the PRF, we use a modification of the Dodis-Yampolskiy VRF [DY05], which outputs elements of bilinear group $G_1$. We show that this is secure under the assumption that DDHI holds in this group.

However, simply combining a commitment scheme and PRF with the NIZK proof system is not sufficient. We must also ensure that (1) the key space of the PRF is a subset of the domain over which the commitment is binding and (2) it is possible to tell if a commitment commits to a value in the given key space.

To see that this is nontrivial, consider the implementation of the Groth-Sahai proof system based on the Subgroup Decision Assumption in composite order groups. Here the commitment is proven to be binding over one of the prime order subgroups. However, if we limit the key space to this subgroup, there is no way to tell whether a given commitment commits to a valid value in this key space. In fact, if $g_p, g_q$ are generators for the two subgroups, then a commitment to $g_p^a$ is distributed identically to a commitment to $g_p^a g_q$. On the other hand, if we use the PRF described above, then these two values will correspond to very different PRFs.

We show that the other two constructions given by Groth and Sahai do not suffer from this problem, and thus can be used in our application.

For the signature scheme, we note that verification of Boneh-Boyen signatures [BB04b] can be phrased as a pairing product equation. However, as noted in Belenkiy et al. [BCKL08], because Groth-Sahai proofs are only partially extractable, we need a stronger unforgeability. Here we need that it be impossible to produce $F(m), \mathsf{Sign}_{sk}(m)$ for an unsigned message $m$, where $F(m)$ is a value that can be extracted from a commitment to $m$. Belenkiy et al. gave a construction which satisfies this definition, but only allows signatures on a single message. We need the bank to be able to sign multiple message blocks, thus we extend that construction to construct a multi-block P-signature scheme. We also show that issu-

ing can be done efficiently using more recent techniques given in [BCC+08]. (The original [BCKL08] construction relied on general two party computation for arithmetic circuits.)

We also need to be able to prove that the coin value falls within a given range. The original Camenisch et al. construction uses a technique by [Bou00], which relies on the fact that the underlying RSA group has unknown order. Groth-Sahai proofs, on the other hand, rely on the cryptographic bilinear group model, and it is not known how to construct such groups with unknown order. Thus, we must use a different technique for our range proofs. We follow the basic concept of [TS06,CCS08], and implement the range proofs using the new P-signatures mentioned above.

Finally, while Groth and Sahai present a NIZK proof system for a large class of statements, their simpler witness indistinguishable proof system is much more efficient. Thus, we specifically design our protocols to use NIZK proofs only when necessary. As a result, we obtain a construction that is almost competitive in efficiency with the original Camenisch et al. construction.

*E-cash construction.* Our construction is in the common parameters model and relies on several number-theoretic assumptions. Our first building block is a signature scheme and an unconditionally binding commitment scheme that allows for an efficient proof of knowledge of a signature on a set of committed values, as well as for an efficient protocol for getting a committed value signed. This is done by extending the P-signature construction of Belenkiy et al. [BCKL08], which only allows to sign single values, and incorporating the techniques from [BCC+08]. In our construction we will also use P-signatures, together with the techniques of [CCS08] (that relied on interactive proofs) to obtain efficient non-interactive interval proofs.

Our second building block is a pseudorandom function and an unconditionally binding commitment scheme $\mathsf{Com}(.,.)$ (the same as for the P-signature scheme) with an efficient proof system for the serial number $S$ and the double spending tag $T$.

*Simulatable verifiable random functions.* Our main observation is that the NIZK proof for a compact e-cash serial number, a proof of the language $L_F = \{S, C_y, C_s \mid \exists s, y, r_s, r_y$ such that $S = F_s(y), C_y = \mathsf{Com}(y, r_y), C_s = \mathsf{Com}(s, r_s)\}$ is a special case of a simulatable verifiable random function (sVRF), introduced by Chase and Lysyanskaya [CL07]. Chase and Lysyanskaya gave an efficient construction of a multi-theorem non-interactive zero-knowledge proof system for any language $L$ from a single-theorem one for the same language (while other single-theorem to multi-theorem transformations required the Cook-Levin reduction [Coo71] to an NP-complete language first).

Chase and Lysyanskaya [CL07] gave two constructions for sVRFs. The first is based on generic non-interactive zero-knowledge proofs and is therefore impracical. The second construction is based on composite order bilinear pairings [BGN05,FST06], and has several shortcomings. In particular, its range is either only logarithmic in the security parameter or it is only weakly simulatable. Our fully simulatable construction is thus more efficient by a factor of the security parameter; it is also designed in a way that is more modular and therefore easier to understand (and improve). Finally, it relies on a somewhat weaker assumption. Therefore, we believe this result will be of independent interest.

*Our contribution and outline of the paper.* We present the first P-signature scheme for multiple messages, the first fully simulatable VRF with polynomial sized output domain, and the first efficient compact e-cash scheme that does not rely on random oracles. (The security of conventional e-cash was, e.g., studied in [JLO97,STS99,Tro05].) The rest of the paper is organized as follows. In Section 2 we discuss our assumptions and recall useful results about non-interactive zero-knowledge. In Section 3 we define and construct our new P-signature scheme for message blocks. Section 4 and Section 5 revisit simulatable verifiable random functions and compact e-cash respectively.

## 2 Preliminaries

In this section we list our assumptions and recall some useful results about non-interactive zero-knowledge proofs (NIZK).

A function $\nu$ is *negligible* if, for every integer $c$, there exists an integer $K$ such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *infeasible)* if there exists no probabilistic polynomial time (p.p.t.) algorithm to solve it.

*Bilinear Pairings.* Let $G_1, G_2$, and $G_T$ be groups of prime order $p$. The map $e : G_1 \times G_2 \to G_T$ must satisfy the following properties: (a) *Bilinearity*: a map $e : G_1 \times G_2 \to G_T$ is bilinear if $e(a^x, b^y) = e(a, b)^{xy}$; (b) *Non-degeneracy*: for all generators $g \in G_1$ and $h \in G_2$, $e(g, h)$ generates $G_T$; (c) *Efficiency*: There exists a p.p.t. algorithm $\mathsf{BMGen}(1^k)$ that outputs $(p, G_1, G_2, G_T, e, g, h)$ to generate the bilinear map and an efficient algorithm to compute $e(a, b)$ for any $a \in G_1$, $b \in G_2$.

*Assumptions.* The security of our scheme is based on previously proposed number-theoretic assumptions. The unforgeability of our P-signature construction relies on the TDH [BCKL08] and the HSDH [BW07] assumptions; pseudo-randomness is based on the $q$-DDHI assumption [BB04a,CHL05]; and the zero-knowledge of the Groth-Sahai proof system rests on the XDH or DLIN assumption [GS07].

**Definition 1 (Triple DH).** *On input $g, g^x, g^y \in G_1$, $h, h^x \in G_2$, and $\{c_i, g^{1/(x+c_i)}\}_{i=1\ldots q}$ for random $x, y$, and $c_1, \ldots, c_q$, it is computationally infeasible to output a tuple $(h^{\mu x}, g^{\mu y}, g^{\mu xy})$ for $\mu \neq 0$.*

**Definition 2 (Hidden SDH).** *On input $g, g^x, u \in G_1$, $h, h^x \in G_2$ and $\{g^{1/(x+c_\ell)}, h^{c_\ell}, u^{c_\ell}\}_{\ell=1\ldots q}$ for random $x$, and $c_1, \ldots c_q$, it is computationally infeasible to output a new tuple $(g^{1/(x+c)}, h^c, u^c)$.*

**Definition 3 ($q$-DDHI).** *On input $g, g^\alpha, g^{\alpha^2}, \ldots g^{\alpha^q} \in G$ for a random $\alpha \leftarrow Z_p$, it is computationally infeasible to distinguish $g^{\frac{1}{\alpha}}$ from a random element of $G$ with probability non-negligibly better than $1/2$.*

Our sVRF requires that the $q$-DDHI assumption holds either in $G_1$ or $G_2$. Without loss of generality we fix this group to be $G_1$. Note that this is slightly stronger than the assumption used in [DY05] to construct an efficient VRF (there the challenge is $e(g, h)^{\frac{1}{\alpha}}$ or a random element of $G_T$). However, it is still weaker than the BDHBI assumption used in the sVRF construction in [CL07]. For further discussion and a summary of the XDH and DLIN assumptions see Appendix A.

*Composable Non-Interactive Proofs.* We review composable non-interactive proof systems. Let $R(\cdot, \cdot)$ be any polynomial-time computable relation. A non-interactive proof system for an NP language allows a prover to convince a verifier of the truth of the statement $\exists x : R(y, x)$ about instance $y$ using witness $x$. Non-interactive proof systems use a common reference string $params$ as output by $\mathsf{Setup}(1^k)$ that is common input to both the $\pi \leftarrow \mathsf{Prove}(params, y, x)$ and $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(params, x, \pi)$ algorithms. This notion can be generalized for a relation $R(params, y, x)$ parameterized by $params$.

Informally, zero-knowledge captures the notion that a verifier learns nothing from the proof but the truth of the statement. Witness-indistinguishability is a weaker notion that guarantees that the verifier learns nothing about which witness was used in the proof.

In a *composable* (under the definition of Groth and Sahai [GS07]) non-interactive *witness indistinguishable* proof system there exists an algorithm $\mathsf{SimSetup}$ that outputs $params$ together with a trapdoor $sim$, such that (1) $params$ output by $\mathsf{SimSetup}$ are indistinguishable from those output by $\mathsf{Setup}$; (2) the output of $\mathsf{Prove}$ using these parameters is perfectly witness-indistinguishable (in other words, even if there are two witnesses to a statement, they induce identical distributions on the proofs). Composable non-interactive *zero-knowledge* further means that there exists an algorithm $\mathsf{SimProve}$ that outputs a simulated proof using $sim$ and the output of $\mathsf{SimProve}$ is distributed identically to that of $\mathsf{Prove}$ when given the simulated parameters. The big advantage of a composable definition is that it is fairly simple and easy to work with, and yet it still implies the standard multi-theorem definitions.

*Composable proofs about commitments.* The prover and verifier frequently get some set of commitments $(C_1, \ldots, C_n)$ as common input. The prover wants to show that a statement about instance $y = (C_1, \ldots, C_n, \mathsf{Condition})$ holds. The witness to the statement is $(x_1, open_1, \ldots, x_n, open_n, z)$, where $(x_i, open_i)$ is the opening of commitment $C_i$, while $z$ is some value that has nothing to do with

the commitments. The relation is $R = \{(params, y, x) | C_1 = \mathsf{Com}(params, x_1, open_1) \wedge \ldots \wedge C_n = \mathsf{Com}(params, x_n, open_n)$
$\wedge\ \mathsf{Condition}(params, x_1, \ldots, x_n, z)\}$.

*Summary of Groth-Sahai proofs.* Groth and Sahai [GS07] give a composable witness-indistinguishable proof system that lets us efficiently prove statements in the context of groups with bilinear maps. Let $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$ be the setup for pairing groups of prime order $p$.

In a Groth-Sahai proof, the prover and the verifier both know values $\{a_q\}_{q=1\ldots Q} \in G_1$, $\{b_q\}_{q=1\ldots Q} \in G_2$, $t \in G_T$, and $\{\alpha_{q,m}\}_{q=1\ldots Q, m=1\ldots M}, \{\beta_{q,n}\}_{q=1\ldots Q, n=1\ldots N} \in Z_p$. In addition, they both know commitments $\{C_m\}_{m=1\ldots M}$ and $\{D_n\}_{n=1\ldots N}$ to values in $G_1$ and $G_2$ respectively. For each commitment $C_m$ and $D_n$ the prover knows the opening information and the committed value $x_m \in G_1$ or $y_n \in G_2$ respectively ($m = 1\ldots M$, $n = 1\ldots N$).

Groth-Sahai proofs prove that the values in these commitments fulfill the pairing product equation $\prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t$.

*Groth-Sahai commitments.* Throughout the paper we will use Groth-Sahai commitments ($\mathsf{GSCom}$) in our constructions. Under the parameters output by $\mathsf{Setup}$ they are perfectly binding. We will sometimes make use of the fact that they are also extractable.

## 3 A Multi-block P-Signature Scheme

Belenkiy et al. [BCKL08] intruduced signatures with efficient non-interactive proofs of signature possession. Their construction can only be used to sign a single message block. In this section, we briefly review the definition of a P-signature scheme and construct a multi-block P-signature scheme.

Before defining and constructing P-signatures, we recall some particulars about the way Belenkiy et al. use Groth Sahai proofs. In addition to the zero-knowledge or witness indistinguishability property they rely on the fact that they are partially extractable ($f$-extractable [BCKL08]) proofs of knowledge about committed values. By '$x$ in $C$' we denote that there exists $open$ such that $C = \mathsf{Com}(x, open)$. Following Camenisch and Stadler [CS97a] and Belenkiy et al. [BCKL08], we use the following notation to express an $f$-extractable NIPK for instance $y = (C_1, \ldots, C_n, \mathsf{Condition})$ with witness $w = (x_1, open_1, \ldots, x_n, open_n, z)$:

$$\pi \leftarrow \mathsf{NIPK}[x_1 \text{ in } C_1, \ldots, x_n \text{ in } C_n]\{(\ f(params,\ (x_1, open_1, \ldots, x_n, open_n, y)\ )\ ) : \mathsf{Condition}(params, x_1, \ldots, x_n, z)\}.$$

For such a proof there exists a polynomial-time extractor $(\mathsf{ExtractSetup}, \mathsf{Extract})$. $\mathsf{ExtractSetup}(1^k)$ outputs $(td, params)$ where $params$ is distributed identically to the output of $\mathsf{Setup}(1^k)$. For all p.p.t. adversaries $\mathcal{A}$, the probability that $\mathcal{A}(1^k, params)$ outputs $(y, \pi)$ such that $\mathsf{Verify}(params, y, \pi) = \mathsf{accept}$ and $\mathsf{Extract}(td, y, \pi)$ fails to extract $f(params, (x_1, open_1, \ldots, x_n, open_n, z))$, such that $x_i$ is the content of the commitment $C_i$, and $\mathsf{Condition}(params, x_1, \ldots, x_n, z)$ is satisfied is negligible in $k$.

Groth-Sahai proofs use commitments $\mathsf{GSCom}(x, open)$ that allow to extract the value $x$ but not the opening $open$. In short, Groth-Sahai proofs are $f$-extractable proofs of the following form

$$\mathsf{NIPK}[\{x_m \text{ in } C_m\}_{m=1}^{M}, \{y_n \text{ in } D_n\}_{n=1}^{N}]\{(x_1, ..., x_M, y_1, ..., y_N) : \prod_{q=1}^{Q} e(a_q \prod_{m=1}^{M} x_m^{\alpha_{q,m}}, b_q \prod_{n=1}^{N} y_n^{\beta_{q,n}}) = t\}.$$

For our P-signature scheme we will commit to a message $m \in Z_p$ as $\mathsf{Com}(m, (open_1, open_2)) = (\mathsf{GSCom}(h^m, open_1), \mathsf{GSCom}(u^m, open_2))$. Such a commitment allows to extract $F(m) = (h^m, u^m)$.

