Divisible E-Cash in the Standard Model

Malika Izabachène[1] and Benoît Libert[2] ⋆

[1] Ecole Normale Supérieure de Cachan/CNRS/INRIA (France)
[2] Université catholique de Louvain (Belgium)

Abstract. Off-line e-cash systems are the digital analogue of regular cash. One of the main desirable properties is anonymity: spending a coin should not reveal the identity of the spender and, at the same time, users should not be able to double-spend coins without being detected. Compact e-cash systems make it possible to store a wallet of $O(2^L)$ coins using $O(L+\lambda)$ bits, where λ is the security parameter. They are called *divisible* whenever the user has the flexibility of spending an amount of 2^ℓ, for some $\ell \leq L$, more efficiently than by repeatedly spending individual coins. This paper presents the first construction of divisible e-cash in the standard model (*i.e.*, without the random oracle heuristic). The scheme allows a user to obtain a wallet of 2^L coins by running a withdrawal protocol with the bank. Our construction is built on the traditional binary tree approach, where the wallet is organized in such a way that the monetary value of a coin depends on how deep the coin is in the tree.

Keywords. E-Cash, provable security, anonymity, non-interactive proofs.

1 Introduction

Introduced by Chaum [22, 23] and developed in [24, 20, 25, 40, 29], electronic cash is the digital equivalent of regular money. It allows a user to withdraw a wallet of electronic coins from a bank so that e-coins can be spent to merchants who can then deposit them back to the bank.

The withdrawal, spending and deposit protocols should be designed in such a way that it is infeasible to determine when a particular coin was spent: even if the bank colludes with the merchant, after the deposit protocol, it should be unable to link a received coin to a particular withdrawal protocol. At the same time, users should not be able to covertly double-spend coins: should a cheating user attempt to spend a given coin twice, his identity must be exposed and evidence of his misbehavior must be given. Ideally, dishonest users should be identified without the help of a trusted third party and, as in the off-line scenario [24], the bank should preferably not intervene in the spending protocol between the user and the merchant.

Related Work. In 2005, Camenisch, Hohenberger and Lysyanskaya [10] described a *compact* e-cash system allowing a user to withdraw a wallet of 2^L coins with a computational cost of $O(L+\lambda)$, where λ is the security parameter, in the spending and withdrawal protocols. Using appropriate choices [27, 28] of verifiable random functions [34], they also showed how to store a wallet using only $O(L+\lambda)$ bits and additionally described a coin tracing mechanism allowing to trace all the coins of a misbehaving user. The protocol of Camenisch *et al.* was subsequently extended into e-cash systems with coin endorsement [13] or transferability properties [15, 16].

The aforementioned e-cash realizations all appeal to the random oracle model [6] – at least if the amount of interaction is to be minimized in the spending protocol – which is known not to accurately reflect world (see [19] for instance). To fill this gap, Belenkiy, Chase, Kohlweiss and Lysyanskaya [5] described a compact e-cash system with non-interactive spending in the standard model. Their construction cleverly combines multi-block extensions of P-signatures [3] and simulatable verifiable

⋆ This author acknowledges the Belgian Fund for Scientific Research for his "Collaborateur scientifique" fellowship.

random functions [21] with the Groth-Sahai non-interactive proof systems [31]. Independently, Fuchsbauer, Pointcheval and Vergnaud also used Groth-Sahai proofs [30] to build a transferable fair (*i.e.*, involving a semi-trusted third party) e-cash system in the standard model. More recently, Blazy *et al.* [7] gave a similar construction with stronger anonymity properties.

Divisible E-Cash. In the constructions of [10], users have to run the spending protocol N times if the amount to be paid is the equivalent of N coins. One possible improvement is to use wallets containing coins of distinct monetary values as in [17]. Unfortunately, this approach does not allow to split individual coins of large value. This problem is addressed by divisible e-cash systems where users can withdraw a coin of value 2^L that can be spent in several times by dividing the value of that coin. Divisible e-cash makes it possible for users to spend the equivalent of $N = 2^\ell$ (with $0 \leq \ell \leq L$) coins more efficiently than by iterating the spending protocol 2^ℓ times. Constructions of divisible e-cash were proposed in the 90's [36, 38, 26, 37, 20]. Okamoto provided a practical realization [37] that was subsequently improved in [20]. Unfortunately, these schemes are not fully unlinkable since several spends of a given divisible coin can be linked. To address this concern, Nakanishi and Sugiyama [35] described an unlinkable divisible e-cash system but their scheme requires a trusted third party to uncover the identity of double-spenders. In addition, by colluding with the bank, the merchant can obtain information on which part of the divisible coin the user is spending.

In 2007, Canard and Gouget [14] designed the first divisible e-cash system with full anonymity (and unlinkability) where misbehaving users can be identified without involving a trusted third party. Later on, Au *et al.* [1] came up with a more efficient implementation at the expense of substantially weaker security guarantees. More recently, Canard and Gouget [18] showed how to improve upon the efficiency of their original proposal without sacrificing the original security.

Our Contribution. Prior implementations of truly anonymous divisible e-cash all require the random oracle idealization in their security analysis. In this paper, we describe the first anonymous divisible e-cash in the standard model. Like the scheme of Belenkiy *et al.* [5], our construction relies on the Groth-Sahai non-interactive proof systems [31].

Our scheme is less efficient than the fastest random-oracle-based scheme [18] in several metrics. While the spending phase has constant (*i.e.*, independent of the value 2^L of the wallet) communication complexity in [18], our spending protocol requires users to transmit $O(L)$ group elements to the merchant in the worst case. On the other hand, due to the use of bounded accumulators [2], the bank has to set up a public key of size $O(2^L)$ in [18][3] whereas we only need the bank to have a key of size $O(1)$.

Achieving divisibility without resorting to random oracles requires to solve several technical issues. The solutions of Canard and Gouget [14, 18] associate each wallet with a binary tree – where nodes correspond to expandable amounts – combined with cyclic groups of *distinct* but related orders. Since these techniques do not appear compatible with the Groth-Sahai toolbox, we had to find other techniques to split the wallet across the nodes of a binary tree. In particular, we use a different method to authenticate the node corresponding to the spent divided coin in the tree.

As in the first truly anonymous construction of divisible e-cash [14], the communication complexity of our spending algorithm depends on how much the initial amount 2^L has to be divided: from an initial tree of value 2^L, when a coin of value 2^ℓ has to be spent, the communication cost of the spending phase is $O(L - \ell)$. Hence, the more we want to divide the wallet into small coins, the

[3] The reason is that, in all known bounded accumulators, the public key has linear size in the maximal number of accumulated values.

more expensive the spending phase is.

The downside of our e-cash construction is the complexity of the deposit phase, where the computational workload of the bank depends on the number of previously received coins when it comes to check that the received coin does not constitute a double-spending. Even though the bank can be expected to have significant computational resources (and although the double-spending checks can be performed in parallel by clerks testing a subset of previously spent coins each), this would be a real bottleneck in practice. For this reason, our system is not meant to be a practical solution and should only be seen as a feasibility result. We leave it as an open problem to build such a system with a more efficient deposit procedure from the bank's standpoint: as in previous constructions of compact e-cash (e.g. [10, 5]), the bank should only have to look up the coin's serial number in a table of previously spent coins. It would also be interesting to reduce the communication complexity of the spending phase so as to only transmit a constant number of group elements.

2 Background and Definitions

2.1 Definitions for Divisible E-Cash

An e-cash scheme involves a bank $\mathcal{B}$, many users $\mathcal{U}$ and merchants $\mathcal{M}$ (who can be viewed as special users). All these parties interact together with respect to the following protocols:

CashSetup(λ)**:** takes as input a security parameter λ and outputs the public parameters params.

BankKG(params, L)**:** generates bank's public and secret parameters $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ that allow $\mathcal{B}$ to issue wallets of up values up to 2^L (we assume that L is part of $pk_{\mathcal{B}}$). It also defines an empty database $\mathrm{DB}_{\mathcal{B}}$ for later use.

UserKG(params)**:** generates a user key pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$. We denote as $\mathcal{H}_{\mathcal{U}}$ the set of honestly generated public keys.

Withdraw$\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$**:** is an interactive protocol between a user $\mathcal{U}$ and the bank $\mathcal{B}$ that allows an honest user to obtain a coin of value 2^L. The wallet $\mathcal{W}$ comprises the coins, the user's secret key, a signature from the bank on it and some state information state. The bank debits $\mathcal{U}$'s account and stores a piece of tracing information $\mathsf{T}_{\mathcal{W}}$ in a database T that allows uncovering the identity of double-spenders.

Spend$\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W}, 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$**:** is invoked by $\mathcal{U}$ to spend a coin of value 2^ℓ from his wallet and generates a proof Π that the coin is valid. The output is the coin that includes the proof Π and some fresh public information $\mathtt{info}$ specifying the transaction.