### 3.1 Definition of Multi-block P-Signatures

A signature scheme consists of four algorithms: $\mathsf{Setup}$, $\mathsf{Keygen}$, $\mathsf{Sign}$, and $\mathsf{VerifySig}$. $\mathsf{Setup}(1^k)$ generates the public parameters $params$. $\mathsf{Keygen}(params)$ generates a signing key pair $(pk, sk)$. $\mathsf{Sign}(params,$

$sk, m$) computes a signature $\sigma$ on $m$. $\mathsf{VerifySig}(params, pk, m, \sigma)$ outputs accept if $\sigma$ is a valid signature on $m$, reject otherwise. We extend this definition to support multi-block messages $\boldsymbol{m} = (m_1, \ldots m_n)$.

**Definition 4 ($F$-Secure Signature Scheme [BCKL08]).** *Let $F$ be an efficiently computable bijection. With not necessarily efficient inverse $F^{-1}$. We say that a signature scheme is $F$-secure (against adaptive chosen message attacks) if it has the following properties: (a)* Correctness: VerifySig *always accepts a signature $\sigma$ obtained using the* Sign *algorithm; (b) $F$-*Unforgeability: *no adversary should be able to output values $(F_1, \ldots, F_n, \sigma)$ such that for $\boldsymbol{m} = (F^{-1}(F_1), \ldots, F^{-1}(F_n))$ algorithm* $\mathsf{VerifySig}(params, pk, \boldsymbol{m}, \sigma) = \mathsf{accept}$ *unless he has previously obtained a signature on $\boldsymbol{m}$.*

**Definition 5 (P-Signature Scheme [BCKL08]).** *A P-Signature scheme combines an $F$-secure signature scheme with a commitment scheme and three protocols:*

1. *An algorithm* $\mathsf{SigProve}(params, pk, \sigma, \boldsymbol{m} = (m_1, \ldots, m_n))$ *that generates commitments $(C_1, \ldots, C_n)$ and a NIZK proof $\pi \leftarrow \mathsf{NIPK}[m_1 \text{ in } C_1, \ldots, m_n \text{ in } C_n]\{(F(m_1), \ldots F(m_n), \sigma) : \mathsf{VerifySig}(params, pk, \boldsymbol{m}, \sigma) = \mathsf{accept}\}$, and the corresponding* $\mathsf{VerifyProof}(params, pk, \pi, (C_1, \ldots, C_n))$ *algorithm.*
2. *A composable non-interactive zero-knowledge proof system for proving equality of committed values, i.e., a proof of relation $R = \{(params, (x, y), (open_x, open_y)) \mid C = \mathsf{Com}(params, x, open_x) \wedge D = \mathsf{Com}(params, y, open_y) \wedge x = y\}$.*
3. *A secure two party computation [JS07] that lets a signer issue a signature on a committed message vector $\boldsymbol{m}$ without learning any information about $\boldsymbol{m}$. The protocol consists of interactive algorithms* $\mathsf{SigIssue}(params, sk, C_1, \ldots C_n)$ *and* $\mathsf{SigObtain}(params, pk, \boldsymbol{m}, open_1, \ldots, open_n)$.

### 3.2 Construction of a Multi-Block P-Signature Scheme

We first construct an $F$-secure multi-block signature scheme.

$\mathsf{Setup}(1^k)$. Let $(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BMGen}(1^k)$ be the parameters of a bilinear map, let $u$ be an additional generator for $G_1$, and let $params_{GS}$ be the parameters for the corresponding Groth-Sahai NIZK proof system (either in the XDH or the DLIN setup). Output parameters $params = ((q, G_1, G_2, G_T, g, h), u, params_{GS}, z = e(g, h))$.

$\mathsf{Keygen}(params)$ picks random $\alpha, \beta_1, \ldots, \beta_n \leftarrow Z_p$. The signer calculates $v = h^\alpha$, $\tilde{v} = g^\alpha$, $w_i = h^{\beta_i}$, $\tilde{w_i} = g^{\beta_i}$, $1 \le i \le n$. The secret-key is $sk = (\alpha, \boldsymbol{\beta})$. The public-key is $pk = (v, \boldsymbol{w}, \tilde{v}, \tilde{\boldsymbol{w}})$. The public key can be verified by checking that $e(g, v) = e(\tilde{v}, h)$ and $e(g, w_i) = e(\tilde{w_i}, h)$ for all $i$.

$\mathsf{Sign}(params, (\alpha, \beta), \boldsymbol{m})$ chooses a random $r \leftarrow Z_p \setminus \{-(\alpha + \beta_1 m_1 + \cdots + \beta_n m_n)\}$ and calculates $\sigma_1 = g^{1/(\alpha + r + \beta_1 m_1 + \cdots + \beta_n m_n)}$, $\sigma_2 = h^r$, $\sigma_3 = u^r$. The signature is $(\sigma_1, \sigma_2, \sigma_3)$.

$\mathsf{VerifySig}(params, (v, \boldsymbol{w}, \tilde{v}, \tilde{\boldsymbol{w}}), \boldsymbol{m}, (\sigma_1, \sigma_2, \sigma_3))$ outputs accept if $e(\sigma_1, v\sigma_2 \prod_{i=1}^n w_i^{m_i}) = z$ and $e(u, \sigma_2) = e(\sigma_3, h)$.

**Theorem 1.** *Let $F(m) = (h^m, u^m)$. The above signature scheme is $F$-secure given the HSDH and TDH assumptions.* See Appendix B for the proof.

We need to augment the multi-block signature scheme with the three P-Signature protocols.

1. $\mathsf{SigProve}(params, (v, \boldsymbol{w}, \tilde{v}, \tilde{\boldsymbol{w}}), (\sigma_1, \sigma_2, \sigma_3), \boldsymbol{m})$ is defined as follows: We use Com to commit to the $m_i$ as follows: $\mathsf{Com}(m_i, (open_{i,1}, open_{i,2})) = (\mathsf{GSCom}(h^{m_i}, open_{i,1}), \mathsf{GSCom}(u^{m_i}, open_{i,2})) = (H_i, U_i) = C_i$; then we form the Groth-Sahai proof:

$$\pi \leftarrow \mathsf{NIZK}[h^{m_1} \text{ in } H_i, u^{m_1} \text{ in } U_1, \ldots, h^{m_n} \text{ in } H_n, u^{m_n} \text{ in } U_n]\{$$
$$(h^{m_1}, u^{m_1}, w_1^{m_1}, \ldots, h^{m_n}, u^{m_n}, w_n^{m_n}, \sigma_1, \sigma_2, \sigma_3) : e(\sigma_1, v\sigma_2 \textstyle\prod_{i=1}^n w_i^{m_i}) = z \wedge$$
$$e(u, \sigma_2) e(\sigma_3, h^{-1}) = 1 \wedge \{e(\tilde{w_i}, h^{m_i}) e(g^{-1}, w_i^{m_i}) = 1 \wedge e(u, h^{m_i}) e(u^{m_i}, h^{-1}) = 1\}_{i=1}^n\}$$

$\mathsf{VerifyProof}(params, pk, \pi, (C_1, \ldots, C_n))$ simply verifies the proof $\pi$.
To see that the witness indistinguishable proof $\pi$ is also zero-knowledge, the simulation setup sets $u = g^a$. The simulator can then pick $s, m_1, \ldots m_n \leftarrow Z_p$ and compute $\sigma_1 = g^{1/s}$. We implicitly set $r = s - (\alpha + \sum_{i=1}^{n} m_i \beta_i)$. Note that the simulator does not know $r$ and $\alpha$. However, he can compute $h^r = h^s / (v \prod_{i=1}^{n} w_i^{m_i})$ and $u^r = u^s / (\tilde{v} \prod_{i=1}^{n} \tilde{w_i}^{m_i})^a$. Now he can use $h^{m_1}$, $u^{m_1}$, $w_1^{m_1}$,..., $h^{m_n}$, $u^{m_n}$, $w_n^{m_n}$, $\sigma_1$, $\sigma_2 = h^r$, $\sigma_3 = u^r$ as a witness and construct the proof $\pi$ in the same way as the real Prove protocol. By the witness indistinguishability, a proof using the faked witnesses is indistinguishable from a proof using a real witness. See also [BCKL08].

2. The second protocol is a proof of equality of committed values. It is of the form $\mathsf{NIPK}[x \text{ in } C; y \text{ in } D]$ $\{(x, y, h^\theta) : e(x/y, h^\theta) = 1 \wedge e(g, h^\theta) = e(g, h)\}$.
Groth and Sahai [GS07] show that such witness-indistinguishable proofs are also zero-knowledge. A simulator that knows the simulation trapdoor $sim$ for the GS proof system can simulate the two conditions by setting $\theta$ to 0 and 1 respectively. In this way he can fake the proofs for arbitrary commitments.
3. The third protocol is a secure two-party computation for signing a committed value. One could use the same technique as in Belenkiy et al. [BCKL08] to reduce computing a signature to computing an arithmetic circuit using the Jarecki and Shmatikov [JS07] secure two-party computation protocol. Alternatively, we suggest the use of a more efficient protocol based on homomorphic encryption as for example done in [BCC+08,CKW04].

**Theorem 2.** *The above construction is a secure P-Signature scheme given the HSDH and TDH assumption, either the SXDH or DLIN assumption, and the security of the two-party computation protocol.*

The proof follows from the $F$-unforgeability of the multi-block signature scheme and the security of the Groth-Sahai proofs, which depend on either the SXDH or DLIN assumptions. The zero-knowledge simulations are done as sketched above. For details we refer to [GS07,BCKL08,BCC+08].

## 4 Strongly Simulatable Verifiable Random Functions

Here we present our new construction for sVRFs. Later, we will show that an extension of this construction (as described in sections 4.2 and 4.3) can be used to construct provably secure e-cash.

At a high level, a sVRF is an extension of a pseudorandom function (PRF) (and also of a slightly weaker extension, called a VRF [MRV99]). It includes a key generation procedure that generates a seed for the PRF along with a corresponding public key. It also includes a proof system for proving that a particular output is correct with respect to a given input and a given public key. We require fairly strong hiding properties from this proof system – in particular, we do not want it to interfere with the pseudorandomness properties of the PRF. For the full definition, see Appendix C or [CL07].

### 4.1 A New sVRF Construction

Our construction will be in the bilinear group setting where $(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BMGen}(1^k)$. We will use the function $F_s(x) = g^{\frac{1}{s+x}}$ to build an efficient Simulatable VRF.[5] Note that the base function is similar to the Dodis-Yampolskiy VRF [DY05], which uses the function $F_s(x) = e(g, h)^{\frac{1}{s+x}}$ and thus gives output in $G_T$. Moving our function to output elements in $G_1$ is the crucial step which allows us to use the Groth-Sahai proof techniques.

**Theorem 3.** *Let $D_k \subset Z$ denote a family of domains of size polynomial in $k$. Let $p, g, e, G_1, G_2, G_T$ be as described above where $|p| = k$. If the DDHI assumption holds in $G_1$, then the set $\{g^{\frac{1}{s+x}}\}_{x \in D_k}$ is indistinguishable from the set $\{g^{r_x}\}_{x \in D_k}$ where $s, \{r_x\}_{x \in D_k}$ are chosen at random from $Z_p$.* The proof is very similar to that in [DY05].

[5] This function is also known as a Weak Boneh-Boyen signature [BB04b].

We will build an sVRF based on this function as follows:

$\mathsf{Setup}(1^k)$. Let $(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BMGen}(1^k)$ be the parameters of a bilinear map and let $params_{GS}$ be the parameters for the corresponding Groth-Sahai NIZK proof system (either in the XDH or the DLIN setup). Output parameters $params_{VRF} = ((p, G_1, G_2, G_T, g, h), params_{GS})$.

$\mathsf{Keygen}(params_{VRF})$. Pick a random seed $s \leftarrow Z_p$ and random opening information $open_s$, and output secret key $sk = (s, open_s)$ and public key $pk = \mathsf{GSCom}(h^s, open_s)$.

$\mathsf{Eval}(params_{VRF}, sk = (s, open_s), x)$. Compute $y = g^{1/(s+x)}$.

$\mathsf{Prove}(params_{VRF}, sk = (s, open_s), x)$. Compute $y = g^{1/(s+x)}$ and $C_y = \mathsf{GSCom}(y, open_y)$ from random opening $open_y$. Next create the following two proofs: $\pi_1$, a composable NIZK proof that $C_y$ is a commitment to $y$; this is proof that the value $v$ committed to in $C_y$ fulfills the pairing product equation $e(v/y, h^\theta) = 1 \wedge e(g, h^\theta) = e(g, h)$ (see Appendix G and [GS07] for details); $\pi_2$, a GS composable witness indistinguishable proof that $C_y$ is a commitment to $Y$ and $pk$ is a commitment to $S$ such that $e(Y, Sh^x) = e(g, h)$. Output $\pi = (C, \pi_1, \pi_2)$.

$\mathsf{Verify}(params, pk, x, y, \pi = (C, \pi_1, \pi_2))$. Use the Groth-Sahai verification to Verify $\pi_1, \pi_2$ with respect to $C, x, pk, y$.

**Theorem 4.** *This construction with domain size $p$ is a strong sVRF under the q-DDHI for $G_1$ and under the assumption that the Groth-Sahai proof system is secure.* For proof, see Appendix C.1.

### 4.2 A NIZK Protocol for Pseudo-random Functions

In some applications, we need something stronger than an sVRF. In our e-cash application, we need to be certain that the proofs will reveal no information about which wallet was used, which means that they should completely hide the seed used. Furthermore, we do not want to reveal which coin in the wallet is being spent, thus we also want to hide the input $x$.

Thus, we will build a composable NIZK proof for the following language:

$$\mathcal{L}_S = \{C_s, C_x, y | \exists x, s, open_x, open_s \text{ such that } C_s = \mathsf{Com}(s, open_s) \wedge C_x = \mathsf{Com}(x, open_x) \wedge y = F_s(x)\}$$

Note that there are four points where an sVRF proof is weaker than a full NIZK proof. First, the sVRF public key is not guaranteed to hide the secret key, only to hide enough information to preserve the pseudorandomness of the output values. However, this is not a problem in the above construction, since our public key is formed as a commitment. Second, an sVRF has a fixed public key, while we want to be able to compute unlinkable proofs for many different values of the PRF. This again is not relevant in the above construction: since we form our public key using a commitment scheme, we can easily use a different value in each proof. Third, in the sVRF proof, the input $x$ is given in the clear. We can fix this fairly easily by replacing $x$ by a commitment and proof. The final difference is that the sVRF proof need not be fully zero knowledge - the sVRF simulator is given the secret key as input (in our construction, the opening of the commitment $C_s$). We resolve this last point by adding extra commitments $C'_s, C'_x$ (whose opening the zero-knowledge simulator will know), and zero-knowledge proofs that they commit to the same values as $C_s, C_x$.

On input $(C_s, C_x, y)$ and $(x, s, open_x, open_s)$ a NIZK proof of membership in $\mathcal{L}_S$ is done as follows: We first compute commitment $C'_s$ to $h^s$. Then we compute $C_y, \pi_1$ as in the sVRF Prove protocol, with $pk = C'_s$. Next we compute a commitment $C'_x$ to $h^x$, and a GS composable witness-indistinguishable proof $\pi_2$ that $C_y$ is a commitment to $Y$, $C'_x$ is a commitment to $X$, and $C'_s$ is a commitment to $S$ such that $e(Y, SX) = e(g, h)$. Finally, to make the construction zero-knowledge, we add composable NIZK proofs $\pi_s$ and $\pi_x$ that $C_s$ and $C'_s$, and $C_x$ and $C'_x$ are commitments to the same values. Let $v$ be $s$ or $x$, respectively. Then each proof is a proof that the values $v$ and $v'$ committed to in $C_v$ and $C_v$ fulfill the pairing product equation $e(v/v', h^\theta) = 1 \wedge e(g, h^\theta) = e(g, h)$. See [GS07] for why this is zero-knowledge. The final proof is $\pi = (C'_s, C'_x, C'_y, \pi_1, \pi_2, \pi_s, \pi_x)$.

The proof is verified using the Groth-Sahai verification techniques to check $\pi_1, \pi_2, \pi_3, \pi_4$ with respect to $C_s, C_x, y, C'_s, C'_x, C'_y$.

**Theorem 5.** *The above proof system is a secure composable zero knowledge proof system for the language $\mathcal{L}_S(params)$, where $params$ is output by* Setup. The proof appears in Appendix C.2.

### 4.3 NIZK Proofs Doublespending Equations: A More Complex Language

In our application, we use NIZKs about PRFs in two different places. The first is to prove that a given serial number has been computed correctly as $F_s(x)$ according to a committed seed $s$ and committed input $x$. That can be done using the NIZK protocol described in the previous section. However, we also need to be able to prove that the doublespending value $T$ has been computed correctly. Thus, we also need a proof system for the following language:

$$\mathcal{L}_T = \{C_s, C_x, C_{sk}, tag, ch \mid \exists x, s, sk, open_x, open_s, open_{sk} \text{ such that}$$
$$C_s = \mathsf{Com}(s, open_s) \wedge C_x = \mathsf{Com}(x, open_x) \wedge C_{sk} = \mathsf{Com}(sk, open_{sk}) \wedge tag = (g^{sk})^{ch} F_s(x)\}$$

We can generalize our above proof system to handle this as well. For the construction see Appendix D.

### 4.4 Efficiency comparison with previous sVRF construction

As described above, our sVRF proof requires 1 commitment in $G_1$, 1 Groth-Sahai proof, and one zero-knowledge proof of equality of values in $G_1$. Thus, if we instantiate the proofs under the SXDH assumption, our construction requires 14 elements of $G_1$ and 14 elements of $G_2$ to give a proof, and the sVRF outputs a random element of the group $G_1$. Note that the group size is exponential in the security parameter $k$, so this really produces $k$ bits of pseudorandomness.

We compare this to the previous contruction of sVRFs given by Chase and Lysyanskaya [CL07]. That construction was based on composite order bilinear groups. For the order of such groups to resist factorization they must be of a much greater size to achieve the same security as prime order groups. We assume a conservative factor of 5 for this difference [6]. As pairing operations (and exponentiation) have cubic complexity, it is fair to assume that composite order pairings are at least two orders of magnitude slower than prime order pairings.

In addition, the basic construction of [CL07] is only weakly simulatable: for each input value there was a certain restricted set of outputs for which the simulator could output a simulated proof. Finally, the simulator also required some trapdoor information about the desired output value (in the construction it was a discrete logarithm). In order to obtain full simulatability, in which the simulator could produce a simulated proof for any output value in the range of the function with no additional information, this result applied an extractor to the output of the weak sVRF to extract a single bit. The simulator could then sample values from the simulatable range together with some trapdoor information, until it had found one on which the extractor produced the appropriate bit. Clearly extending this approach to achieve more than $O(\log k)$ bits of randomness would be infeasible.

Each proof generated by this construction requires 3 elements of the composite order group $G$. Thus, in order to produce $k$ bits of randomness, even if we assume that we extended the construction to extract $\log k$ bits, we would need $k/\log k$ proofs, for a total of $3 * k/\log k$ elements of $G$.

## 5 New Compact E-Cash Scheme

We construct a compact e-cash scheme using our multi-block P-signatures and sVRF protocols. Compact e-cash as defined by Camenisch et al. [CHL05] lets a user withdraw multiple e-coins simultaneously. There are three types of players: a bank $\mathcal{B}$ as well as many users $\mathcal{U}$ and merchants $\mathcal{M}$ (though merchants

[6] http://www.keylength.com/en/3/

are treated as a special type of user). Please refer to [CHL05] or Appendix E for protocol specifications and a definition of security.[7] We now show how to construct compact e-cash.

$\mathsf{CashSetup}(1^k)$. The setup runs $\mathsf{SigSetup}(1^k)$ and returns the P-signature parameters $params$. Our construction is non-blackbox: we reuse the GS NIPK proof system parameters $params_{GS}$ that are contained in $params$. The parameters $params_{GS}$ in turn contain the setup for a bilinear pairing $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$ for a paring $e : G_1 \times G_2 \to G_T$ for groups of prime order $p$.

$\mathsf{BankKG}(params, n)$. The bank creates two P-signature key pairs, $(pk_w, sk_w) \leftarrow \mathsf{SigKeygen}(params)$ for issuing wallets and $(pk_c, sk_c) \leftarrow \mathsf{SigKeygen}(params)$ for signing coin indices. Then the bank computes a P-signature on the $n$ coin indices $\Sigma_1, \ldots, \Sigma_n$, where $\Sigma_i = \mathsf{SigSign}(sk_c, i)$. The bank's secret-key is $sk_{\mathcal{B}} = (sk_w, sk_c)$ and the bank's public-key is $(pk_w, pk_c, \Sigma_1, \ldots, \Sigma_n)$.

$\mathsf{UserKG}(params)$. The user picks $sk_{\mathcal{U}} \leftarrow Z_p^*$ and returns $(pk_{\mathcal{U}} = e(g,h)^{sk_{\mathcal{U}}}, sk_{\mathcal{U}})$.

Merchants generate their keys in the same way but also have a publicly known identifier $id_{\mathcal{M}} = f(pk_{\mathcal{M}})$ associated with their public keys ($f$ is some publicly known mapping).

$\mathsf{Withdraw}(\mathcal{U}(params, pk_{\mathcal{B}}, sk_{\mathcal{U}}, n), \mathcal{B}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}}, n))$. The user obtains a wallet from the bank.

1. The user picks $s', t' \leftarrow Z_p$; computes commitments $comm_{sk} = \mathsf{Com}(sk_{\mathcal{U}}, open_{sk_{\mathcal{U}}})$, $comm_{s'} = \mathsf{Com}(s', open_{s'})$, and $comm_{t'} = \mathsf{Com}(t', open_{t'})$; and sends $comm_{sk}$, $comm_{s'}$, and $comm_{t'}$ to the bank. The user proves in zero-knowledge that he knows the opening to these values, and that $comm_{sk}$ corresponds to the secret key used for computing $pk_{\mathcal{U}}$.[8]
2. If the proofs verify, the bank sends the user random values $s'', t'' \in Z_p$.
3. The user picks random $open_s, open_t$, commits to $comm_s = \mathsf{Com}(s' + s'', open_s)$, and $comm_t = \mathsf{Com}(t' + t'', open_t)$, sends $comm_s$ and $comm_t$ to the bank, and proves that they are formed correctly. Let $s = s' + s''$ and $t = t' + t''$.
4. The user and bank run $\mathsf{SigObtain}(params, pk_w, (sk_{\mathcal{U}}, s, t), (open_{sk}, open_s, open_t)) \leftrightarrow \mathsf{SigIssue}($ $params, sk_w, (comm_{sk}, comm_s, comm_t))$ respectively. The user obtains a P-signature $\sigma$ on $(sk_{\mathcal{U}}, s, t)$. The user stores the wallet $W = (s, t, pk_{\mathcal{B}}, \sigma, n)$; the bank stores tracing information $T_W = pk_{\mathcal{U}}$.

$\mathsf{SpendCoin}(params, (s, t, pk_{\mathcal{B}}, \sigma, J), pk_{\mathcal{M}}, info)$. The user calculates a serial number $S = F_s(J) = g^{1/(s+J)}$. The user needs to prove that he knows a signature $\sigma$ on $(sk_{\mathcal{U}}, s, t)$ and a signature $\Sigma_J$ on $J$ such that $S = F_s(J)$. Next the user constructs a double-spending equation $T = (g^{id_{\mathcal{M}} \| info})^{sk_{\mathcal{U}}} F_t(J)$.[9] The user proves that $T$ is correctly formed for the $sk_{\mathcal{U}}, t, J$, signed in $\sigma$ and $\Sigma_J$.

All these proofs need to be done non-interactively. We now give more details. The user runs SigProve, first on $\sigma$ and $pk_w$ to obtain commitments and proof $((C_{id}, C_s, C_t), \pi_1) \leftarrow \mathsf{SigProve}(params, pk_w, \sigma, (sk_{\mathcal{U}}, s, t))$ for $sk_{\mathcal{U}}, s, t$ respectively and second on $\Sigma_J$ and $pk_c$ to obtain commitment and proof $(C_J, \pi_2) \leftarrow \mathsf{SigProve}(params, pk_c, \Sigma_J, J)$ for $J$.

Then the user constructs non-interactive zero-knowledge proofs that indeed $(S, T, C_{id}, C_s, C_t, C_J, id_{\mathcal{M}} \| info)$ are well formed. This is done by computing two proofs $\pi_F$ and $\pi_T$: $\pi_F$ proves that $(C_s, C_J, S) \in \mathcal{L}_S$ and is computed as described in Section 4.2, where $\mathcal{L}_S$ is defined as:

$$\mathcal{L}_S = \{C_s, C_x, y | \exists x, s, open_x, open_s \text{ such that } C_s = \mathsf{Com}(s, open_s) \wedge C_x = \mathsf{Com}(x, open_x) \wedge y = F_s(x)\} ;$$

[7] The original [CHL05] definition had an interactive Spend protocol, while we break it up into two non-interactive protocols: $\mathsf{SpendCoin}(params, W, pk_{\mathcal{M}}, info)$ and $\mathsf{VerifyCoin}(params, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin)$. The merchant sends the user a $info$, the user runs SpendCoin and gives the resulting e-coin for the merchant to verify using VerifyCoin. We prefer to use a non-interactive spend protocol because often two-way communication is not available or impractical, e.g. when sending an e-coin by email.

[8] These and the rest of the proofs in the issue protocol can be done using efficient sigma protocols [CS97b,Dam02] and their zero-knowledge compilers [Dam00].

[9] The merchant is responsible for assuring that $info$ is locally unique. Coins which have the same serial number and the same $id_{\mathcal{M}} \| info$ cannot be deposited and the damage lies with the merchant. The dangers that users get cheated by verifiers that do not accept coins with correct $info$ can be mitigated using techniques such as endorsed e-cash [CLM07].

$\pi_T$ proves that $(C_t, C_J, C_{id}, T, (id_{\mathcal{M}}|info)) \in \mathcal{L}_T$ and is computed as described in Section 4.3, where $\mathcal{L}_T$ is defined as:

$$\mathcal{L}_T = \{C_s, C_x, C_{sk}, tag, ch \mid \exists x, s, sk, open_x, open_s, open_{sk} \text{ such that } C_s = \mathsf{Com}(s, open_s) \wedge C_x = \mathsf{Com}(x, open_x) \wedge C_{sk} = \mathsf{Commit}(sk, open_{xsk}) \wedge tag = (g^{sk})^{ch} F_s(x)\} .$$

The user outputs a $coin = (S, T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T, id_{\mathcal{M}} \| info)$.

$\mathsf{VerifyCoin}(params, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin)$. To verify parses $coin$ as $(S, (T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T), id_{\mathcal{M}}{}' \| info)$ and checks that the following checks succeed: (1) Check that $id_{\mathcal{M}}{}' = f(pk_{\mathcal{M}})$. (2) $\mathsf{SigVerify}(params, pk_w, \pi_1, (C_{id}, C_s, C_t)) = \mathsf{accept}$. (3) $\mathsf{SigVerify}(params, pk_c, \pi_2, C_J) = \mathsf{accept}$. (4) $\mathsf{Verify}_{\mathcal{L}_S}(params_{GS}, (C_s, C_J, S), \pi_S) = \mathsf{accept}$. (5) $\mathsf{Verify}_{\mathcal{L}_T}(params_{GS}, (C_t, C_J, C_{id}, T, (id_{\mathcal{M}} \| info)), \pi_T) = \mathsf{accept}$.

Note that the merchant is responsible for assuring that $info$ is unique over all of his transactions. Otherwise his deposit might get rejected by the following algorithm.

$\mathsf{Deposit}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, state_{\mathcal{B}})$. The algorithm parses the coin as $coin = (S, T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T, id_{\mathcal{M}} \| info)$ and performs the same checks as VerifyCoin. The bank maintains a database $state_{\mathcal{B}}$ of all previously accepted coins. The output of the algorithm is an updated database $state'_{\mathcal{B}} = state_{\mathcal{B}} \cup \{coin\}$ and the flag $result$, that is computed as follows:

(i) If the coin verifies and if no coin with serial number $S$ is stored in $state_{\mathcal{B}}$, $result = \mathsf{accept}$ to indicate that the coin is correct and fresh. The bank deposits the value of the e-coin into the merchant's account and adds $coin$ to $state_{\mathcal{B}}$.

(ii) If the coin doesn't verify or if there is a coin with the same serial number and the same $id_{\mathcal{M}} \| info$ already stored in $state_{\mathcal{B}}$, $result = \mathsf{merchant}$ to indicate that the merchant cheated. The bank refuses to accept the e-coin because the merchant failed to properly verify it.

(iii) If the coin verifies but there is a coin with the same serial number $S$ but different $id_{\mathcal{M}} \| info$ in $state_{\mathcal{B}}$, $result = \mathsf{user}$ to indicate that a user doublespent. The bank pays the merchant (who accepted the e-coin in good faith) and punishes the double-spending user.