VerifyCoin$\big(\mathsf{params}, pk_{\mathcal{M}}, pk_{\mathcal{B}}, coin, v = 2^\ell\big)$**:** allows $\mathcal{M}$ to verify the validity of the received coin and output a bit depending on whether the test is successful.

Deposit$\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, 2^\ell, \mathrm{DB}_{\mathcal{B}}\big)$**:** $\mathcal{B}$ updates the database $\mathrm{DB}_{\mathcal{B}}$ with $\{(coin, \mathsf{flag}, 2^\ell)\}$ where flag indicates whether *coin* is a valid coin of value 2^ℓ and whether a cheating attempt is detected.

- If *coin* does not verify, $\mathcal{B}$ rejects it and sets flag = "$\mathcal{M}$" to indicate a cheating merchant.
- If *coin* verifies, $\mathcal{B}$ runs a double spending detection algorithm, using the database $\mathrm{DB}_{\mathcal{B}}$ containing already received coins. If an overspent is detected, the bank sets flag = "$\mathcal{U}$", outputs the two coins and reports the double-spending.
- If the coin passes all the tests, the bank accepts the coin, sets flag = "accept" and credits the merchant's account.

Identify$\big(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b\big)$**:** the bank retrieves the double-spender's public key $pk_{\mathcal{U}}$ using its database $\mathrm{DB}_{\mathcal{B}}$ and the two different coins.

The security model builds on the model of non-interactive compact e-cash from [5]. An e-cash scheme is secure if it provides *Correctness, Anonymity, Balance, Identification* and *Exculpability* simultaneously.

ANONYMITY. Unlike the model of [14], ours adopts a simulation-based formulation of anonymity (note that simulation-based definitions are often stronger than indistinguishability-based ones). No coalition of banks and merchants should distinguish a real execution of the Spend protocol from a simulated one. In the security experiment, the adversary is allowed to obtain users' public keys, withdraw and spend coins using the oracles $\mathcal{Q}_{\mathsf{GetKey}}, \mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$, respectively, which are defined below. Formally, an e-cash system is *anonymous* if there exists a simulator $(\mathsf{SimCashSetup}, \mathsf{SimSpend})$ such that, for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ such that:

$$
\begin{aligned}
| \Pr[\, & \mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}) : \\
& \mathcal{A}_2^{\mathcal{Q}_{\mathsf{Spend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot)}(\mathsf{state}) = 1] \\
- \Pr[& (\mathsf{params}, \mathsf{Sim}) \leftarrow \mathsf{SimCashSetup}(\lambda); (pk_{\mathcal{B}}, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}) : \\
& \mathcal{A}_2^{\mathcal{Q}_{\mathsf{SimSpend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot)}(\mathsf{state}) = 1] \mid < \mathsf{negl}(\lambda)
\end{aligned}
$$

To formalize security against coalition of users, bank and merchants, the anonymity game allows the adversary to generate the bank's public key. It is granted dynamic access to the list of oracles hereafter and has to decide whether it is playing the real game, where the spending oracle is an actual oracle, or the simulation, where the spending oracle is a simulator.

- $\mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, j)$: outputs $pk_{\mathcal{U}_j}$. If $\mathcal{U}_j$ does not exist, the oracle generates $(sk_{\mathcal{U}_j}, pk_{\mathcal{U}_j}) \leftarrow \mathsf{UserKG}(\mathsf{params})$ and outputs $pk_{\mathcal{U}_j}$.
- $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, j, f)$: given a wallet identifier f, this oracle plays the role of user j – and creates the key pair $(sk_{\mathcal{U}_j}, pk_{\mathcal{U}_j})$ if it does not exist yet – in an execution of the withdrawal protocol $\mathsf{Withdraw}\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}_j}), \mathcal{A}(\mathtt{states})\big)$, while the adversary $\mathcal{A}$ plays the role of the bank. The oracle then creates a wallet $\mathcal{W}_f$ of value 2^i.
- $\mathcal{Q}_{\mathsf{Spend}}\big(\mathsf{params}, pk_{\mathcal{B}}, f, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$: the oracle firstly checks if wallet $\mathcal{W}_f$ has been created via an invocation of $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{B}}, j, f)$. If not, the oracle outputs $\perp$. Otherwise, if $\mathcal{W}_f$ contains a sufficient amount, $\mathcal{Q}_{\mathsf{Spend}}$ runs $\mathsf{Spend}\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W}_f, i, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$ and outputs a coin of value v from the wallet $\mathcal{W}_f$. In any other case (e.g. if the expandable amount of $\mathcal{W}_f$ is less than 2^ℓ), it outputs $\perp$.
- $\mathcal{Q}_{\mathsf{SimSpend}}\big(\mathsf{params}, pk_{\mathcal{B}}, f, v = 2^\ell, pk_{\mathcal{M}}, \mathtt{info}\big)$: if f is not the index of a valid withdrawn wallet obtained from $\mathcal{Q}_{\mathsf{withdraw}}$, the oracle outputs $\perp$. Otherwise, the oracle runs a simulator $\mathsf{SimSpend}$ on input $\big(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, v = 2^\ell, \mathtt{info}\big)$. Note that $\mathsf{SimSpend}$ does not use the user's wallet or his public key.

BALANCE. No coalition of users can spend more coins than they withdrew. The adversary is a user and can withdraw or spend coins via oracles defined below. An e-cash system provides the *Balance* property if, for any adversary, every value $L \in \mathsf{poly}(\lambda)$, we have:

$$
\begin{aligned}
\Pr[\, & \mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(\mathsf{params}, L); \\
& (q_w, n_d) \leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, pk_{\mathcal{B}}, \cdot)(), \mathcal{Q}_{\mathsf{deposit}}(\mathsf{params}, pk_{\mathcal{B}}, \mathrm{DB}_{\mathcal{B}})} : q_w \cdot 2^L < n_d] < \mathsf{negl}(\lambda),
\end{aligned}
$$

where n_d is the total amount of deposited money after q_d successful calls to oracle $\mathcal{Q}_{\mathsf{deposit}}$ (by successful, we mean that the oracle sets $\mathsf{flag} =$ "accept"), q_w is the number of successful calls to Q_{Withdraw}.

– $\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})$: the oracle plays the role of the bank in an execution of the Withdraw protocol, on input $\big(\mathcal{A}(\mathtt{states}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$, in interaction with the adversary acting as a cheating user. At the end of the protocol, $\mathcal{Q}_{\mathsf{withdraw}}$ stores a piece of tracing information $\mathsf{T}_{\mathcal{W}}$ in a database T.
– $\mathcal{Q}_{\mathsf{deposit}}(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, v, \mathrm{DB}_{\mathcal{B}})$: the oracle plays the role of the bank and the adversary plays the role of the merchant. The oracle initializes the bank database $\mathrm{DB}_{\mathcal{B}}$ at $\emptyset$ and returns the same response as $\mathsf{Deposit}(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, v, \mathrm{DB}_{\mathcal{B}})$.

Identification. Given two fraudulent but well-formed coins, the bank should identify the double-spender. This property is defined via an experiment where the adversary $\mathcal{A}$ is the double-spender and has access to a $\mathcal{Q}_{\mathsf{withdraw}}$ oracle defined hereafter. Its goal is to deposit a coin twice without being identified by the bank. We denote by $coin_a$ and $coin_b$ the two coins produced by $\mathcal{A}$. Their corresponding entries in $\mathrm{DB}_{\mathcal{B}}$ are of the form $(coin_a, \mathsf{flag}_a, v_a, pk_{\mathcal{M}_a})$ and $(coin_b, \mathsf{flag}_b, v_b, pk_{\mathcal{M}_b})$, respectively, with $coin_a = (\mathtt{info}_a; *)$ and $coin_b = (\mathtt{info}_b; *)$. We also define a predicate SameCoin that given two coins $coin_a$ and $coin_b$ and their respective values v_a and v_b, outputs 1 if the bank detects a double-spending during the deposit of $coin_a$ and $coin_b$: in the context of divisible e-cash, it means that either $coin_a$ and $coin_b$ are the same coin or that one of them, say $coin_a$, is the result of dividing the other one (and thus v_a divides v_b). The adversary is successful if its coins satisfy $\mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1$ but the bank fails to identify the user using the database T of tracing pieces of information that were collected during executions of $\mathcal{Q}_{\mathsf{withdraw}}$. An e-cash scheme provides double-spenders identification if for any adversary $\mathcal{A}$ and any $L \in \mathsf{poly}(\lambda)$,

$$\begin{aligned}
\Pr\big[&\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda); (pk_{\mathcal{B}}, sk_{\mathcal{B}}) \leftarrow \mathsf{BankKG}(\mathsf{params}, L);\\
&\big((coin_a, v_a), (coin_b, v_b)\big) \leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, sk_{\mathcal{B}}, \cdot)}(\mathsf{params}, pk_{\mathcal{B}}) :\\
&(\mathtt{info}_a; pk_{\mathcal{M}_a}) \neq (\mathtt{info}_b; pk_{\mathcal{M}_b}) \wedge\ \mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1\\
&\wedge\ \mathsf{VerifyCoin}(\mathsf{params}, pk_{\mathcal{M}_t}, pk_{\mathcal{B}}, coin_t, v_t) = 1 \text{ for } t \in \{a, b\}\\
&\wedge\ \mathsf{Identify}(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b) \notin \mathsf{T}\big] < \mathsf{negl}(\lambda)
\end{aligned}$$

The oracle $\mathcal{Q}_{\mathsf{withdraw}}$ has the same specification as in the Balance property.