$\mathsf{Identify}(params, pk_{\mathcal{B}}, coin_1, coin_2)$ allows the bank to identify a double-spender. Parse $coin_1 = (S, (T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T), id_{\mathcal{M}1} \| info_1)$ and $coin_2 = (S', (T', C'_{id}, C'_s, C'_t, C'_J, \pi'_1, \pi'_2, \pi'_S, \pi'_T), id_{\mathcal{M}2} \| info_2)$.

The algorithm aborts if one of the coins doesn't verify, if $S \neq S'$, or if $id_{\mathcal{M}1} \| info_1 = id_{\mathcal{M}2} \| info_2$. Otherwise, the algorithm outputs $T_W = pk_{\mathcal{U}} = e((T/T')^{1/(id_{\mathcal{M}1} \| info_1 - id_{\mathcal{M}2} \| info_2)}, h)$ , which the bank compares to the trace information it stores after each withdrawal transaction.

**Theorem 6.** *This e-cash scheme is a secure compact e-cash scheme given the security of the P-signature scheme, the PRF, and the Groth-Sahai NIZK proof system.*

We review the security definition of compact e-cash in Appendix E and prove security in Appendix F.

### 5.1 Efficiency of our E-Cash Construction

Here we will consider an instantiation of the Groth-Sahai proof system based on SXDH.

Groth-Sahai commitments consist of 2 elements in the appropriate groups and require 2 multiexponentiations to compute. Groth-Sahai proofs consist of 4 elements in each group and require 4 multi-exponentiations to compute. A Groth-Sahai proof for $Q$ pairings (with $q = Q$) takes $16 + 4Q$ pairing operations to verify. An exception is a proof in which all committed elements are in the same group (without loss of generality say $G_1$). In this case, the proof consists of only 2 elements in $G_2$, and can be computed with 2 multiexponentiations and verified with $4 + 2Q$ pairings.

Our NIZK proof of equality requires 1 additional commitment in $G_2$, 1 full proof, and 1 proof with commitments only in $G_2$. This means the proof will have a total of 6 elements in each group and will take the same number of multiexponentiations to generate, and will require 26 pairings to verify.

Our P-signature proofs require commitments to $(h^{m_i}, u^{m_i}, w_i^{m_i})$ for each of the $n$ messages. Then we need commitments to $\sigma_1, \sigma_3 \in G_1$ and $\sigma_2 \in G_2$. Finally, we need proofs for each $(h^{m_i}, u^{m_i})$ pair, and 2 proofs to verify the $\sigma_1, \sigma_2, \sigma_3$. One of the proofs has $Q = 1$ the rest have $Q = 2$. We also need proofs that the values $w_i^{m_i}$ are computed correctly. This only involves commitments to elements in $G_2$, so each proof can be done with 2 elements of $G_1$ using the optimization mentioned above. Thus, the total P-signature proof will require $8n + 12$ elements of $G_1$ and $8n + 10$ elements of $G_2$, and will require the same number of multiexponentiations to compute. It will require $32n + 44$ pairings to verify.

Our serial number proof requires 3 additional commitments, 1 in $G_1$, and 2 in $G_2$. It requires 1 NIZK proof of equality in $G_1$ and 1 in $G_2$, and 1 witness-indistinguishable proof with $Q = 1$. Thus, in total we need 24 elements of $G_1$ and 26 elements of $G_2$. It will take the corresponding number of multiexponentiations to compute, and will require 98 pairings to verify.

Our doublespending tag proof requires 5 additional commitments, 2 in $G_1$ and 3 in $G_2$. It requires 4 NIZK proofs of equality, and 2 witness indistinguishable proofs, 1 with $Q = 1$ and 1 with $Q = 2$. Thus, in total we need 36 elements of $G_1$ and 38 elements of $G_2$. Verification will require 148 pairings.

A coin will consist of values $S, T$, four commitments in $G_2$, P-signature proofs for $n = 1$ and $n = 3$, a serial number proof, and a doublespending tag proof. Thus, the total size is 118 elements of $G_1$ and 124 elements of $G_2$, the total computation required to generate the coin is 118 multiexponentiations in $G_1$ and 124 multiexponentiations in $G_2$, and the total number of pairings to verify is 294.

In comparison, the construction of Camenisch et al. has coins which consist of 18 group elements, and require 18 multiexponentiations to compute, and 11 exponentiations to verify. However, of these 14 must be in an RSA group. Generally it is assumed that RSA groups must be significantly larger than prime order groups to obtain the same level of security. If we assume that RSA groups must be larger by at least a factor of 5 [10], then this would be the equivalent of 74 prime order group elements. Thus, our construction would generate coins which are larger only by roughly a factor of 4. (Verification will still be much slower than the Camenisch et al. construction, however.)

Furthermore, while the performance of our scheme is still on the border of being practical, its modular design would translate any performance improvements for GS proofs, P-signatures, and sVRFs into a more efficient e-cash scheme.

## References

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective id secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 54–73. Springer, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Verlag, 2004.

[BCC+08] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Delegatable anonymous credentials. Cryptology ePrint Archive, Report 2008/428, `http://eprint.iacr.org/2008/502`, 2008.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *Proceedings of the Fifth Theory of Cryptography Conference (TCC)*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 194–211. Springer-Verlag, 1990.

[10] http://www.keylength.com/en/3/

[BGdMM] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-Resistant Storage. Johns Hopkins University, CS Technical Report # TR-SP-BGMM-050705. ht\tp://spar.isi.jhu.edu/~mgreen/correlation.pdf, 2005.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[Bra93a] Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, April 1993.

[Bra93b] Stefan Brands. Untraceable off-line cash in wallets with observers. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, 1993.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography*, pages 1–15, 2007.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *ASIACRYPT '08: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, page to appear, London, UK, 2008. Springer-Verlag.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Verlag, 1990.

[Cha83] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology — Proceedings of CRYPTO '82*, pages 199–203. Plenum Press, 1983.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 201–210, New York, NY, USA, 2006. ACM Press.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In Ronald Cramer, editor, *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2004.

[CL07] Melissa Chase and Anna Lysyanskaya. Simulatable vrfs with applications to multi-theorem nizk. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2007.

[CLM07] Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. Endorsed e-cash. In *IEEE Symposium on Security and Privacy*, pages 101–115, 2007.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.

[CP93] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 390–407. Springer-Verlag, 1993.

[CPS94] 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 *Lecture Notes in Computer Science*, pages 428–432. Springer Verlag Berlin, 1994.

[CS97a] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.

[CS97b] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich, March 1997.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 13–25, Berlin, 1998. Springer Verlag.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.

[Dam02] Ivan Damgård. On $\Sigma$-protocols. Available at http://www.daimi.au.dk/~ivan/Sigma.ps, 2002.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proc. 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 542–552, 1991.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Public Key Cryptography 2005*, pages 416–432, 2005.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1987.

[FST06] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. http://eprint.iacr.org/.

[FTY96] 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 *Lecture Notes in Computer Science*, pages 286–300. Springer Verlag, 1996.

[FY92] 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.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Proc. 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 102–115. IEEE Computer Society Press, 2003.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, pages 228–244. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[GR04] S. Galbraith and V. Rotger. Easy decision diffie-hellman groups. *Journal of Computation and Mathematics*, 7:201–218, 2004.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. http://eprint.iacr.org/2007/155, 2007.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 150–164, London, UK, 1997. Springer-Verlag.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–130. IEEE Computer Society Press, 1999.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1992.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–553. IEEE Computer Society Press, 1999.

[Sco02] Mike Scott. Authenticated id-based key exchange and remote log-in with insecure to ken and pin number. http://eprint.iacr.org/2002/164, 2002.

[SPC95] 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 *Lecture Notes in Computer Science*, pages 209–219. Springer Verlag, 1995.

[STS99] Tomas Sander and Amnon Ta-Shma. Auditable, anonymous electronic cash extended abstract. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 555–572, London, UK, 1999. Springer-Verlag.

[Tro05] Mårten Trolin. A universally composable scheme for electronic cash. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 347–360. Springer, 2005.

[TS06] Isamu Teranishi and Kazue Sako. *k*-times anonymous authentication with a constant proving cost. In *Public Key Cryptography*, pages 525–542, 2006.

[Tsi97] Yiannis S. Tsiounis. *Efficient Electonic Cash: New Notions and Techniques*. PhD thesis, Northeastern University, Boston, Massachusetts, 1997.

[Ver04] Eric R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17(4):277–296, 2004.

## A Formal Assumptions

Boyen and Waters [BW07] defined the Hidden SDH assumption over symmetric bilinear maps $e : G \times G \to G_T$. We give a definition over asymmetric maps $e : G_1 \times G_2 \to G_T$. Note that in the symmetric setting, this is identical to the Boyen Waters HSDH assumption.

**Definition 6 (Hidden SDH).** *On input $g, g^x, u \in G_1$, $h, h^x \in G_2$ and $\{g^{1/(x+c_\ell)}, h^{c_\ell}, u^{c_\ell}\}_{\ell=1\ldots q}$, it is computationally infeasible to output a new tuple $(g^{1/(x+c)}, h^c, u^c)$. Formally, there exists a negligible*

*function $\nu$ such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k);$$
$$u \leftarrow G_1; x, \{c_\ell\}_{\ell=1\ldots q} \leftarrow Z_p;$$
$$(A, B, C) \leftarrow \mathcal{A}(p, G_1, G_2, G_T, e, g, g^x, h, h^x, u, \{g^{1/(x+c_\ell)}, g^{c_\ell}, u^{c_\ell}\}_{\ell=1\ldots q}) :$$
$$(A, B, C) = (g^{1/(x+c)}, h^c, u^c) \wedge c \notin \{c_\ell\}_{\ell=1\ldots q}] < \nu(k).$$

When $(p, G_1, G_2, G_T, e, g, h)$ and $H = h^x$ are fixed, we refer to tuples of the form $(g^{1/(x+c)}, h^c, u^c)$ as *HSDH tuples*.

Note that we can determine whether $(A, B, C)$ form an HSDH tuple using the bilinear map $e$, as follows: suppose we get a tuple $(A, B, C)$. We check that $e(A, BH) = e(g, h)$ and that $e(u, B) = e(C, h)$.

[BCKL08] introduced the Triple DH assumption.

**Definition 7 (Triple DH).** *On input $g, g^x, g^y, h, h^x, \{c_i, g^{1/(x+c_i)}\}_{i=1\ldots q}$, it is computationally infeasible to output a tuple $(h^{\mu x}, g^{\mu y}, g^{\mu xy})$ for $\mu \neq 0$. Formally, there exists a negligible function $\nu$ such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k);$$
$$(x, y) \leftarrow Z_p; \{c_i\}_{i=1\ldots q} \leftarrow Z_p;$$
$$(A, B, C) \leftarrow \mathcal{A}(p, G_1, G_2, G_T, e, g, g^x, g^y, h, h^x, \{c_i, g^{1/(x+c_i)}\}_{i=1\ldots q}) :$$
$$\exists \mu \neq 0 : (A, B, C) = (h^{\mu x}, g^{\mu y}, g^{\mu xy})] < \nu(k).$$

We also recall the DLIN and SXDH assumptions. These assumptions are needed for the Groth-Sahai pairing product equation proofs [GS07] (see Section 2).

**Definition 8 (Decisional Linear Assumption [BBS04]).** *There exists a negligible function $\nu$ such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); r, s \leftarrow Z_p; u, v, w \leftarrow G_1;$$
$$b \leftarrow \{0, 1\}; z_0 \leftarrow w^{r+s}; z_1 \leftarrow G_1 :$$
$$\mathcal{A}(p, G_1, G_2, G_T, e, g, h, u, v, w, u^r, v^s, z_b) = b] < 1/2 + \nu(k).$$

**Definition 9 (External Diffie-Hellman Assumption (XDH)).** *There exists a negligible function $\nu$ such that*

$$\Pr[(p, G_1, G_2, G_T, e, g, h) \leftarrow \mathsf{BilinearSetup}(1^k); r, s \leftarrow Z_p;$$
$$b \leftarrow \{0, 1\}; z_0 \leftarrow g^{rs}, z_1 \leftarrow G_1 : \mathcal{A}(p, G_p, G_T, e, g, g^r, g^s, z_b) = b] < 1/2 + \nu(k).$$

The XDH assumption can be similarly defined to hold in $G_2$. The **SXDH** assumption states that XDH holds in both $G_1$ and $G_2$. The SXDH assumption was first used by Scott [Sco02], and has been discussed and used extensively since [BBS04,GR04,Ver04,BGdMM].

Finally, our sVRF construction relies on the assumption that the DDHI assumption [CHL05,BB04a] holds even in a group with an efficient bilinear map. In standard groups, the DDHI assumption is as follows:

**Definition 10 ($Q$-DDHI).** *A family $\mathcal{G}$ of groups satisfies the $Q(k)$-decisional Diffie-Hellman inversion assumption if no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2 with non-negligible advantage where $instance$ and $challenge$ are defined as follows: $instance = (G, p, g, g^\alpha, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{Q(k)}})$ where $p$ is a prime of length $poly(k)$, $G$ is a group of order $p$ returned by $\mathcal{G}(q)$, $g \leftarrow G$, $\alpha \leftarrow Z_p^*$, challenge of type 1 is $g^{\frac{1}{\alpha}}$, while challenge of type 2 is $g^R$ for random $R \leftarrow Z_p^*$.*

Applied to bilinear groups, it would be:

**Definition 11 ($Q$-DDHI in bilinear groups).** *A family $\mathcal{G}$ of groups satisfies the $Q(k)$-bilinear Diffie-Hellman inversion assumption if no PPT $\mathcal{A}$, on input $(instance, challenge)$ can distinguish if its challenge is of type 1 or type 2 with non-negligible advantage where $instance$ and $challenge$ are defined as follows: $instance = (G_1, G_2, G_T, p, e, g, h, g^{\alpha}, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{Q(k)}})$ where $p$ is a prime of length $poly(k)$, $G_1, G_2, G_T$ are groups of order $p$ returned by $\mathcal{G}(q)$, $e : G_1 \times G_2 \to G_T$ is a bilinear map, $g \leftarrow G_1$, $h \leftarrow G_2$, $\alpha \leftarrow Z_p^*$, challenge of type 1 is $g^{\frac{1}{\alpha}}$, while challenge of type 2 is $g^R$ for random $R \leftarrow Z_p^*$.*

Note that this is slightly stronger than the $Q$-DBDHI assumption used in [DY05] to construct an efficient VRF (there the challenge is $e(g,h)^{\frac{1}{\alpha}}$ or a random element of $G_T$). However it is still weaker than the $Q$-BDHBI assumption used in the previous sVRF construction [CL07].

## B Proof of Theorem 1

**Theorem.** *Let $F(m) = (h^m, u^m)$. The signature scheme in Section 3.2 is $F$-secure given the HSDH and TDH assumptions.*

*Proof.* We distinguish two types of forgeries. In a Type 1 forgery the signature consists of a tuple $(\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1 \neq \sigma_1^{(q)}$ for any $\sigma_1^{(q)}$ used in answering the forger's signature queries. We will show how a Type 1 forger can be used to break the HSDH assumption. In a Type 2 forgery, $\exists q : \sigma_1 = \sigma_1^{(q)}$. We will show how a Type 2 forger can be used to break the TDH assumption.

*Type 1 forgeries:* The reduction gets as input $g, \tilde{v} = g^{\alpha}, u, h, v = h^{\alpha}, \{S_q = g^{1/(\alpha+\sigma_q)}, H_q = h^{c_q}, U_q = u^{c_q}\}_{q=1\ldots Q}$, as well as a description of the groups $p, G_1, G_2, G_T$. It needs to compute a new tuple $(g^{1/(\alpha+c)}, h^c, u^c)$ such that $\forall q : c \neq c_q$.

$\mathsf{Setup}(1^k)$**.** The reduction computes $params_{GS}$ and gives the adversary public parameters $params = (p, G_1, G_2, G_T, g, h, u, params_{GS})$.

$\mathsf{Keygen}(params)$**.** The reduction picks random $\beta_1, \ldots, \beta_n \leftarrow Z_p$. It computes, $\forall i \in [1, n] : w_i = h^{\beta_i}$ and $\tilde{w_i} = g^{\beta_i}$. The reduction gives the adversary the public key $pk = (v, \boldsymbol{w}, \tilde{v}, \tilde{\boldsymbol{w}})$. (The secret key is $sk = (\alpha, \boldsymbol{\beta})$, though the reduction does not know $\alpha$.)

$\mathcal{O}\mathsf{Sign}(params, (\alpha, \beta), \boldsymbol{m})$**.** Note that the reduction does not know $\alpha$ and will use elements of the HSDH instance to answer signature queries. At the first sign query, the reduction sets the counter $q = 1$, and increments it after responding to each sign query. The reduction will implicitly set $c_q = r + \sum_{i=1}^n \beta_i m_i$. Thus, $r = (c_q - \sum_{i=1}^n \beta_i m_i)$. However, the reduction does not know $c_q$ and $r$. The reduction computes $\sigma_1, \sigma_2, \sigma_3$ as follows:

$$\sigma_1 = S_q = g^{1/(\alpha+c_q)}$$
$$\sigma_2 = H_q / h^{\sum_{i=1}^n \beta_i m_i} = h^{c_q - \sum_{i=1}^n \beta_i m_i} = h^r$$
$$\sigma_3 = \left(U_q / u^{\sum_{i=1}^n \beta_i m_i}\right) = u^{(c_q - \sum_{i=1}^n \beta_i m_i)} = u^r$$

**Forgery.** Eventually, the adversary returns a Type 1 forgery $F(\boldsymbol{m}) = (h^{m_1}, u^{m_1}, \ldots, h^{m_n}, u^{m_n}) = (A_1, B_1, \ldots, A_n, B_n)$ and $\sigma_1 = g^{1/(\alpha+r+\beta_1 m_1 + \cdots + \beta_n m_n)}$, $\sigma_2 = h^r$, and $\sigma_3 = u^r$. We implicitly set $c = r + \beta_1 m_1 + \cdots + \beta_n m_n$. We now have $\sigma_1 = g^{1/(\alpha+c)}$. We compute the rest of the HSDH tuple:

$$A = \sigma_2 \prod_{i=1}^n A_i^{\beta_i} = h^r \prod_{i=1}^n h^{m_i \beta_i} = h^c$$
$$B = \sigma_3 \prod_{i=1}^n B_i^{\beta_i} = u^r \prod_{i=1}^n u^{m_i \beta_i} = u^c$$

The tuple $(\sigma_1, A, B)$ is a fresh HSDH tuple because this is a Type 1 forgery ($\forall q : c \neq c_q$).

*Type 2 forgeries:* The reduction gets as input $g, G = g^x, u = g^y, h, H = h^x, \{c_q, S_q = g^{1/(x+c_q)}\}_{q=1\ldots Q}$, as well as a description of the groups $p, G_1, G_2, G_T$. It needs to compute the tuple $(h^{\mu x}, g^{\mu y}, g^{\mu xy})$ such that $\mu \neq 0$.

$\mathsf{Setup}(1^k)$**.** The reduction computes $params_{GS}$ and gives the adversary public parameters $params = (p, G_1, G_2, G_T, g, h, u, params_{GS})$.

$\mathsf{Keygen}(params)$**.** The reduction chooses a random $t \leftarrow \{1, n\}$ and pick random $\alpha, \{\beta_i\}_{i=1,i\neq t}^n \leftarrow Z_p$. It computes, $\forall i \in [1, n] : w_i = h^{\beta_i}$ and $\tilde{w}_i = g^{\beta_i}$. Then, the reduction picks a random $\gamma \leftarrow Z_p$, and sets $w_t = H^\gamma = h^{x\gamma}$ and $\tilde{w}_t = G^\gamma = g^{x\gamma}$. The reduction gives the adversary the public key $pk = (v, \boldsymbol{w}, \tilde{v}, \tilde{\boldsymbol{w}})$. The secret key is $sk = (\alpha, \boldsymbol{\beta})$, though the reduction does not know $\beta_t = x\gamma$.

$\mathcal{O}\mathsf{Sign}(params, (\alpha, \beta), \boldsymbol{m})$**.** At the first sign query, the reduction sets the counter $q = 1$, and increments it after responding to each sign query. The reduction sets $c_q = (\alpha + r + \sum_{i=1,i\neq t}^n \beta_i m_i)/\gamma m_t$, and solves for $r$. Then it is easy to verify that

$$(x + c_q)\gamma m_t = x\gamma m_t + \alpha + r + \sum_{i=1,i\neq t}^{n} \beta_i m_i = \alpha + r + \sum_{i=1}^{n} \beta_t m_t.$$

This is precisely the inverse of the exponent which forms the first part of the signature, $\sigma_1$. The reduction sets $\sigma_1 = S_q^{1/\gamma m_t} = g^{1/(x+c_q)\gamma m_t}$. Since the reduction knows $r$, it computes $\sigma_2 = h^r$ and $\sigma_3 = u^r$.

**Forgery.** Eventually, the adversary returns a Type 2 forgery $F(\boldsymbol{m}) = (h^{m_1}, u^{m_1}, \ldots, h^{m_n}, u^{m_n}) = (A_1, B_1, \ldots, A_n, B_n)$ and $\sigma_1 = g^{1/(\alpha+r+\sum_{i=1}^n \beta_i m_i)}$, $\sigma_2 = h^r$, and $\sigma_3 = u^r$.

Since this is a Type 2 forgery, the reduction already has a message / signature pair such that $r + \sum_{i=1}^n \beta_i m_i = r^{(q)} + \sum_{i=1}^n \beta_i m_i^{(q)}$. Since it is a forgery, $\exists m_i \in \boldsymbol{m} : m_i \neq m_i^{(q)}$. With probability $1/n$, $i = t$.

That means that $m_t \neq m_t^{(q)}$ but $\beta_t m_t + r + \sum_{i=1,i\neq t}^n \beta_i m_i = \beta_t m_t^{(q)} + r^{(q)} + \sum_{i=1,i\neq t}^n \beta_i m_i^{(q)}$.

Now, we will implicitly set $\mu = (m_t^{(q)} - m_t)\gamma$ (Note that if we have guessed $t$ correctly, this will be nonzero). We cannot compute this value, however we will be able to compute $h^{\mu x}, g^{\mu y}, g^{\mu xy}$ as follows:

Compute the following values:

$$W_1 = ((u^{m_t^{(q)}})/B_t)^\gamma = (u^{m_t^{(q)} - m_t})^\gamma = u^\mu = g^{y\mu}$$

$$W_2 = \prod_{i=1,i\neq t}^{n} (A_i/(h^{m_i^{(q)}}))^{\beta_i} \sigma_2/h^{r^{(q)}} = h^{\sum_{i=1,i\neq t}^n \beta_i (m_i - m_i^{(q)})} h^{(r - r^{(q)})} = h^{\beta_t (m_t^{(q)} - m_t)}$$

$$= h^{x\gamma(m_t^{(q)} - m_t)} = h^{x\mu}$$

$$W_3 = \prod_{i=1,i\neq t}^{n} (B_i/(u^{m_i^{(q)}}))^{\beta_i} (\sigma_3/u^{r^{(q)}}) = u^{\sum_{i=1,i\neq t}^n \beta_i (m_i - m_i^{(q)})} u^{(r - r^{(q)})} = u^{\beta_t (m_t^{(q)} - m_t)}$$

$$= u^{x\gamma(m_t^{(q)} - m_t)} = u^{x\mu} = g^{xy\mu}$$

The reduction will output $W_1, W_2, W_3$, which will be a valid tuple iff this is a type 2 forgery such that $m_t \neq m_t^{(q)}$.

## C Formal Definition for sVRF and Proofs

First we review the definition of sVRFs (as defined in [CL07]).

**Definition 12 (Trapdoor-indistinguishable sVRF).** *Let* $\mathsf{Setup}(\cdot)$ *be an algorithm generating public parameters* $params_{VRF}$ *on input security parameter* $1^k$. *Let* $D(params_{VRF})$ *and* $R(params_{VRF})$ *be families of efficiently samplable domains for all* $params_{VRF} \in \mathsf{Setup}$. *The set of algorithms* $(G, \mathsf{Eval}, \mathsf{Prove}, \mathsf{Verify})$ *constitutes a verifiable random function (VRF) for parameter model* Setup*, input domain* $D(\cdot)$ *and output range* $R(\cdot)$ *if*

**Correctness** *Informally, correctness means that the verification algorithm* Verify *will always accept* $(params_{VRF}, pk, x, y, \pi)$ *when* $y = \mathsf{Eval}(params_{VRF}, sk, x)$*, and* $\pi$ *is the proof of this fact generated using* Prove*. More formally,* $\forall k, params_{VRF} \in \mathsf{Setup}(1^k), x \in D(params_{VRF})$*,*

$$\Pr[(pk, sk) \leftarrow G(params_{VRF}); y = \mathsf{Eval}(params_{VRF}, sk, x); \pi \leftarrow \mathsf{Prove}(params_{VRF}, sk, x); \\ b \leftarrow \mathsf{Verify}(params_{VRF}, pk, x, y, \pi) \ : \ b = 1] = 1 \ .$$

**Pseudorandomness** *Informally, pseudorandomness means that, on input* $(params_{VRF}, pk)$*, even with oracle access to* $\mathsf{Eval}(params_{VRF}, sk, \cdot)$ *and* $\mathsf{Prove}(params_{VRF}, sk, \cdot)$*, no adversary can distinguish* $F_{pk}(x)$ *from a random element of* $R(params_{VRF})$ *without explicitly querying for it. More formally,* $\forall$ *PPT* $\mathcal{A}$*,* $\exists$ *negligible* $\nu$ *such that*

$$\Pr[params_{VRF} \leftarrow \mathsf{Setup}(1^k); (pk, sk) \leftarrow G(params_{VRF}); \\ (Q_e, Q_p, x, state) \leftarrow \mathcal{A}^{\mathsf{Eval}(params_{VRF}, sk, \cdot), \mathsf{Prove}(params_{VRF}, sk, \cdot)}(params_{VRF}, pk); \\ y_0 = \mathsf{Eval}(params_{VRF}, sk, x); y_1 \leftarrow R(params_{VRF}); b \leftarrow \{0,1\}; \\ (Q'_e, Q'_p, b') \leftarrow \mathcal{A}^{\mathsf{Eval}(params_{VRF}, sk, \cdot), \mathsf{Prove}(params_{VRF}, sk, \cdot)}(state, y_b) \\ : \ b' = b \wedge x \notin (Q_e \cup Q_p \cup Q'_e \cup Q'_p)] \leq 1/2 + \nu(k)$$

*where* $Q_e$ *and* $Q_p$ *denote, respectively, the contents of the query tape that records* $\mathcal{A}$*'s queries to its* Eval *and* Prove *oracles in the first query phase, and* $Q'_e$ *and* $Q'_p$ *denote the query tapes in the second query phase.*

**Verifiability.** *For all* $k$*, for all* $params_{VRF} \in \mathsf{Setup}(1^k)$*, there do not exist* $(pk, x, y_1, \pi_1, y_2, \pi_2)$ *such that* $y_1 \neq y_2$*, but* $\mathsf{Verify}(params_{VRF}, pk, x, y_1, \pi_1) = \mathsf{Verify}(params_{VRF}, pk, x, y_2, \pi_2) = accept$*.*

**Trapdoor-Indistinguishable Simulatability.**[11] *There exist algorithms* $(\mathsf{SimSetup}, \mathsf{SimG}, \mathsf{SimProve})$ *such that the distribution* $\mathsf{Setup}(1^k)$ *is computationally indistinguishable from the distribution* $\mathsf{SimSetup}(1^k)$ *and for all PPT* $\mathcal{A}$*,* $\mathcal{A}$*'s views in the following two games are indistinguishable:*

- **Game Real Proofs.** $(params_{VRF}, t) \leftarrow \mathsf{SimSetup}(1^k)$*,* $(pk, sk) \leftarrow G(params_{VRF})$ *and then* $\mathcal{A}($ $params_{VRF}, t, pk)$ *gets access to the following oracle* Real*: On query* $x$*,* Real *returns* $y = \mathsf{Eval}(params_{VRF}, sk, x)$ *and* $\pi \leftarrow \mathsf{Prove}(params_{VRF}, sk, x)$*.*
- **Game Simulated Proofs.** $(params_{VRF}, t) \leftarrow \mathsf{SimSetup}(1^k)$*,* $(pk, sk) \leftarrow \mathsf{SimG}(params_{VRF}, t)$*, and then* $\mathcal{A}(params_{VRF}, t, pk)$ *gets access to the following oracle* Sim*: On query* $x$*,* Sim *(1) checks if* $x$ *has previously been queried, and if so, computes* $\pi \leftarrow \mathsf{SimProve}(params_{VRF}, sk, x, y, t)$ *for the stored* $y$ *and returns* $(y, \pi)$*; (2) otherwise, obtains a random* $y \leftarrow R(params_{VRF})$ *and* $\pi \leftarrow \mathsf{SimProve}(params_{VRF}, sk, x, y, t)$*, returns* $(y, \pi)$ *and stores* $y$*.*

*In the rest of the paper, we will refer to the output of* $\mathsf{Eval}(params, sk, x)$ *as* $F_s(x)$ *when the parameters are clear.*

Trapdoor-indistinguishable simulatability can be shown via a simple hybrid argument to imply full simulatability, where the adversaries can interact with many public keys and still cannot distinguish $\mathsf{Setup}, \mathsf{Prove}, \mathsf{Eval}$ from $\mathsf{SimSetup}, \mathsf{SimProve}$ and random sampling from $R(params_{VRF})$. To prove simulatability property of our sVRF construction, we will thus prove that it satisfies trapdoor-indistinguishable simulatability.