Exculpability. No coalition of bank and merchants interacting with an honest user $\mathcal{U}$ should be able to produce two coins $(coin_a, coin_b)$ such that $\mathsf{Identify}(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b) = pk_{\mathcal{U}}$ while user $\mathcal{U}$ never double-spent. More formally, we define a game with the challenger playing the role of an honest user and the adversary playing the role of the bank and merchants. The adversary $\mathcal{A}$ is given access to oracles $\mathcal{Q}_{\mathsf{GetKey}}, \mathcal{Q}_{\mathsf{withdraw}}, \mathcal{Q}_{\mathsf{Spend}}$ that allow it to obtain users' keys, create wallets and spend coins. The exculpability property holds if, for any PPT adversary $\mathcal{A}$, we have

$$\begin{aligned}
\Pr\big[&\mathsf{params} \leftarrow \mathsf{CashSetup}(\lambda);\ (pk_{\mathcal{B}}, st) \leftarrow \mathcal{A}(\mathsf{params});\\
&(pk_{\mathcal{B}}, coin_a, coin_b, v_a, v_b) \leftarrow \mathcal{A}^{\mathcal{Q}_{\mathsf{Spend}}(\mathsf{params}, pk_{\mathcal{B}}, \cdot, \cdot), \mathcal{Q}_{\mathsf{GetKey}}(\mathsf{params}, \cdot), \mathcal{Q}_{\mathsf{withdraw}}(\mathsf{params}, \cdot, \cdot, \cdot)}(\mathsf{params}, st);\\
&\mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1\\
&pk_{\mathcal{U}} \leftarrow \mathsf{Identify}(\mathsf{params}, coin_a, coin_b) : pk_{\mathcal{U}} \in \mathcal{H}_{\mathcal{U}}\big] < \mathsf{negl}(\lambda),
\end{aligned}$$

where $\mathcal{H}_{\mathcal{U}}$ denotes the set of honest users. Oracles $\mathcal{Q}_{\mathsf{GetKey}}$, $\mathcal{Q}_{\mathsf{withdraw}}$ and $\mathcal{Q}_{\mathsf{Spend}}$ are defined exactly as in the notion of anonymity.

2.2 F-Unforgeable Signatures

Since the e-cash construction described in the paper relies on a common reference string, the following algorithms all take as input a set of common public parameters $\mathsf{params}_{\mathsf{GS}}$. To lighten notations, we omit to explicitly write them in the syntax hereafter.

Definition 1. *A multi-block signature scheme consists of efficient algorithms* $\Sigma = (\mathsf{SigSetup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following specification.*

SigSetup(λ)**:** *takes as input a security parameter* $\lambda \in \mathbb{N}$ *and outputs* params *that gives the length* $n \in \mathsf{poly}(\lambda)$ *of message vectors to be signed.*

Keygen(params)**:** *takes as input the public parameters and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.

Sign$(\mathsf{sk}, \vec{m})$**:** *is a (possibly randomized) algorithm that takes as input a private key* sk *and a vector* $\vec{m} = (m_1, \ldots, m_n)$ *of messages. It outputs a signature* σ.

Verify$(\mathsf{pk}, \vec{m}, \sigma)$**:** *is a deterministic algorithm that takes as input a public key* pk*, a signature* σ *and a message vector* $\vec{m} = (m_1, \ldots, m_n)$*. It outputs* 1 *if* σ *is deemed valid for* $\vec{m}$ *and* 0 *otherwise.*

Definition 2 ([5]). *A multi-block signature scheme* Σ *is F-unforgeable, for some injective function* $F(.)$*, if no probabilistic polynomial time (PPT) adversary has non-negligible advantage in the following game:*

1. *The challenger runs* Setup *and* Keygen *to obtain a pair* $(\mathsf{pk}, \mathsf{sk})$*, it then sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *adaptively queries a signing oracle. At each query* i*,* $\mathcal{A}$ *chooses a vector* $\vec{m} = (m_1, \ldots, m_n)$ *and obtains* $\sigma_i = \mathsf{Sign}(\mathsf{sk}, \vec{m})$.
3. $\mathcal{A}$ *outputs a pair* $\big((F(m_1^\star), \ldots, F(m_n^\star)), \sigma^\star\big)$ *and wins if: (a)* $\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1$*; (b)* $\mathcal{A}$ *did not obtain any signature on the vector* $(m_1^\star, \ldots, m_n^\star)$.

Definition 3 ([5]). *A multi-block P-signature combines an F-unforgeable multi-block signature scheme* Σ *with a commitment scheme* $(\mathsf{Com}, \mathsf{Open})$ *and three protocols:*

1. *An algorithm* $\mathsf{SigProve}(\mathsf{params}, \mathsf{pk}, \sigma, \vec{m} = (m_1, \ldots, m_n))$ *that generates a series of* n *commitments* C_σ, $C_{m_1}, \ldots, C_{m_n}$ *and a NIZK proof*

   $$\pi \leftarrow \mathsf{NIZPK}\big(m_1 \text{ in } C_{m_1}, \ldots, m_n \text{ in } C_{m_n},\ \sigma \text{ in } C_\sigma \mid \{(F(m_1), \ldots, F(m_n), \sigma) : \mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) = 1\}\big)$$

   and the corresponding $\mathsf{VerifyProof}(\mathsf{params}, \mathsf{pk}, C_{m_1}, \ldots, C_{m_n}, C_\sigma)$ *algorithm.*
2. *A NIZK proof that two commitments open to the same value, i.e., a proof for the relation*

   $$R = \{((x, y), (open_x, open_y)) \mid C = \mathsf{Com}(x, open_x) \ \wedge \ D = \mathsf{Com}(y, open_y) \ \wedge \ x = y\}.$$

3. *A protocol* $\mathsf{SigIssue} \rightleftarrows \mathsf{SigObtain}$ *allowing a user to obtain a signature on a committed vector* $\vec{m} = (m_1, \ldots, m_n)$ *without letting the signer learn any information on* $\vec{m}$.

2.3 Bilinear Maps and Complexity Assumptions

We consider a configuration of *asymmetric* bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p with a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that: (1) $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}$; (2) $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g \neq 1_{\mathbb{G}_1}$ and $h \neq 1_{\mathbb{G}_2}$. Since we rely on the hardness of DDH in $\mathbb{G}_1$ and $\mathbb{G}_2$, we additionally require that no isomorphism be efficiently computable between $\mathbb{G}_2$ and $\mathbb{G}_1$.

Definition 4. *The* q**-Hidden Strong Diffie-Hellman problem** *(q-HSDH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *consists in, given* $(g, u, h, \Omega = h^{\omega}) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ *and tuples* $(g^{1/(\omega+c_i)}, g^{c_i}, h^{c_i}, u^{c_i})$ *with* $c_1, \ldots, c_q \xleftarrow{R} \mathbb{Z}_p^*$*, finding* $(g^{1/(\omega+c)}, h^c, u^c)$ *such that* $c \neq c_i$ *for* $i = 1, \ldots, q$*.*

Definition 5. *The* q**-Decision Diffie-Hellman Inversion problem** *(q-DDHI) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *consists in, given* $(g, g^{(\alpha)}, \ldots, g^{(\alpha^q)}) \in \mathbb{G}_1^{q+1}$ *and* $\eta \in \mathbb{G}_1$*, deciding if* $\eta = g^{1/\alpha}$ *or* $\eta \in_R \mathbb{G}_1$*.*

Definition 6 ([3]). *The* **Triple Diffie-Hellman problem** *(TDH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is, given a tuple* $(g, g^a, g^b, h, h^a) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$*, and* $(c_i, g^{1/a+c_i})_{i=1,\ldots,q}$ *where* $a, b \xleftarrow{R} \mathbb{Z}_p^*$*, to find a triple* $(g^{\mu b}, h^{\mu a}, g^{\mu ab})$ *such that* $\mu \neq 0$*.*

Definition 7. *The* **Decision 3-party Diffie-Hellman problem** *(D3DH) in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is, given* $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, \Gamma) \in \mathbb{G}_1^4 \times \mathbb{G}_2^4 \times \mathbb{G}_1$*, where* $a, b, c \xleftarrow{R} \mathbb{Z}_p$*, to decide if* $\Gamma = g^{abc}$ *or* $\Gamma \in_R \mathbb{G}_1$*.*

2.4 Building Blocks

Non-interactive witness indistinguishable proofs. Our construction uses Groth-Sahai proofs for pairing product equations (PPE) of the form:

$$\prod_{j=1}^{n} e(\mathcal{A}_j, \mathcal{Y}_j) \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{B}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{Y}_j)^{\gamma_{i,j}} = t_T,$$

where $\mathcal{X}_i, \mathcal{Y}_j$ are variables in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $\mathcal{A}_j \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2$ and $t_T \in \mathbb{G}_T$ are constants for $i \in [1, m]$ and $j \in [1, n]$.