[1] Note that TI-Simulatability implies standard Simulatability as described in [CL07], which in turn implies pseudorandomness.

### C.1 Proof of Theorem 4

**Theorem.** *The construction in Section 4 with domain size $q$ is a strong sVRF under the $q$-DDHI assumption for $G_1$ and under the assumption that the Groth-Sahai proof system is secure.*

*Proof. Correctness and Verifiability* follow from the corresponding properties of the WI and NIZK GS proof systems.
*Pseudorandomness* can be shown via an approach very similar to our proof below for Simulatability, so we will not show it here.
*Trapdoor-Indistinguishable Simulatability* We define the following simulator algorithms:

$\mathsf{SimSetup}(1^k)$. Let $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map of order $q$. Let $g$ be a generator for $G_1$, and $h$ be a generator for $G_2$. Let $(params_{GS}, auxsim) \leftarrow \mathsf{GSSimSetup}(q, G_1, G_2, G_2, G_T, g, h)$ be simulated parameters for a Groth-Sahai NIZK proof system. Output parameters $params_{VRF} = (q, G_1, G_2, G_T, g, h, params_{GS})$ and trapdoor $t = sim$.

$\mathsf{SimG}(params_{VRF})$. Pick a random seed $s \leftarrow Z_p$ and random opening information $open_s$ and output $sk = (s, open_s)$ and public key $pk = \mathsf{GSCom}(h^s, open_s)$.

$\mathsf{SimProve}(params_{VRF}, sk = (s, open_s), x, y, t)$. Compute $y' = g^{\frac{1}{s+x}}$ and commitment $C = \mathsf{GSCom}(y')$.
Use the NIZK simulator to compute simulated proof $\pi_1$ that $C$ is a commitment to $y$.
Create $\pi_2$ as an honest GS witness indistinguishable proof that $C$ is a commitment to $y'$ and $pk$ is a commitment to $S$ such that $e(y', Sh^x) = e(g, h)$
Output $\pi = (\pi_1, \pi_2)$.

Now we need to show that Game Simulated Proofs, when instantiated using this simulator, is indistinguishable from Game Real Proofs. We will do this by considering a series of intermediate games.

First consider the following intermediate simulator algorithm:

$\mathsf{HybSimProve}(params_{VRF}, sk = (s, open_s), x, t)$. Computes $y = g^{\frac{1}{s+x}}$, and then proceeds as $\mathsf{SimProve}$:
It computes $y' = g^{\frac{1}{s+x}}$ and commitment $C = \mathsf{GSCom}(y')$. It uses the NIZK simulator to compute simulated proof $\pi_1$ that $C$ is a commitment to $y$.
It creates $\pi_2$ honestly: Create $\pi_2$ as an honest GS witness indistinguishable proof that $C$ is a commitment to $y'$ and $pk$ is a commitment to $S$ such that $e(y', Sh^x) = e(g, h)$
It outputs $\pi = (\pi_1, \pi_2)$.

**Game Hybrid Sim 1.** Runs as Game Real Proofs except that it uses $\mathsf{HybSimProve}$ instead of $\mathsf{Prove}$: $(params_{VRF}, t) \leftarrow \mathsf{SimSetup}(1^k)$, $(pk, sk) \leftarrow G(params_{VRF})$ and then $\mathcal{A}(params_{VRF}, t, pk)$ gets access to the following oracle $\mathsf{Real}$: On query $x$, $\mathsf{Real}$ returns $y = \mathsf{Eval}(params_{VRF}, sk, x)$ and $\pi \leftarrow \mathsf{HybSimProve}(params_{VRF}, sk, x, t)$.

Game Hybrid Sim 1 is indistinguishable from Game Real Proofs by the zero knowledge property of the GS NIZK and by the perfect WI property of the GS WI proofs.

Now consider a second intermediate game:

**Game Hybrid Sim 2.** Runs as Game Simulated Proofs except that it computes $y$ correctly: $(params_{VRF}, t) \leftarrow \mathsf{SimSetup}(1^k)$, $(pk, sk = (s, open_s)) \leftarrow \mathsf{SimG}(params_{VRF}, t)$, and then $\mathcal{A}(params_{VRF}, t, pk)$ gets access to the following oracle $\mathsf{Sim}$: On query $x$, $\mathsf{Sim}$ (1) checks if $x$ has previously been queried, and if so, computes $\pi \leftarrow \mathsf{SimProve}(params_{VRF}, sk, x, y, t)$ for the stored $y$ and returns $(y, \pi)$; (2) otherwise, it computes $y = g^{\frac{1}{s+x}}$ and $\pi \leftarrow \mathsf{SimProve}(params_{VRF}, sk, x, y, t)$, returns $(y, \pi)$, and stores $y$.

First note that Game Hybrid Sim 2 is identical to Game Hybrid Sim 1. Next, we can see that by theorem 3, Game Hybrid Sim 2 is indistinguishable from Game Simulated Proofs. Thus, Game Simulated Proofs is indistinguishable from Game Real Proofs, and the simulatability property holds.

### C.2 Proof of Theorem 5

**Theorem.** *The proof system in Section 4.2 is a secure composable zero knowledge proof system for the language $\mathcal{L}_S(params)$, where $params$ is output by* Setup.

*Proof.* Correctness and Soundness follow from the corresponding properties of the underlying proof systems.

Thus, we need only show zero knowledge. Consider the following simulator:

SimSetup$(1^k)$. runs the GS simulation setup to generate simulated parameters $params$ and trapdoor $sim$.

SimProve$(params, sim, Com_x, Com_s, y)$. We first choose random $s', x' \leftarrow Z_p$, random opening information $open'_s, open'_x$ and form new commitments $C'_s = \mathsf{GSCom}(h^{s'}, open'_s)$ and $C'_x = \mathsf{GSCom}(h^{x'}, open'_x)$. Then we use the GS NIZK simulator to compute simulated zero knowledge proof $\pi_1$ that $C_s$ and $C'_s$ are commitments to the same value and simulated proof $\pi_2$ that $C_x$ and $C'_x$ are commitments to the same value using the techniques described in Appendix G.
Next, we compute a commitment $C'_y$ to $F_{s'}(x')$ and use the GS NIZK simulator to generate simulated proof $\pi_3$ that $C'_y$ is a commitment to $y$ as in Appendix G.
Finally, we compute a GS witness indistingushable proof $\pi_4$ that the value committed to in $C'_y$ is the correct output given the seed in $C'_s$ and the input in $C'_x$. (Note that this statement is true given our choice of $C'_y, C'_s, C'_x$.)
The final proof is $\pi = (C'_s, C'_x, C'_y, \pi_1, \pi_2, \pi_3, \pi_4)$.

Note that when parameters are generated by SimSetup, the proof $\pi_4$ and the commitments $C'_y, C'_s, C'_x$ generated by SimProve are distributed identically to those generated by Prove. Further, by the composable zero knowledge properties of the GS NIZK for equality of committed values, the simulated proofs $\pi_1, \pi_2, \pi_3$ will also be distributed identically to those generated by the honest Prove algorithm. Thus, SimSetup, SimProve as described here satisfy the definition of zero knoweldge for Setup, Prove, Verify.

## D A Construction for NIZK Proofs Doublespending Equations

Here we give a proof system for the language $\mathcal{L}_T$ presented in Section 4.3.

Prove$(params, C_s, C_x, C_{sk}, tag, ch, s, open_s, x, open_x, sk, open_{sk})$. We first form new commitments $C'_s = \mathsf{GSCom}(h^s, open'_s)$, $C'_x = \mathsf{GSCom}(h^x, open'_x)$, and $C'_{sk} = \mathsf{GSCom}(h^{sk}, open'_{sk})$. Then we compute zero knowledge proofs $\pi_1$ for ($C_s$,$C'_s$), $\pi_2$ for ($C_x$, $C'_x$), and $\pi_3$ for ($C_{sk}, C'_{sk}$), showing that both commitments in each pair commit to the same value using the techniques described in Appendix G.
Next, we compute a commitment $C'_y$ to $F_s(x)$, and a commitment $C''_{sk} = \mathsf{GSCom}(g^{sk}, open''_{sk})$.
We then compute a GS witness indistingushable proof $\pi_4$ that the value committed to in $C'_y$ is the correct output given the seed in $C'_s$ and the input in $C'_x$. I.e. that $C'_y$ commits to $Y$, $C'_s$ commits to $S$ and $C'_x$ commits to $X$ such that $e(Y, SX) = e(g, h)$.
Next we compute a GS witness indistinguishable proof $\pi_5$ that the value committed to in $C''_{sk}$ is correct with respect to $C'_{sk}$, i.e that $C''_{sk}$ commits to $K''$ and $C'_{sk}$ commits to $K'$ such that $e(K'', h) = e(g, K')$.
We can also compute $C'_{tag} = {C''_{sk}}^{ch} C'_y$. Note that by the homomorphic properties of the commitment scheme, this means $C'_{tag}$ should be a commitment to $(g^{sk})^{ch} y$ which is the correct value for $tag$.
Finally, we compute a zero knowledge proof $\pi_6$ that $C'_{tag}$ is a commitment to $tag$ as in Appendix G.
The final proof is $\pi = (C'_s, C'_x, C'_{sk}, C'_y, C''_{sk}, C'_{tag}, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6)$.

Verify$(params, C_s, C_x, C_{sk}, ch, \pi = (C'_s, C'_x, C'_{sk}, C'_y, C''_{sk}, C'_{tag}, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6))$. Uses the Groth-Sahai verification procedure to verify $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6$ with respect to $C_s, C_x, C_{sk}, info, C'_s, C'_x, C'_{sk}, C'_y, C''_{sk}, C'_z$.

**Theorem 7.** *The proof system* Setup, Prove, Verify *is a secure composable zero knowledge proof system for the language $\mathcal{L}_T(params)$ described above, where $params$ is output by* Setup.

*Proof.* Correctness and Soundness follow from the corresponding properties of the underlying proof systems.

To prove zero knowledge, consider the following simulator algorithms:

SimSetup$(1^k)$. runs the GS simulation setup to generate simulated parameters $params$ and trapdoor $sim$.

SimProve$(params, sim, C_x, C_s, C_{sk}, tag, ch)$. We first choose random $s', x', sk' \leftarrow Z_p$, random opening information $open'_s, open'_x, open'_{sk}$ and form new commitments $C'_s = \mathsf{GSCom}(h^{s'}, open'_s)$, $C'_x = \mathsf{GSCom}(h^{x'}, open'_x)$,and $C'_{sk} = \mathsf{GSCom}(h^{sk'}, open'_{sk})$.

Then we use the GS NIZK simulator to compute simulated zero knowledge proof $\pi_1$ that $C_s$ and $C'_s$ are commitments to the same value and simulated proof $\pi_2$ that $C_x$ and $C'_x$ are commitments to the same value, and simulated proof $\pi_3$ that $C_{sk}$ and $C'_{sk}$ are commitments to the same value using the techniques described in Appendix G.

Next, we compute a commitment $C'_y$ to $F_{s'}(x')$ and $C''_{sk}$ to $g^{sk'}$.

Then we compute a GS witness indistingushable proof $\pi_4$ that the value committed to in $C'_y$ is the correct output given the seed in $C'_s$ and the input in $C'_x$. (Note that this statement is true given our choice of $C'_y, C'_s, C'_x$.)

Similarly, we compute a GS witness indistinguishable proof $\pi_5$ that the value committed to in $C''_{sk}$ is correct with respect to the value committed to in $C'_{sk}$.

We also compute $C'_{tag} = C'_{sk}(C'_y)^{ch}$.

Finally, we use the GS simulator to generate simulated proof $\pi_6$ that $C'_{tag}$ is a commitment to $tag$ as in Appendix G.

The final proof is $\pi = (C'_s, C'_x, C'_{sk}, C'_y, C''_{sk}, C'_{tag}, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6)$.

Note that when parameters are generated by SimSetup, the proofs $\pi_4$ and $\pi_5$ and the commitments $C'_y, C'_s, C'_x, C'_{sk}, C'_y, C''_{sk}, C'_{tag}$ generated by SimProve are distributed identically to those generated by Prove. Further, by the composable zero knowledge properties of the GS NIZK for equality of committed values, the simulated proofs $\pi_1, \pi_2, \pi_3, \pi_6$ will also be distributed identically to those generated by the honest Prove algorithm. Thus, SimSetup, SimProve as described here satisfy the definition of zero knoweldge for Setup, Prove, Verify.

## E Definition of Compact e-Cash

Compact e-cash as defined by [CHL05] involves a bank $\mathcal{B}$ as well as many users $\mathcal{U}$ and merchants $\mathcal{M}$. Merchants are treated as a special type of user that have a publicly known identifier $id_{\mathcal{M}}$. The parties $\mathcal{B}, \mathcal{U}$, and $\mathcal{M}$ interact using the algorithms CashSetup, BankKG, UserKG, SpendCoin, VerifyCoin, Deposit, Identify, and the interactive protocol Withdraw. We write Protocol$(\mathcal{A}(I_{\mathcal{A}}), \mathcal{B}(I_{\mathcal{B}}))$ to denote an interactive protocol Protocol between $\mathcal{A}$ and $\mathcal{B}$ with secret inputs $I_{\mathcal{A}}, I_{\mathcal{B}}$ and secret outputs $O_{\mathcal{A}}, O_{\mathcal{B}}$ respectively.

CashSetup$(1^k)$ creates the public parameters $params$.

BankKG$(params, n)$ outputs the key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ that is used by the bank to issue wallets of $n$ coins. For simplicity, we assume that the secret key contains the corresponding public key.

UserKG$(params)$ generates a user (or merchant) key pair $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$. The keys are used for authentication and non-repudiation.[12]

[12] Our scheme does not involve the user's secret in the creation of the wallet. This means that we are unable to implement the exculpability scheme of [CHL05]. Instead we provide a different solution.

$\mathsf{Withdraw}(\mathcal{U}(params, pk_{\mathcal{B}}, sk_{\mathcal{U}}), \mathcal{B}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}}))$ is an interactive protocol in which a user withdraws a *wallet* $W$ of $n$ coins from the bank where $n$ is specified in $pk_{\mathcal{B}}$. The wallet includes the public key of the bank. The bank learns some trace information $T_W$ that it can later use to identify double-spenders. After a successful protocol run the bank adds $T_W$ to its database $DB_T$.

$\mathsf{SpendCoin}(params, W, pk_{\mathcal{M}}, info)$ allows a user with a non-empty wallet $W$ and some unique transaction information $info$ to create a coin. The output of the algorithm is $(W', coin)$, the updated wallet and an e-coin that can be given to a merchant. The e-coin consists of a serial number $S$, transaction information $info$, and a proof $\pi$.