A proof system is a tuple of four algorithms $(\mathsf{Setup}_{\mathsf{GS}}, \mathsf{Prove}_{\mathsf{GS}}, \mathsf{VerifyProof}_{\mathsf{GS}})$: $\mathsf{Setup}_{\mathsf{GS}}$ outputs a common reference string (CRS) crs, $\mathsf{Prove}_{\mathsf{GS}}$ first generates commitments of variables and constructs proofs that these variables satisfy the statement, and $\mathsf{VerifyProof}_{\mathsf{GS}}$ verifies the proof. GS proofs are witness-indistinguishable and some of these can be made zero-knowledge as shown later. The proofs satisfy correctness, soundness and witness-indistinguishability. *Correctness* requires that a verifier always accepts honestly generated proofs for true statements. *Soundness* guarantees that cheating provers can only prove true statements. *Witness-indistinguishability* requires that an efficient simulator $\mathsf{GSSimSetup}$ should be able to produce a common reference string (CRS) crs' that is computationally indistinguishable from a normal crs. When commitments are computed using crs', they are perfectly hiding and the corresponding non-interactive proofs are witness indistinguishable: *i.e.*, they leak no information on the underlying witnesses. *Zero-knowledge* additionally requires the existence of an algorithm $\mathsf{GSSimProve}$ that, given a simulated CRS crs' and some trapdoor information τ, generates a simulated proof of the statement without using the witnesses and in such a way that the proof is indistinguishable from a real proof.

As a building block, we will use a NIZK proof of equality of committed group elements as defined in [3, 5].

If $C_x = \mathsf{GSCom}(x, open_x)$ and $C_y = \mathsf{GSCom}(y, open_y)$ are Groth-Sahai commitments to the group element $x = y \in \mathbb{G}_1$, the NIZK proof can be a proof that committed values $(x, y, \theta) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$ satisfy $e(x/y, h^{\theta}) = 1$ and $e(g, h^{\theta})e(1/g, h) = 1_{\mathbb{G}_T}$. Using the trapdoor of the CRS, we can trapdoor open to 1 a commitment to 0 and generate fake proofs for the latter relation. Setting $\theta = 0$ and $\theta = 1$, respectively, allows to construct a valid (simulated) witness for each of the two equations. Under the SXDH assumption, commitments cost 2 elements in the group. Thus, if the commitment to y is in $\mathbb{G}_1^2$, the proof above costs 8 elements $\mathbb{G}_1$ and 6 in $\mathbb{G}_2$, 6 multi-exponentiations and 26 pairings (to verify). This includes commitments to $y \in \mathbb{Z}_p$ and h^{θ}.

Multi Block P-signatures. In [5], a multi-block P-signature was proved F-secure under the HSDH and the TDH assumptions. Let $(p, \mathbb{G}_1, \mathbb{G}_2, G_T, e, g, h)$ be parameters for a bilinear map, the public parameters are then defined as $(p, \mathbb{G}_1, \mathbb{G}_2, G_T, e, g, h, \mathsf{params}_{\mathsf{GS}}, e(g,h))$, where g and h are random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The public key and the private key are defined as $\mathsf{pk} = (u, U = g^{\beta}, \tilde{U} = h^{\beta}, \{V_i = g^{a_i}, \tilde{V}_i = h^{a_i}\}_{i=1}^{n})$ and $\mathsf{sk} = (\beta, \vec{a} = (a_1, \ldots, a_n))$, where $u \xleftarrow{R} \mathbb{G}_1$ and for random scalars $\beta, a_1, \ldots, a_n$. To sign a vector of message $\vec{m} = (m_1, \ldots, m_n)$, the signer chooses a random scalar r such that $r \neq -(\beta + \sum_{i=1}^{n} a_i m_i)$ and computes $\sigma = (g^{1/\beta + r + \sum_{i=1}^{n} a_i m_i}, h^r, u^r)$. Verification of a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ on some block $\vec{m}$ is done by checking whether

$$e\big(\sigma_1, \tilde{U} \cdot \sigma_2 \cdot \prod_{i=1}^{n} \tilde{V}_i^{m_i}\big) = e(g,h) \qquad \text{and} \qquad e(u, \sigma_2) = e(\sigma_3, h).$$

As shown in [5], the above scheme can be augmented with the following P-signature protocols.

SigProve$(\mathsf{params}, \mathsf{pk}, \sigma, \vec{m})$**:** parse the signature σ as $(\sigma_1, \sigma_2, \sigma_3)$ and the vector $\vec{m}$ as $(m_1, \ldots, m_n)$. To commit to an exponent $m_i \in \mathbb{Z}_p$, compute Groth-Sahai commitments of h^{m_i} and u^{m_i} as

$$\begin{aligned}(C_{i,1}, C_{i,2}, C_{i,3}) &= \mathsf{Com}\big(m_i, (open_{m_i,1}, open_{m_i,2}, open_{m_i,3})\big) \\ &= \big(\mathsf{GSCom}(h^{m_i}, open_{m_i,1}), \mathsf{GSCom}(u^{m_i}, open_{m_i,2}), \mathsf{GSCom}(\tilde{V}_i^{m_i}, open_{m_i,3})\big).\end{aligned}$$

Generate an auxiliary variable $\theta = 1 \in \mathbb{Z}_p$ with its own commitment $C_\theta = \mathsf{GSCom}(\theta, open_\theta)$. Then, generate commitments $\{C_{\sigma_\tau}\}_{\tau=1}^{3}$ to $\{\sigma_\tau\}_{\tau=1}^{3}$ and give a NIZK proof that:

$$e(g^{\theta}, h) = e\big(\sigma_1, \tilde{U} \cdot \sigma_2 \cdot \prod_{i=1}^{n} \tilde{V}_i^{m_i}\big), \qquad e(u, \sigma_2) = e(\sigma_3, h), \qquad \theta = 1$$

$$e(g, \tilde{V}_i^{m_i}) = e(V_i, h^{m_i}), \qquad e(u, h^{m_i}) = e(u^{m_i}, h) \qquad \text{for } i \in \{1, \ldots, n\}$$

We denote the complete proof by $\pi^{sig} = \big(\{C_{\sigma_\tau}\}_{\tau=1}^{3}, \pi_1^{sig}, \pi_2^{sig}, \pi_\theta^{sig}, \{\pi_{m_i,1}^{sig}, \pi_{m_i,2}^{sig}\}_{i=1}^{n}\big)$. Note that π_1^{sig} is a proof for a quadratic equation and requires 4 elements of $\mathbb{G}_1$ and 4 elements of $\mathbb{G}_2$. Other equations are linear: each of π_2^{sig} and $\{\pi_{m_i,2}^{sig}\}_{i=1}^{n}$ demands 2 elements of $\mathbb{G}_1$ and 2 elements of $\mathbb{G}_2$ whereas proofs $\{\pi_{m_i,1}^{sig}\}_{i=1}^{n}$ only takes two elements of $\mathbb{G}_1$ each since all variables are in $\mathbb{G}_2$. The NIZK property stems from the fact that, on a simulated reference string, a commitment to 0 can be trapdoor opened to 1. For this reason, except for the equation $\theta = 1$ (for which one can simply equivocate the commitment), all other proofs can be simulated using the witnesses $1_{\mathbb{G}_1}$, $1_{\mathbb{G}_2}$ and $\theta = 0$ (see [31] for details).

VerifyProof$(\mathsf{params}, \mathsf{pk}, \pi^{sig}, (C_1, \ldots, C_n))$ Works in the obvious way and returns 1 if and only if the proof π^{sig} generated by SigProve is convincing.

EqComProve$(\mathsf{params}, \mathsf{pk}, x, y)$ The protocol for proving that two commitments open to the same value employ the usual technique already used in [3, 5] and is reviewed section 2.4.

SigIssue$(\mathsf{sk}, (C_1, \ldots, C_n)) \leftrightarrows$ **SigObtain**$(\mathsf{params}, \mathsf{pk}, \vec{m}, (C_1, open_1), \ldots, (C_n, open_n))$ is a secure two-party protocol between the issuer and the receiver where the latter obtains a signature on a committed vector of messages. As suggested in [5], this can be done using the 2-party protocol of Jarecki and Shmatikov [32] for computing a circuit on committed inputs. Another option would be to use the two-party computation protocol from [4] that relies on homomorphic encryption.

Theorem 1 ([5]). *If the HSDH and the TDH assumptions hold in $(\mathbb{G}_1, \mathbb{G}_2)$, the scheme is F-unforgeable w.r.t. the injective function $F(m) = (h^m, u^m)$.*

The proof of the above theorem can be found in [5].