$\mathsf{VerifyCoin}(params, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin)$ allows a merchant to verify $coin = (S, \pi, pk_{\mathcal{M}} \| info, )$ received from a user. The output of the algorithm is either accept or reject. The merchant accepts the coin on accept but only if he has never accepted a coin with the same $info$ before.

$\mathsf{Deposit}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, state_{\mathcal{B}})$ allows the bank to verify a coin received from merchant. The bank needs to maintain a database $state_{\mathcal{B}}$ of all previously accepted coins. The output of the algorithm is an updated database $state'_{\mathcal{B}}$ and the flag $result$, which can have three values:

(i) accept indicates that the coin is correct and fresh. The bank deposits the value of the e-coin into the merchant's account and adds $(pk_{\mathcal{M}}, coin)$ to $state_{\mathcal{B}}$.

(ii) merchant indicates that either $\mathsf{VerifyCoin}(params, sk_{\mathcal{M}}, pk_{\mathcal{B}}, coin) = 0$, or that $state_{\mathcal{B}}$ already contains an entry $(pk_{\mathcal{M}} \| coin)$. The bank refuses to accept the e-coin because the merchant failed to properly verify it.

(iii) user indicates that there exists a second coin with the same serial number $S$ registered in $state_{\mathcal{B}}$. (Using the two coins the bank will identify the double-spending user.) The bank pays the merchant (who accepted the e-coin in good faith) and punishes the double-spending user.

$\mathsf{Identify}(params, pk_{\mathcal{B}}, coin, coin')$ allows the bank to identify a double-spender. The algorithm outputs $T_W$, which the bank compares to the trace information it stores after each withdrawal transaction.

*Notes.* Camenisch et al. [CHL05] only define spending as the interactive protocol $\mathsf{Spend}(\mathcal{U}(params, W, pk_{\mathcal{M}}), \mathcal{M}(params, sk_{\mathcal{M}}, pk_{\mathcal{B}})$. We can derive their protocol from our non-interactive algorithms. First the merchant sends the user $info$. Then the user runs $\mathsf{SpendCoin}(params, W, pk_{\mathcal{M}}, info)$ and sends the resulting coin back to the merchant. The merchant accepts the e-coin only if $\mathsf{VerifyCoin}(params, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin)$ outputs accept and the $info$ used to construct the e-coin is correct. Non-interactive spend protocols are important when two-way communication is not available or impractical, e.g. when sending an e-coin by email.

**Definition 13 (Secure Compact E-Cash with Non-Interactive Spend).** *A compact e-cash scheme consists of the non-interactive algorithms* CashSetup, BankKG, UserKG, SpendCoin, VerifyCoin, Deposit, Identify, *and the interactive protocol* Withdraw. *We say that such a scheme is secure if it has the* Correctness, Anonymity, Balance, *and* Identification *properties.*

**Correctness.** When the bank and user are honest, and the user has sufficient funds, Withdraw will always succeed. An honest merchant will always accept an e-coin from an honest user. A honest bank will always accept an e-coin from an honest merchant.

**Anonymity.** A malicious coalition of banks and merchants should not be able to distinguish if the Spend protocol is executed by honest users or by a simulator that does not know any of the users' secret data. Formally, there must exist a simulator $\mathsf{Sim} = (\mathsf{SimCashSetup}, \mathsf{SimSpend})$, such that for all non-uniform polynomial time $\mathcal{A}$ there exists a negligible function $\nu$ such that:

$$\Big| Pr[params \leftarrow \mathsf{CashSetup}(1^k); (pk_{\mathcal{B}}, state) \leftarrow \mathcal{A}_1(params) : \mathcal{A}_2^{\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, \cdot, \cdot)}(state) = 1] - Pr[(params, sim) \leftarrow \mathsf{SimCashSetup}(1^k); (pk_{\mathcal{B}}, state) \leftarrow \mathcal{A}_1(params) : \mathcal{A}_2^{\mathcal{O}_{\mathsf{SimSpend}}(params, pk_{\mathcal{B}}, \cdot, \cdot, \cdot), \mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, \cdot, \cdot)}(state) = 1] \Big| < \nu(k)$$

The oracles $\mathcal{O}_{\mathsf{GetKey}}$, $\mathcal{O}_{\mathsf{Withdraw}}$, $\mathcal{O}_{\mathsf{SpendCoin}}$, and $\mathcal{O}_{\mathsf{SimSpend}}$ are defined as follows:

$\mathcal{O}_{\mathsf{GetKey}}(params, i)$**.** The oracle returns $pk_{\mathcal{U}i}$, the public-key of user $\mathcal{U}_i$. If $pk_{\mathcal{U}i}$ doesn't exist, the oracle generates $(pk_{\mathcal{U}i}, sk_{\mathcal{U}i})$ using $\mathsf{UserKG}(params)$.

$\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, i, j)$**.** The oracle runs the Withdraw protocol with the adversary: $\mathsf{Withdraw}(\mathcal{U}(params, pk_{\mathcal{B}}, sk_{\mathcal{U}i}), \mathcal{A}_2(state))$. The adversary plays the role of the bank and the oracle takes the role of user $\mathcal{U}_i$. If $sk_{\mathcal{U}i}$ doesn't exist, the oracle generates it using $\mathsf{UserKG}(params)$. The value $j$ serves to identify the wallet to the oracle for later use; the adversary must use a fresh value $j$ each time it calls $\mathcal{O}_{\mathsf{Withdraw}}$. The oracle will not reveal the wallet $W_j$ it obtained to the adversary.

$\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, info)$**.** The oracle runs $\mathsf{SpendCoin}(params, W_j, pk_{\mathcal{M}}.info)$ and returns the resulting $coin$ to the adversary. The oracle outputs error if the adversary has not previously called $\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, i, j)$, or if the adversary has already called $\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, \cdot)$ $n$ times.

$\mathcal{O}_{\mathsf{SimSpend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, info)$. The oracle runs $\mathsf{SimSpend}(params, sim, pk_{\mathcal{B}}, pk_{\mathcal{M}}, info)$ and return the resulting $coin$. The oracle outputs error if the adversary has not previously called $\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, i, j)$, or if the adversary has already called $\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, \cdot)$ $n$ times.

**Balance.** No coalition of users should be able to deposit more e-coins than they collectively withdrew. Formally, for all non-uniform polynomial time $\mathcal{A}$ and every $n < poly(k)$ there exists a negligible function $\nu$ such that

$$\begin{aligned} Pr[&params \leftarrow \mathsf{CashSetup}(1^k); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(params, n); \\ &(withdrawals, deposits) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Withdraw}}(params,\cdot,sk_{\mathcal{B}}),\mathcal{O}_{\mathsf{Deposit}}(params,pk_{\mathcal{B}},\cdot,\cdot,state_{\mathcal{B}})} : \\ &withdrawals < deposits] < \nu(k) \end{aligned}$$

Where $withdrawals$ is the total number of successful calls to $\mathcal{O}_{\mathsf{Withdraw}}$ multiplied by $n$ (i.e. the number of coins withdrawn) and $deposits$ is the total number of successful calls to $\mathcal{O}_{\mathsf{Deposit}}$. Success means that the oracles output accept. We define the oracles as follows:

$\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}})$**.** Runs the Withdraw protocol, where the oracle acts as the bank and the adversary plays the role of the user: $\mathsf{Withdraw}(\mathcal{A}(state), \mathcal{B}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}}))$. $\mathcal{O}_{\mathsf{Withdraw}}$ outputs accept if the protocol outputs accept.

$\mathcal{O}_{\mathsf{Deposit}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, state_{\mathcal{B}})$ Runs the Deposit protocol, where the oracle acts as the bank and the adversary plays the role of a merchant. If this is the first call to $\mathcal{O}_{\mathsf{Deposit}}$, then the oracle sets $state_{\mathcal{B}}$ to $\perp$. Then the oracle updates $state_{\mathcal{B}}$ in the usual way. The oracle outputs accept only if the Deposit protocol outputs accept.

**Identification.** The bank will be able to identify any user who generates two valid e-coins (i.e. e-coins that pass the VerifyCoin test) with the same serial number. Formally, for all non-uniform polynomial time $\mathcal{A}$ and every $n < poly(k)$ there exists a negligible function $\nu$ such that

$$\begin{aligned} Pr[&params \leftarrow \mathsf{CashSetup}(1^k); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(params, n); \\ &(coin_1, coin_2) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Withdraw}}(params,\cdot,sk_{\mathcal{B}})}(params, pk_{\mathcal{B}}) : \\ &coin_1 = (S, \pi_1, pk_{\mathcal{M}1} \| info_1) \wedge coin_2 = (S, \pi_2, pk_{\mathcal{M}2} \| info_2) \\ &\wedge pk_{\mathcal{M}1} || info_1 \neq pk_{\mathcal{M}2} || info_2 \\ &\wedge \mathsf{VerifyCoin}(params, pk_{\mathcal{M}1}, pk_B, coin_1) = \mathsf{accept} \\ &\wedge \mathsf{VerifyCoin}(params, pk_{\mathcal{M}2}, pk_B, coin_2) = \mathsf{accept} \\ &\wedge \mathsf{Identify}(params, coin_1, coin_2) \notin T] < \nu(k) \end{aligned}$$

Oracle $\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}})$ runs protocol Withdraw, with the oracle acting as the bank and the adversary playing the part of the user: $\mathsf{Withdraw}(\mathcal{A}(state), \mathcal{B}(params, pk_{\mathcal{U}}, sk_{\mathcal{B}}))$. The bank adds the resulting $T_W$ to its database $T$.

**Weak Exculpability.** If a user never double-spends, then even a malicious bank colluding with malicious merchants will not be able to frame the honest user, i.e. to produce $coin_1, coin_2$ such that $\mathsf{Identify}(params, coin_1, coin_2) = T_W$ where $T_W$ is the trace information for this user.
Formally, for all non-uniform polynomial time $\mathcal{A}$ and every $n < poly(k)$ there exists a negligible function $\nu$ such that

$$Pr[params \leftarrow \mathsf{CashSetup}(1^k);$$
$$(pk_{\mathcal{B}}, coin_1, coin_2) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{GetKey}}(params,\cdot),\mathcal{O}_{\mathsf{Withdraw}}(params,\cdot,\cdot,\cdot)(params)\mathcal{O}_{\mathsf{Spend}}(params,\cdot,\cdot,\cdot)};$$
$$pk_U \leftarrow \mathsf{Identify}(params, coin_1, coin_2) : pk_U \in \textit{HonestUsers}] < \nu(k)$$

where the oracles are defined as follows:

$\mathcal{O}_{\mathsf{GetKey}}(params, i)$**.** The oracle returns $pk_{\mathcal{U}i}$, the public-key of user $\mathcal{U}_i$. If $pk_{\mathcal{U}i}$ doesn't exist, the oracle generates $(pk_{\mathcal{U}i}, sk_{\mathcal{U}i})$ using $\mathsf{UserKG}(params)$. Finally, the user stores $pk_{\mathcal{U}i}$ on list *HonestUsers*

$\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, i, j)$**.** The oracle runs the Withdraw protocol with the adversary: $\mathsf{Withdraw}(\mathcal{U}(params, pk_{\mathcal{B}}, sk_{\mathcal{U}i}), \mathcal{A}_2(state))$. The adversary plays the role of the bank and the oracle takes the role of user $\mathcal{U}_i$. If $sk_{\mathcal{U}i}$ doesn't exist, the oracle generates it using $\mathsf{UserKG}(params)$ (and stores $pk_{\mathcal{U}i}$ on the list *HonestUsers*). The value $j$ serves to identify the wallet to the oracle for later use; the adversary must use a fresh value $j$ each time it calls $\mathcal{O}_{\mathsf{Withdraw}}$. The oracle will not reveal the wallet $W_j$ it obtained to the adversary.

$\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, info)$**.** The oracle runs $\mathsf{SpendCoin}(params, W_j, pk_{\mathcal{M}}.info)$ and returns the resulting $coin$ to the adversary. The oracle outputs error if the adversary has not previously called $\mathcal{O}_{\mathsf{Withdraw}}(params, pk_{\mathcal{B}}, i, j)$, or if the adversary has already called $\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, pk_{\mathcal{M}}, i, j, \cdot)$ $n$ times.

## F Proof of Security of Our Compact E-Cash Scheme

This proof makes use of the security definitions of P-Signatures as of [BCKL08] and the standard security notions of pseudo-random functions and zero-knowledge proof systems. We refer the reader to [BCKL08] for a definition of the *Signer privacy*, *User privacy*, *Correctness*, *Unforgeability*, and *Zero-knowledge* properties of a P-signature scheme and the corresponding simulatior protocols (SigSimIssue, SigSimObtain,SigSimSetup, SigSimProve) and extraction algorithms (SigExtractSetup, SigExtract) of a P-signature scheme.

**Theorem.** *The e-cash scheme presented in Section 5 is a secure e-cash scheme given the security of the P-signature scheme, the PRF, and the NIZK proof system.*

*Proof.* We need to prove that CashSetup, BankKG, UserKG, SpendCoin, VerifyCoin, Deposit, Identify, and the interactive protocol Withdraw fulfill the *Correctness*, *Anonymity*, *Balance*, and *Identification* properties.

*Correctness.* Correctness is straight forward.

*Anonymity.* Consider the following simulator $\mathsf{Sim} = (\mathsf{SimCashSetup}, \mathsf{SimSpend})$:

$\mathsf{SimCashSetup}(1^k)$. Runs $\mathsf{SigSimSetup}(1^k)$ to obtain $params, sim_P$. Our construction is non-blackbox: we reuse the GS-NIZK proof system parameters $params_{GS}$ that are contained in $params$ and the GS NIZK simulation parameters $sim_{GS}$ contained in $sim_P$. The parameters $params_{GS}$ in turn contain the setup for a bilinear pairing $params_{BM} = (p, G_1, G_2, G_T, e, g, h)$ for a paring $e : G_1 \times G_2 \rightarrow G_T$ for groups of prime order $p$. The algorithm returns $(params, sim_P)$.

$\mathsf{SimSpend}(params, sim, pk_{\mathcal{B}}, pk_{\mathcal{M}}, info)$.

- The simulator uses $\mathsf{SigSimProve}(params, sim_P, pk_w, 3)$ to compute $((C_{id}, C_s, C_t), \pi_1)$
- The simulator uses $\mathsf{SigSimProve}(params, sim_P, pk_c, 1)$ to compute $(C_J, \pi_2)$.

– The simulator picks a random serial number and double spending tag $S, T \leftarrow G_1$ and simulates the non-interactive zero-knowledge proofs $\pi_S$ and $\pi_T$ using the zero-knowledge simulator for $\mathcal{L}_S$ and $\mathcal{L}_T$.