3 Construction of Divisible E-cash

Known approaches for divisible e-cash systems [37, 35, 15, 18] make use of a binary tree with $L+1$ levels (for a monetary value of 2^L) where each node corresponds to an amount of money which is exactly one of half the amount of its father. Double-spenders are detected by making sure that each user cannot spend a coin corresponding to a node and one of its descendants or one of its ancestors.

In these tree-based constructions, one difficulty is for the user to efficiently prove that the path connecting the spent node to the root is well-formed. In [14, 18], this problem is solved using groups of distinct order: [14] uses a sequence of $L+1$ groups $\mathbb{G}_1, \ldots, \mathbb{G}_{L+1}$ of prime order p_ν where $\mathbb{G}_\nu$ is a subgroup of $\mathbb{Z}^\star_{p_{\nu-1}}$ for $\nu = 1, \ldots, L+1$. The solution of [18] uses $L+2$ bounded accumulators (one for each level of the tree and one for the whole tree) so as to only use two distinct group orders.

The use of groups of distinct order (and double discrete logarithms in [14]) is hardly compatible with Groth-Sahai proofs and, in our system we need to find a different technique to prevent users from spending coins associated with a node and one of its ancestors in the same tree.

3.1 General Description of the Scheme

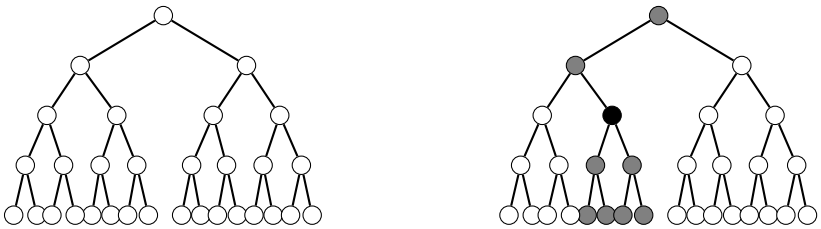

Fig. 1. Binary tree for spending one coin in a wallet of 2^4 coins

Our construction uses the tree-based approach. Each wallet $\mathcal{W}$ consists of a divisible coin of value 2^L, for some $L \in \mathbb{N}$, and the complexity of the spending phase depends on the depth of the node in the tree $\mathcal{W}$: the deeper the node is, the more expensive the spending phase will be. When an honest user $\mathcal{U}$ with key pairs $(pk_\mathcal{U}, sk_\mathcal{U})$ interacts with the bank $\mathcal{B}$, he obtains a wallet $\mathcal{W} = (s, t, sk_\mathcal{U}, \sigma, \mathsf{state})$ consisting of the bank's signature σ on the vector $(s, t, sk_\mathcal{U})$ where s, t are seeds for the Dodis-Yampolskiy PRF [28]. In our notation, state is a variable indicating the availability of coins.

To spend a coin of value $v = 2^\ell$ (with $\ell \leq L$) in the tree, the user $\mathcal{U}$ determines the next node corresponding to an unspent coin at height ℓ: the root of the tree is used if the user wants to spend his entire wallet at once whereas the leaves correspond to the smallest expandable amounts. Each node will be assigned a unique label consisting of an integer in the interval $[1, 2^{L+1} - 1]$. A simple assignment is obtained by labeling the root as $x_0 = 1$ and the rightmost leaf as $2^{L+1} - 1$, all other nodes being considered in order of appearance, from the left to the right and starting from the root.

In order to construct a valid coin, the user has to choose a previously unspent node of label x_{coin} at the appropriate level and do the following: (1) Prove his knowledge of a valid signature on committed messages $(s, t, sk_\mathcal{U})$ and his knowledge of $sk_\mathcal{U}$. (2) Commit to the PRF seeds via

commitments to the group elements $(S, T) = (h^s, h^t)$. (3) Commit to the path that connects x_{coin} to the root and prove that commitments pertain to a valid path. (4) Evaluate a coin serial number $Y_{L-\ell} = g^{1/(s+x_{coin})}$ where the input is the label of the node to be spent. (5) Generate NIZK proofs that everything was done consistently. (6) Add some material that makes it impossible to subsequently spend an ancestor or a descendant of x_{coin} without being detected.

At step (1), we use the multi-block P-signature scheme to sign the block $(s, t, sk_{\mathcal{U}})$. Using the proof produced by $\mathsf{SigProve}$ in the P-signature, we can efficiently prove knowledge of a signature on committed inputs in NIZK.

The trickiest problem to solve is actually (6). If $\{x_0, \ldots, x_{L-\ell}\}$ denotes the path from the root x_0 to $x_{L-\ell} = x_{coin}$, for each $j \in \{0, \ldots, L-\ell\}$, we include in the coin a pair $(T_{j,1}, T_{j,2})$ where $T_{j,1} = h^{\delta_{j,1}}$, for some random $\delta_{j,1} \xleftarrow{R} \mathbb{Z}_p$, and $T_{j,2} = e(Y_j, T_{j,1})$, where $Y_j = g^{1/(s+x_j)}$ is the value of the PRF for the label x_j. In addition, $\mathcal{U}$ must add a NIZK proof that the pair $(T_{j,1}, T_{j,2})$ was correctly calculated. By doing so, at the expense of n_s pairing evaluations at each deposit (where n_s denotes the number of previously spent coins), the bank will be able to detect whether a spent node is in the path connecting a previously spent node to the root. At the same time, if $\mathcal{U}$ does not overspend at any time, the coins he spends remain computationally unlinkable.

By itself, the pair $(T_{j,1}, T_{j,2})$ only renders cheating attempts evident. In order to expose the public key $pk_{\mathcal{U}}$ of double spenders, $\mathcal{U}$ is required to add a pair $(T_{j,3}, T_{j,4}) = (h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(Y_j, T_{j,3}))$ at each node of the path: by doing so, $pk_{\mathcal{U}}$ is exposed if $\mathcal{U}$ subsequently spends a node above x_{coin} in the path. However, we have to consider a second kind of double-spending, where the two coins involve the same tree node $x_{coin} = x_{L-\ell}$. To deal with this case, we require $\mathcal{U}$ to additionally use the seed t of his wallet and the merchant's data R and compute another security tag $Z_{L-\ell} = g^{sk_{\mathcal{U}}} g^{R/(t+x_{L-\ell})}$. The latter will be used to identify cheating users in the same way as in [10].

Finally, in order to obtain the exculpability property (and prevent a dishonest bank from wrongly accusing the user of double-spending coins), we need to add yet another pair of the form $(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(Y_j, h^{\delta_3}))$, where $g_0 \in \mathbb{G}_1$ is part of the CRS, in such a way that framing the user requires to compute $e(g_0, h^{sk_{\mathcal{U}}})$ and solve a (computational) Bilinear Diffie-Hellman instance.

In order to solve problem (5), we need to generate non-interactive proofs for a number of pairing-product equations. Since the notion of anonymity requires to build a simulator that emulates the prover without knowing any witness, it means that we need NIZK proofs for pairing product equations on multiple occasions. Fortunately, the specific equations fall into the category of equations for which NIZK proofs are possible at the cost of introducing extra variables. For this reason, we will have to introduce auxiliary variables for each pairing product equations.

Example. Suppose that, in his wallet $L = 4$, $\mathcal{U}$ uses the seed s to spend the amount of $v = 2^2$. The left part of Figure 1 represents the state of the wallet when no coin has been spent. In the rightmost tree, the black node indicates the target node x_{coin} of value v and greyed nodes are those that cannot be spent any longer once the black node was spent.

3.2 Construction

We now describe our divisible e-cash system where the withdrawal protocol allows users to obtain a wallet of a divisible coin of value 2^L.

$\mathsf{CashSetup}(\lambda)$: chooses bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order $p > 2^\lambda$ and generators $g, g_0 \xleftarrow{R} \mathbb{G}_1$, $h \xleftarrow{R} \mathbb{G}_2$. It also generates a Groth-Sahai common reference string $\mathsf{params}_{GS} = \{g, h, \vec{u}_1, \vec{u}_2, \vec{v}_1, \vec{v}_2\}$

for the perfectly soundness setting. The algorithm also selects a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. The output is $\mathsf{params} := \{(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T),\ g_0,\ \mathsf{params}_{GS},\ H\}$.

BankKG(params, L)**:** runs $\mathsf{SigSetup}(\lambda, n)$ with $n = 3$ to obtain a key pair $(\mathsf{sk}, \mathsf{pk})$ for the P-signature of section 2.4. The bank's key pair is defined to be $(sk_{\mathcal{B}}, pk_{\mathcal{B}}) = (\mathsf{sk}, \mathsf{pk})$ and $pk_{\mathcal{B}}$ consists of

$$pk_{\mathcal{B}} = \big(u, U = g^{\beta}, \tilde{U} = h^{\beta}, \{V_i = g^{a_i}, \tilde{V}_i = h^{a_i}\}_{i=1}^{3},\ L\big).$$

UserKG(params)**:** the user $\mathcal{U}$ defines his key pair as $(sk_{\mathcal{U}}, pk_{\mathcal{U}} = e(g,h)^{sk_{\mathcal{U}}})$ for a random $sk_{\mathcal{U}} \xleftarrow{R} \mathbb{Z}_p$.

Withdraw$\big(\mathcal{U}(\mathsf{params}, pk_{\mathcal{B}}, sk_{\mathcal{U}}), \mathcal{B}(\mathsf{params}, pk_{\mathcal{U}}, sk_{\mathcal{B}})\big)$**:** $\mathcal{U}$ interacts with $\mathcal{B}$ as follows:

1. $\mathcal{U}$ first picks $s', t' \xleftarrow{R} \mathbb{Z}_p$ and computes perfectly hiding commitments $C_{s'} = \mathsf{Com}(s', open_{s'})$, $C_{t'} = \mathsf{Com}(t', open_{t'})$ and $C_{sk_{\mathcal{U}}} = \mathsf{Com}(sk_{\mathcal{U}}; open_{sk_{\mathcal{U}}})$. The user sends $(C_{s'}, C_{t'}, C_{sk_{\mathcal{U}}})$ to $\mathcal{B}$ and provides interactive witness indistinguishable proofs that he knows how to open $(C_{s'}, C_{t'})$. In addition, he provides an interactive zero-knowledge[4] proof that $C_{sk_{\mathcal{U}}}$ is a commitment to the private key $sk_{\mathcal{U}}$ that was used to generate $pk_{\mathcal{U}}$.
2. If the proofs verifies, $\mathcal{B}$ picks $(s'', t'') \leftarrow \mathbb{Z}_p^2$ which are sent to $\mathcal{U}$.
3. The user $\mathcal{U}$ sets $s = s' + s''$ and $t = t' + t''$, updates commitments $C_{s'}$ and $C_{t'}$ into commitments $C_s = \mathsf{Com}(s, open_s)$ and $C_t = \mathsf{Com}(t, open_t)$. The user sends (C_s, C_t) to the bank with a proof that these commitments were properly calculated.
4. $\mathcal{U}$ and $\mathcal{B}$ jointly run the protocol

   $$\mathsf{SigIssue}(\mathsf{params}, \mathsf{sk}, (C_s, C_t, C_{sk_{\mathcal{U}}})) \leftrightarrows \mathsf{SigObtain}(\mathsf{params}, \mathsf{pk}, (s, t, sk_{\mathcal{U}}), (open_s, open_t, open_{sk_{\mathcal{U}}}))$$

   in such a way that $\mathcal{U}$ eventually obtains $\mathcal{B}$'s signature σ on $(s, t, sk_{\mathcal{U}})$. The user $\mathcal{U}$ stores the wallet $\mathcal{W} = (s, t, sk_{\mathcal{U}}, \sigma, \mathsf{state})$, where $\mathsf{state} = \emptyset$.
5. $\mathcal{B}$ records a debit of value $v = 2^L$ on $\mathcal{U}$'s account. Then, $\mathcal{B}$ stores the transcript of the protocol and the tracing information $pk_{\mathcal{U}}$ in its database T.

Spend$\big(\mathsf{params}, pk_{\mathcal{B}}, \mathcal{W} = (s, t, sk_{\mathcal{U}}, \sigma, \mathsf{state}), 2^{\ell}, pk_{\mathcal{M}}, \mathtt{info}\big)$**:** suppose that the user $\mathcal{U}$ wants to spend a coin of value 2^{ℓ} for the wallet $\mathcal{W}$ of initial value 2^L. Using state, $\mathcal{U}$ determines the label $x_{coin} \in [1, 2^{L+1} - 1]$ of the first node corresponding to an unspent coin at height ℓ in the tree associated with the wallet. Let $\{x_0, x_1, \ldots, x_{L-\ell}\}$ denote the path connecting node $x_{coin} = x_{L-\ell}$ to the root $x_0 = 1$ of the tree. The user $\mathcal{U}$ computes $S = h^s$ and $T = h^t$ and conducts the following steps.

1. $\mathcal{U}$ has to prove that he knows a signature σ on the committed vector $(s, t, sk_{\mathcal{U}})$. To this end, he generates $\big(\{C_{S,i}\}_{i=1}^{3}, \{C_{T,i}\}_{i=1}^{3}, \{C_{\mathcal{U},i}\}_{i=1}^{3}, \pi^{sig}\big) \leftarrow \mathsf{SigProve}(\mathsf{params}, \mathsf{pk}, \sigma, (s, t, sk_{\mathcal{U}}))$. Note that the commitments produced by $\mathsf{SigProve}$ include $\{C_{\mathcal{U},i}\}_{i=1}^{3}$, which are commitments to $(L_{\mathcal{U},1}, L_{\mathcal{U},2}, L_{\mathcal{U},3}) = (h^{sk_{\mathcal{U}}}, u^{sk_{\mathcal{U}}}, \tilde{V}_3^{sk_{\mathcal{U}}})$. Other commitments $\{C_{S,i}, C_{T,i}\}_{i=1}^{3}$ are commitments to $(L_{S,1}, L_{S,2}, L_{S,3}) = (h^s, u^s, \tilde{V}_1^s)$ and $(L_{T,1}, L_{T,2}, L_{T,3}) = (h^t, u^t, \tilde{V}_2^t)$. In addition, $\mathcal{U}$ generates an additional commitment $C_{K_{\mathcal{U}}} = \mathsf{GSCom}(h^{sk_{\mathcal{U}}}, open'_{\mathcal{U}})$ to $K_{\mathcal{U}} = h^{sk_{\mathcal{U}}}$ and a NIZK proof $\pi_{K_{\mathcal{U}}} \leftarrow \mathsf{EqComProve}(L_{\mathcal{U},1}, K_{\mathcal{U}})$ that $C_{K_{\mathcal{U}}}$ and $C_{\mathcal{U},1}$ are commitment to the same value. This amounts to prove that

   $$e(L_{\mathcal{U},1}/K_{\mathcal{U}}, h^{\theta}) = 1_{\mathbb{G}_T} \quad \text{and} \quad \theta = 1, \tag{1}$$

   for some variable $\theta \in \mathbb{Z}_p$ contained in $C_{\theta} = \mathsf{GSCom}(\theta, open_{\theta})$ and that will be re-used in subsequent steps of the spending protocol.

[4] The zero-knowledge property will be needed in the proof of weak exculpability.

2. For $j = 0$ to $L - \ell$ do the following.

   a. If $j > 0$, generate a commitment $C_{X_j} = \mathsf{GSCom}(h^{x_j}, open_{x_j})$ to $X_j = h^{x_j}$ and a proof that $x_j = 2x_{j-1} + b_j$, for some bit $b_j \in \{0,1\}$. To this end, generate the commitments $C_{b_j} = \mathsf{GSCom}(g^{b_j}, open_{b_j})$ and $C'_{b_j} = \mathsf{GSCom}(h^{b_j}, open'_{b_j})$ as well as a NIZK proof $\pi_{x_j} \leftarrow \mathsf{EqComProve}(C_{X_j}, C'_{X_j})$ that C_{X_j} and $C'_{X_j} = C^2_{X_{j-1}} \cdot C'_{b_j}$ open to the same value. To prove that $b_j \in \{0,1\}$, the user generates a NIZK proof $(\pi_{b_j,1}, \pi_{b_j,2})$ for the pairing-product equations $e(g^{b_j}, h) = e(g, h^{b_j})$ and $e(g^{b_j}, h^{b_j}) = e(g^{b_j}, h)$, which guarantee that $b_j^2 = b_j$, so that $b_j \in \{0,1\}$.

   b. If $j < L - \ell$, generate a commitment $C_{Y_j} = \mathsf{GSCom}(Y_j, open_{Y_j})$ to $Y_j = g^{1/(s+x_j)}$ and a NIZK proof π_{Y_j} that $e(Y_j, L_{S,1} \cdot X_j) = e(g,h)$, where $X_j = h^{x_j}$. This consists of a commitment $C_{\Phi_{Y_j}}$ to a variable $\Phi_{Y_j} \in \mathbb{G}_1$ and a proof that $e(\Phi_{Y_j}, L_{S,1} \cdot X_j) = e(g,h)$ and $e(Y_j/\Phi_{Y_j}, h^\theta) = 1_{\mathbb{G}_T}$. Then, pick $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\begin{array}{lll} T_{j,1} = h^{\delta_{j,1}}, & T_{j,3} = h^{\delta_{j,2}}, & T_{j,5} = h^{\delta_{j,3}}, \\ T_{j,2} = e(Y_j, h)^{\delta_{j,1}}, & T_{j,4} = pk_{\mathcal{U}} \cdot e(Y_j, h)^{\delta_{j,2}}, & T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(Y_j, h)^{\delta_{j,3}}. \end{array}$$