We consider a sequence of 5 Games:

**Game 1.** Corresponds to the game $\mathcal{A}$ plays when interacting with $\mathcal{O}_{\mathsf{Spend}}(params, pk_{\mathcal{B}}, \cdot, \cdot)$.
**Game 2.** As Game 1, except that CashSetup is replaced by SimCashSetup to obtain $sim_P$.
**Game 3.** As Game 2, except that the oracle uses $sim_P$ and SigSimProve to compute $((C_{id}, C_s, C_t), \pi_1)$.
**Game 4.** As Game 3, except that the oracle uses $sim_P$ and SigSimProve to compute $(C_J, \pi_2)$.
**Game 5.** As Game 4, except that the oracle uses $sim_{GS}$ and the zero-knowledge simulator for languages $L_S$ and $L_T$.
**Game 6.** As Game 5, except that $S$ and $T$ are now chosen at random. This corresponds to the game with $\mathcal{O}_{\mathsf{SimSpend}}(params, pk_{\mathcal{B}}, \cdot, \cdot, \cdot)$.

Games 1 and 2 are indistinguishable by the properties of the P-signature scheme.

A non-negligible probability to distinguish between Games 2 and 3 and between Games 3 and 4 allows to break the zero-knowledge property of the P-signature scheme. A non-negligible probability to distinguish between Games 4 and 5 breaks the zero knowledge property of the proof system.

A non-negligible probability to distinguish between Games 5 and 6 allows to break the pseudorandomness of the PRF through the following reduction. The reduction either gets oracle access to two pseudorandom functions $F_s(.)$ and $F_t(.)$ or to two random functions.[13] It can simulate all the rest of the spend without knowing $s, t$ given the simulators. In one case it's Game 5, in the other case it's Game 6. If a distinguisher can distinguish between the two games we can break the pseudorandomness of the PRF.

As Game 6 corresponds to the view generated by the simulator, the success probability of an adversary in breaking the *anonymity* property is bounded by the sum of the distinguishing advantages in the above games. This advantage is negligible.

*Balance.* A successfully deposited coin can be parsed as $coin = (S, (T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T), id_{\mathcal{M}} \| info)$. We consider multiple games.

**Game 1.** The first game is the same as the balance definition with the oracles using the real protocol.
**Game 2.** As Game 1 except that in CashSetup algorithm SigSetup is replaced with SigExtractSetup to obtain $td$.
**Game 3.** As Game 2 except that in $\mathcal{O}_{\mathsf{Deposit}}$ the game checks every deposited coin and uses $td$ and the SigExtract algorithm to extract $y_s$, $y_t$ and $y_{id}$ from $C_{id}, C_s, C_t, \pi_1$ and $y_J$ from $C_J, \pi_2$. It aborts if the triple $(y_s, y_t, y_J)$ already appeared in a previously deposited coin.
**Game 4.** As Game 3 except that it also aborts if the value $y_J$ is not in $\{F(1), \ldots, F(n)\}$.
**Game 5.** As Game 4 except that it abort if the number of deposited coins with different $(y_s, y_t)$ pairs is bigger than $withdrawals$.

Games 1 and 2 are indistinguishable as SigSetup and SigExtractSetup are indistinguishable.

Games 2 and 3 are indistinguishable because Game 3 aborts only with negligible probability. An abort can occur only if one of the $F(y_s)^{-1}$, $F(y_t)^{-1}$, $F(y_J)^{-1}$ does not correspond to the opening of $C_s, C_t, C_J$ in which case we found a forgery for one of the two P-signatures or if we broke the soundness of the proof system used to prove language $\mathcal{L}_S$ and $\mathcal{L}_T$. We guess which of the three options is the case to do a reduction and break the unforgeability of the P-signature scheme or the soundness of the proof system.

A distinguisher between Game 3 and 4 allows to break the unforgeability of the P-signature scheme as the only time Game 4 aborts is when it obtains a signature on a value $F^{-1}(y_J) > n$. As such a $J$ value was never signed we obtain a P-signature forgery.

A distinguisher between Game 4 and 5 allows to break the unforgeability of the P-signature scheme as the number of different $(y_s, y_t)$ pairs with corresponding signatures pairs is greater than the number

[13] A standard hybrid argument can be used to show that $S$ and $T$ can be replaced one after the other.

of correctly generated signatures be $\mathcal{O}_{\mathsf{Withdraw}}$. We create a reduction and guess which $(y_s, y_t)$ is the forgery to break P-signature unforgeability with probability at least $1/(withdrawals + 1)$.

In Game 5, we can bound the number of successful deposits to at most $withdrawals \cdot n$. The success probability of $\mathcal{A}$ is bounded by the sum of the distinguishing probabilities between the Games 1 to 5. This probability is negligible.

*Identification.* A successful adversary $\mathcal{A}$ in the identification game outputs two coins $(coin_1, coin_2)$ that verify and have the same serial number $S$ but different $id_{\mathcal{M}} \| info$. We consider multiple games.

**Game 1.** Is the same as the original security game.
**Game 2.** As Game 1 but in CashSetup algorithm SigSetup is replaced with SigExtractSetup to obtain $td$.
**Game 3.** As Game 2 but the game parses the coins $coin_1 = (S, (T, C_{id}, C_s, C_t, C_J, \pi_1, \pi_2, \pi_S, \pi_T), pk_{\mathcal{M}1}, info_1)$ and $coin_2 = (S, (T', C'_{id}, C'_s, C'_t, C'_J, \pi'_1, \pi'_2, \pi'_S, \pi'_T), id_{\mathcal{M}2} \| info_2)$ obtained from the adversary and uses $td$ and the SigExtract algorithm to extract $y_{id}, y_s, y_t, y_J$ from $(C_{id}, C_s, C_t, \pi_1)$, $(C_J, \pi_2)$ and $y'_{id}, y'_s, y'_t, y'_J$ from $(C'_{id}, C'_s, C'_t, \pi_1), (C_J, \pi_2)$. Game 3 aborts if the values $y_J$ or $y'_J$ are not in $\{F(1), \ldots, F(n)\}$.
**Game 4.** As Game 3 but it also aborts if $y_s \neq y'_s$.
**Game 5.** As Game 4 but it also aborts if $y_J \neq y'_J$.
**Game 6.** As Game 5 but it aborts if $y_t \neq y_t$ or $y_{id} \neq y'_{id}$.

Games 1 and 2 are indistinguishable as SigSetup and SigExtractSetup are indistinguishable.

A distinguisher between Game 2 and 3 allows to break the unforgeability of the P-signature scheme as the only time Game 3 aborts is when it obtains a signature on a value $F^{-1}(y_J) > n$. As such a $J$ value was never signed we obtain a P-signature forgery.

Games 3 and 4 are indistinguishable. The abort in Game 4 can only happens with probability greater than $n^2/p$ ($\sim$ the probability of collision if $S$ is computed by a random function) in one of the following 4 cases: (i. and ii.) one of the $F(y_s)^{-1}$, $F(y'_s)^{-1}$ does not correspond to the opening of $C_s, C'_s$ respectively (in this case we can find a forgery for the P-signatures scheme), iii. we break the soundness of the proof system for language $\mathcal{L}_S$, or iv. we break the pseudorandomness of $F$.

The reduction to the pseudo-randomness works as follows: We have a polynomial sized domain, so an adversary in the pseudorandomness game can compute the output on every element in the domain. By pseudorandomness, this should look like a completely random polynomially sized subset of $Ga$. So the probability that two randomly chosen seeds will produce intersecting ranges should be negligible. Otherwise we can build a reduction which breaks the pseudorandomness property without even seeing the seed: we are given oracle access to the PRF (or a random function). We query it on all points in the domain. Then we choose another random seed and compute that on all points in the domain. If there is an intersection with the first set, we output "pseudo random" otherwise we output "random".

Note that $n < poly(k)$ and thus the Games 3 and 4 are indistinguishable, or otherwise we guess which of the 4 cases mentioned above holds do the appropriate reduction.

Games 4 and 5 are indistinguishable for the same reason except that the probability of a collision for perfectly random functions corresponds to the propability of distinguishing a random function from a random permutation given only $n < poly(k)$ queries.

Games 5 and 6 are indistinguishable as aborts in 6 occur only with negligible probability. In Game 6 we abort if $y_t \neq y_t$ or $y_{id} \neq y'_{id}$. As the seed $s$ is chosen at random, it is highly unlikely that two withdrawn wallets contain the same seed. Consequently $y_t = y_t$ and $y_{id} = y_{id}$ or we break the unforgeability of the P-signature scheme.

In Game 6 the probability of $e((T/T')^{1/(id_{\mathcal{M}1} \| info_1 - id_{\mathcal{M}2} \| info_2)}, h) \notin DB_T$ is bounded by the soundness error of the proof protocol or the probability that $y_{id}$ was never signed by the bank or does not correspond to the commitment $C_{id}$. If the probability of the first is non-negligible we break the soundness of the proof protocol, if the probability of the latter is non-negligible we break the unforteability of our P-signature scheme.

As Games 1 to 6 are computationally indistinguishable, $\mathcal{A}$'s success probability in the real game is also negligible.

*Weak Exculpability.* A successful adversary $\mathcal{A}$ in the weak exculpability game outputs two coins $(coin_1, coin_2)$ that verify and have the same serial number $S$ but different $id_{\mathcal{M}} \| info$.

While $\mathcal{A}$ knows the users public key $pk_{\mathcal{U}} = e(g^{sk_{\mathcal{U}}}, h)$ it is hard to compute $g^{sk_{\mathcal{U}}}$ from $pk_{\mathcal{U}}$ alone without knowing $sk_{\mathcal{U}}$. As no additional information about $g^{sk_{\mathcal{U}}}$ is revealed until $\mathcal{U}$ reuses an e-token, an adversary computing $g^{sk_{\mathcal{U}}}$ can be used to asymmetric computation Diffie Hellman (asymmetric CDH) assumption: given random $g^a$, $h^b$ compute $g^{ab}$. Asymmetric CDH is implied by q-BDDH and DLIN.

More formally we define a sequence of games to eliminate all sources of information that an adversary may potentially have about honest user's keys besides the public key learned through a $\mathcal{O}_{\mathsf{GetKey}}$ query:

**Game 1.** Here the adversary plays the same game as in the weak exculpability definition.
**Game 2.** As Game 1, but CashSetup is replaced with SimCashSetup.
**Game 3.** As Game 2, but in $\mathcal{O}_{\mathsf{Withdraw}}$ algorithm SigObtain is replaced with the P-signature simulator SigSimObtain.
**Game 4.** As Game 3, but in $\mathcal{O}_{\mathsf{Spend}}$ algorithm SpendCoin is replaced with SimSpend.

Games 1 and 2 are indistinguishable by the properties of the P-signature scheme (and the anonymity of the e-cash scheme itself). If Games 2 and 3 can be distinguish we break the user privacy of the P-signature scheme. Games 3 and 4 are indistinguishable based on the anonymity of the e-cash scheme.

In order to break the asymmetric CDH assumption we do the following reduction. We answer a random $\mathcal{O}_{\mathsf{GetKey}}$ with $e(g^a, h^b)$. Then we use the simulators SimCashSetup, SigSimObtain, and SimSpend to simulate all interactions with the adversary where this user is involved. A successful adversary outputs $g^{ab}$ as $(T/T')^{1/(id_{\mathcal{M}1}\|info_1 - id_{\mathcal{M}2}\|info_2)}$.

## G Zero-Knowledge Proof of Equality of Committed Exponents

Here we review the construction of Zero-Knowledge proof for equality of two commitments described in [BCKL08], which relies heavily on GS proof techniques.

Groth and Sahai provide some useful tools for helping prove that particular GS proofs are zero-knowledge. Since GS proofs are witness indistinguishable, all a simulator has to do is come up with some witness for the equations. Witness indistinguishability guarantees that it is distributed identically to real witnesses. Groth and Sahai construct a $\mathsf{GSSimSetup}(params)$ function that outputs ${params}_{GS}{}'$ that are (1) computationally indistinguishable from the output of $\mathsf{GSSetup}(params)$ and (2) allow us to open $C = \mathsf{GSCom}(params_i', b^\theta, open)$ to any $b^{\theta'}$ as long as we know $b \in G_i, \theta, \theta', open$. If a GS proof contains multiple pairing product equations, we can open $C$ in a different way for each equation. Thus, we can have different witnesses for each equation. (This does not work for value $C' = \mathsf{GSCom}(params_i, x, open)$) for $x \in G_i$.

Suppose we know $c_1 = \mathsf{GSCom}(params_1, \alpha)$ and $c_2 = \mathsf{GSCom}(params_1, \alpha)$ as well as the opening information to $c_1$ and $c_2$.[14] We want to prove the statement

$$\mathsf{NIPK}\{((c_1 : \alpha), (c_2 : \beta)) : \alpha = \beta\}.$$

We calculate $d = \mathsf{GSCom}(params_2, h^1)$. Then we construct the proof

$$\pi \leftarrow \mathsf{NIPK}\{((c_1 : a), (c_2 : b), (d : h^\theta)) : e(a/b, h^\theta) = 1 \wedge e(g, h^\theta)e(1/g, h) = 1\}.$$

[14] Proofs for $G_2$ are done analogously.

*Composable Zero Knowledge.* We need to construct $\mathsf{Sim} = (\mathsf{SimSetup}, \mathsf{SimProve})$. We use the $\mathsf{GSSimSetup}$ algorithm provided by Groth and Sahai that outputs a trapdoor that allows us to open $\mathsf{GSCom}(params_i, b^\theta, open)$ any way we want, as long as we know $b \in G_i, \theta, open$ (see above). We can open it to different values of $\theta$ in each pairing product equation.

The simulator gets as input $c_1$ and $c_2$. All the simulator needs to do is construct a witness for the individual equations of the proof

$$\pi \leftarrow \mathsf{NIPK}\{((c_1 : a), (c_2 : b), (d : h^\theta)) : e(a/b, h^\theta) = 1 \wedge e(g, h^\theta)e(1/g, h) = 1\}.$$

It sets $\theta = 0$ and computes $d = \mathsf{GSCom}(params_2, h^0, open)$. Thus, we satisfy the pairing product equation $e(a/b, h^\theta) = 1$ because $h^\theta = 1$. To satisfy the second pairing product equation, we open $d$ to $\theta = 1$. Thus, we satisfy $e(g, h^\theta)e(1/g, h) = 1$. As a result, the simulator has a witness for the proof. By witness indistinguishability, the simulated witness is indistinguishable from real witnesses. Thus we get zero-knowledge.

*NIZK proof that a commitment commits to a given value* Note that we can use the above technique to prove that a commitment $C = \mathsf{GSCom}(x, open_x)$ for some public value $x$. This is done by forming a second commitment $C' = \mathsf{GSCom}(x, open_{public})$ using some public auxiliary information $open_{public}$ (so the verify can compute $C'$ independently and very that it matches the value that the prover uses). Then the above proof system can be used to show that $C$ and $C'$ commit to the same value.