   Then, generate NIZK proofs $(\pi_{j,T_1}, \pi_{j,T_3}, \pi_{j,T_5})$ that variables $(Y_j, K_{\mathcal{U}})$ satisfy

$$T_{j,2} = e(Y_j, T_{j,1}) \qquad T_{j,4} = e(g, K_{\mathcal{U}}) \cdot e(Y_j, T_{j,3}) \qquad T_{j,6} = e(g_0, K_{\mathcal{U}}) \cdot e(Y_j, T_{j,5}). \quad (2)$$

   These proofs require new commitments $\{C_{\Phi_{j,k}}\}_{k=1,3,5}$, $C_{\Phi'_{Y_j}}$ and $C_{\Phi''_{Y_j}}$ to auxiliary variables $\{\Phi_{j,k}\}_{k=1,3,5}$, $\Phi'_{Y_j}, \Phi''_{Y_j} \in \mathbb{G}_1$ respectively and proofs for relations

$$\begin{array}{lll} T_{j,2} = e(Y_j, \Phi_{j,1}), & T_{j,4} = e(g, K_{\mathcal{U}}) \cdot e(\Phi_{Y'_j}, \Phi_{j,3}), & T_{j,6} = e(g_0, K_{\mathcal{U}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5}), \\ e(Y_j/\Phi_{Y'_j}, h^\theta) = 1_{\mathbb{G}_T}, & e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}, & \{e(g^\theta, T_{j,k}/\Phi_{j,k}) = 1_{\mathbb{G}_T}\}_{k \in \{1,3,5\}}. \end{array}$$

   c. If $j = L - \ell$, compute the serial number $Y_{L-\ell} = g^{1/(s+x_{L-\ell})}$ and generate a NIZK proof $\pi_{Y_{L-\ell}}$ that $e(Y_{L-\ell}, L_{S,1} \cdot X_{L-\ell}) = e(g,h)$. This proof consists of a commitment $C_{\Phi_{Y_{L-\ell}}}$ to $\Phi_{Y_{L-\ell}} \in \mathbb{G}_1$ and proofs for equations

$$e(\Phi_{Y_{L-\ell}}, L_{S,1} \cdot X_{L-\ell}) = e(g,h), \qquad e(Y_{L-\ell}/\Phi_{Y_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}.$$

   Then, compute $Z_{L-\ell} = g^{sk_{\mathcal{U}}} \cdot g^{R/(t+x_{L-\ell})}$, where $R = H(\mathtt{info}, pk_{\mathcal{M}}) \in \mathbb{Z}_p$, and a NIZK proof $\pi_{Z_{L-\ell}}$ that $Z_{L-\ell}$ is well-formed. This requires new commitments $C_{W_{L-\ell}}, C_{\Phi_{W_{L-\ell}}}$ to auxiliary variables $W_{L-\ell} = g^{1/(t+x_{L-\ell})}$, $\Phi_{W_{L-\ell}} = g^{1/(t+x_{L-\ell})}$ and a proof that:

$$\begin{array}{ll} e(Z_{L-\ell}, h) = e(g, K_{\mathcal{U}}) \cdot e(W_{L-\ell}, h^R), & e(W_{L-\ell}, L_{T,1} \cdot X_{L-\ell}) = e(g,h) \\ e(W_{L-\ell}/\Phi_{W_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}. & \end{array}$$

Finally, update state into $\mathsf{state}' = \mathsf{state} \cup \{(x_{coin})\}$ and output the coin

$$\begin{aligned} coin = \Big(& \{C_{S,i}\}_{i=1}^3, \{C_{T,i}\}_{i=1}^3, \{C_{\mathcal{U},i}\}_{i=1}^3,\ C_{K_{\mathcal{U}}},\ \pi_{K_{\mathcal{U}}},\ \pi^{sig},\ \{C_{X_j},\ C_{b_j},\ C'_{b_j},\ \pi_{x_j},\ \pi_{b_j,1}\ \pi_{b_j,2}\}_{j=1}^{L-\ell}, \\ & \{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}),\ C_{Y_j},\ C_{\Phi_{Y_j}},\ C'_{\Phi_{Y_j}},\ C''_{\Phi_{Y_j}},\ \{C_{\Phi_{j,k}}\}_{k \in \{1,3,5\}},\ \pi_{Y_j},\ \pi_{j,T_1}, \\ & \pi_{j,T_3},\ \pi_{j,T_5}\}_{j=0}^{L-\ell-1},\ Y_{L-\ell},\ Z_{L-\ell},\ C_{\Phi_{Y_{L-\ell}}}, C_{W_{L-\ell}},\ C_{\Phi_{W_{L-\ell}}},\ \pi_{Y_{L-\ell}},\ \pi_{Z_{L-\ell}},\ \mathtt{info} \Big) \end{aligned}$$

VerifyCoin$(\mathsf{params}, pk_{\mathcal{M}}, pk_{\mathcal{B}}, v = 2^{\ell}, coin)$**:** parse *coin* as above. Return 1 iff all proofs verify.

Deposit$(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, 2^{\ell}, \mathrm{DB}_{\mathcal{B}})$**:** parse *coin* as above and perform the same checks as VerifyCoin. Then, define $\mathrm{DB}'_{\mathcal{B}} = \mathrm{DB}_{\mathcal{B}} \cup \{(coin, \mathsf{flag}, 2^{\ell}, pk_{\mathcal{M}})\}$ where the value of flag depends on whether *coin* is a valid coin of value 2^{ℓ} and whether a cheating attempt is detected.

- If *coin* does not properly verify, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{M}$" to indicate a cheating merchant.
- If *coin* verifies, $\mathcal{B}$ runs the following test. For each entry $(coin_s, \mathsf{flag}_s, 2^{\ell_s}, pk_{\mathcal{M}_s}) \in \mathrm{DB}_{\mathcal{B}}$, where $s = 1$ to $|\mathrm{DB}_{\mathcal{B}}|$, $\mathcal{B}$ parses $coin_s$ as above. If $(\mathtt{info}_s, pk_{\mathcal{M}_s}) = (\mathtt{info}, pk_{\mathcal{M}})$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{M}$". Otherwise, from $coin_s$, it extracts the path $\{(T_{s,j,1}, T_{s,j,2}, T_{s,j,3}, T_{s,j,4})\}_{j=0}^{L-\ell_s-1}$, the serial number $Y_{L-\ell_s} \in \mathbb{G}_1$ and the tag $Z_{L-\ell_s} \in \mathbb{G}_1$. It also parses *coin* to extract the path $\{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4})\}_{j=0}^{L-\ell-1}$, the serial number $Y_{L-\ell} \in \mathbb{G}_1$ and the tag $Z_{L-\ell}$. If $\ell < \ell_s$ and $T_{L-\ell_s,2} = e(Y_{L-\ell_s}, T_{L-\ell_s,1})$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{U}$", outputs $coin_s$ and *coin* and reports a double-spending. Likewise, if $\ell > \ell_s$ and $T_{s,L-\ell,2} = e(Y_{L-\ell}, T_{s,L-\ell,1})$, $\mathcal{B}$ also sets $\mathsf{flag} =$ "$\mathcal{U}$" and outputs $coin_s$ and *coin*. Finally, if $\ell = \ell_s$, $\mathcal{B}$ sets $\mathsf{flag} =$ "$\mathcal{U}$" if and only if $Y_{L-\ell} = Y_{L-\ell_s}$.
- If *coin* verifies and no double-spending is detected, $\mathcal{B}$ sets $\mathsf{flag} =$ "accept" and credits the account of $pk_{\mathcal{M}}$ by the amount of 2^{ℓ}.

After the above tests, the updated database $\mathrm{DB}'_{\mathcal{B}}$ supersedes $\mathrm{DB}_{\mathcal{B}}$.

Identify$(\mathsf{params}, pk_{\mathcal{B}}, coin_a, coin_b)$**:** on input of fraudulent coins $coin_a$ and $coin_b$, the bank $\mathcal{B}$ can identify the double-spender as follows.

1. Parse $coin_a$ to extract $\mathtt{info}_a$, $\{(T_{j,1}^{(a)}, T_{j,2}^{(a)}, T_{j,3}^{(a)}, T_{j,4}^{(a)})\}_{j=0}^{L-\ell_a-1}$, $(Y_{L-\ell_a}^{(a)}, Z_{L-\ell_a}^{(a)}) \in \mathbb{G}_1^2$ and $coin_b$ to obtain $\mathtt{info}_b$, $\{(T_{j,1}^{(b)}, T_{j,2}^{(b)}, T_{j,3}^{(b)}, T_{j,4}^{(b)})\}_{j=0}^{L-\ell_b-1}$ and $(Y_{L-\ell_b}^{(b)}, Z_{L-\ell_b}^{(b)}) \in \mathbb{G}_1^2$.
2. If $\ell_b > \ell_a$, recover $pk_{\mathcal{U}}$ as $pk_{\mathcal{U}} = T_{L-\ell_b,4}^{(a)} / e(Y_{L-\ell_b}^{(b)}, T_{L-\ell_b,3}^{(a)})$. If $\ell_b < \ell_a$, $pk_{\mathcal{U}}$ can be obtained as $pk_{\mathcal{U}} = T_{L-\ell_a,4}^{(b)} / e(Y_{L-\ell_a}^{(a)}, T_{L-\ell_a,3}^{(b)})$. In the case $\ell_a = \ell_b$, we must have $Y_{\ell_a} = Y_{\ell_b}$. Then, $\mathcal{B}$ computes $R_a = H(\mathtt{info}_a, pk_{\mathcal{M}_a})$, $R_b = H(\mathtt{info}_b, pk_{\mathcal{M}_b})$ and then $\kappa = (Z_{L-\ell_a}^{(a)} / Z_{L-\ell_b}^{(b)})^{1/(R_a - R_b)}$, which allows recovering $g^{sk_{\mathcal{U}}} = Z_{L-\ell_a}^{(a)} / \kappa^{R_a}$ and thus $pk_{\mathcal{U}} = e(g^{sk_{\mathcal{U}}}, h)$.

The security of the scheme relies on the collision-resistance of H and the intractability assumptions recalled in Section 2.3. More precisely, we state the following theorem for which a proof is given in Appendix A.

Theorem 2. *Assuming that H is a collision-resistant hash function and that the SXDH, D3DH, TDH, D3DH, q_w-HSDH and the 2^{L+2}-DDHI assumptions where q_w denotes the number of $\mathcal{Q}_{\mathsf{withdraw}}$ queries all hold in $(\mathbb{G}_1, \mathbb{G}_2)$, our e-cash scheme provides anonymity, balance, identification and weak-exculpability.*

The most difficult part of the security proof is the proof of anonymity. More precisely, when it comes to build a simulator, we need to simulate NIZK proofs for pairing product equations of the form (2), which is non-trivial. Indeed, as noted in [31], this is only known to be possible when the target element of the equation (which lives in $\mathbb{G}_T$) can be written as a pairing of known elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. The problem is that, in equations like (2), some pairing values have to be gradually replaced by uniformly random values of $\mathbb{G}_T$. To deal with this problem, we appeal to the D3DH assumption in a similar way to [33]. Namely, the D3DH input element Γ, which is either g^{abc} or a random element of $\mathbb{G}_1$, is available as a "pre-image" of the target pairing value and makes it possible to simulate proofs for pairing product equations at the expense of introducing auxiliary variables.

4 Conclusion

This paper presented the first construction of divisible e-cash system that does not require the random oracle model or interaction during the spending phase. Our scheme relies on the availability of a common reference string, which is inevitable as long as NIWI and NIZK proof systems are needed. In the future, it will be interesting to find better solutions in the standard model. Indeed, the computational complexity of the deposit protocol at the bank's end makes our scheme impractical. For this reason, it is only a feasibility result.

References

1. M. H. Au, W. Susilo, Y. Mu. Practical Anonymous Divisible E-Cash from Bounded Accumulators. In *Financial Cryptography 2008*, LNCS 5143, pp. 287–301 , 2008.
2. M. H. Au, Q. Wu, W. Susilo, Y. Mu. Compact E-Cash from Bounded Accumulator. In *CT-RSA'07*, LNCS 4377, pp. 178–195, 2007.
3. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC'08*, LNCS 4948, pages 356–374, 2008.
4. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, H. Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *Crypto'09*, LNCS 5677, pp. 108–125, 2009
5. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact E-Cash and Simulatable VRFs Revisited. In *Pairing'09*, LNCS 5671, pp. 114–131, 2009.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pp. 62–73, 1993.
7. O. Blazy, S. Canard, G. Fuchsbauer, A. Gouget, H. Sibert, J. Traoré. Achieving Optimal Anonymity in Transferable E-Cash with a Judge. In *Africacrypt 2011*, LNCS 6737, pp. 206–223, 2011.
8. D. Boneh, C. Gentry, B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Crypto'05*, *LNCS* 3621, pp. 258–275, 2005.
9. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC'07* , LNCS 4450, pp. 1–15, 2007.
10. J. Camenisch, S. Hohenberger, A. Lysyanskaya. Compact E-Cash. In *Eurocrypt'05*, LNCS 3494, pp. 302–321, 2005.
11. J. Camenisch, S. Hohenberger, A. Lysyanskaya. Balancing Accountability and Privacy Using E-Cash. In *SCN'06*, LNCS 4116, pp. 141–155, 2006.
12. J. Camenisch, M. Kohlweiss, C. Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC'09*, *LNCS* 5443, pp. 481–500, 2009.
13. J. Camenisch, A. Lysyanskaya, M. Meyerovich. Endorsed E-Cash In *IEEE Security & Privacy'07*, pp. 101–115, 2007.
14. S. Canard, A. Gouget. Divisible E-Cash Systems Can Be Truly Anonymous. In *Eurocrypt'07*, LNCS 4515, pp. 482–497, 2007.
15. S. Canard, A. Gouget, J. Traoré. Improvement of Efficiency in (Unconditional) Anonymous Transferable E-Cash. In *Financial Cryptography 2008*, LNCS 5143, pp 202–214, 2008.
16. S. Canard, A. Gouget. Anonymity in Transferable E-cash. In *ACNS'08*, LNCS 5037, pp. 207–223, 2008.
17. S. Canard, A. Gouget, E. Hufschmitt. A Handy Multi-coupon System. In *ACNS'06*, LNCS 3989, pp. 66–81, 2006.
18. S. Canard, A. Gouget. Multiple Denominations in E-cash with Compact Transaction Data. In *Financial Cryptography 2010*, LNCS 6052, pp. 82–97, 2010.
19. R. Canetti, O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited. In *STOC'98*, pp. 209–218, ACM Press, 1998.
20. A.-H. Chan, Y. Frankel, Y. Tsiounis. Easy Come - Easy Go Divisible Cash. In *Eurocrypt'98*, *LNCS* 1403, pp. 561–575, 1998.
21. M. Chase, A. Lysyanskaya. Simulatable VRFs with Applications to Multi-theorem NIZK. In *Crypto'07*, *LNCS* 4622, pp. 303–322, 2007.
22. D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto'82*, pp. 199–203, 1982.
23. D. Chaum. Blind Signature Systems. In *Crypto'83*, p. 153, 1983.

24. D. Chaum, A. Fiat, M. Naor. Untraceable Electronic Cash. In *Crypto'88*, *LNCS* 403, pp. 319–327, 1988.
25. D. Chaum, T. Pedersen. Transferred Cash Grows in Size. In *Eurocrypt'92*, *LNCS* 658, pp. 390–407, 1992.
26. S. D'Amiano, G. Di Crescenzo. Methodology for Digital Money based on General Cryptographic Tools. In *Eurocrypt'94*, *LNCS* 950, pp. 156–170, 1994.
27. Y. Dodis. Efficient Construction of (Distributed) Verifiable Random Functions. In *PKC'03*, *LNCS* 2567, pp. 1–17, 2003.
28. Y. Dodis, A. Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In *PKC'05*, *LNCS* 3386, pp. 416–431, 2005.
29. M. K. Franklin, M. Yung. Secure and Efficient Off-Line Digital Money. In *ICALP'93*, *LNCS* 700, pp. 265–276, 1993.
30. G. Fuchsbauer, D. Pointcheval, D. Vergnaud. Transferable Constant-Size Fair E-Cash. In *CANS'09*, *LNCS* 5888, pp. 226–247, 2009.
31. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt'08*, LNCS 4965, pp. 415–432, 2008.
32. S. Jarecki, V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. In *Eurocrypt'07*, LNCS 4515, pp. 97–114, 2007.
33. B. Libert, D. Vergnaud. Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model. In *CANS'09*, *LNCS* 5888, pp. 498-517, 2009.
34. S. Micali, M.-O. Rabin, S. Vadhan. Verifiable Random Functions. In *FOCS'99*, pp. 120–130, 1999.
35. T. Nakanishi, Y. Sugiyama. Unlinkable Divisible Electronic Cash. In *ISW'00*, LNCS 1975, pp. 121–134, 2000.
36. T. Okamoto. K. Ohta. Universal Electronic Cash. In *Crypto'91*, LNCS 576, pp. 324–337, 1991.
37. T. Okamoto. An Efficient Divisible Electronic Cash Scheme. In *Crypto'95*, *LNCS* 963, pp. 438–451, 1991.
38. J.-C. Pailles. New Protocols for Electronic Money. In *Auscrypt'92*, *LNCS* 718, pp. 263–274, 1992.
39. T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Crypto'91*, *LNCS* 576, pp. 129–140, 1991.
40. Y. Tsiounis. Efficient Electronic Cash: New Notions and Techniques. PhD Thesis, Northeaster University, Boston, 1997.

A Proof of Theorem 2

To prove the above theorem, we separately consider the various security notions.

A.1 Anonymity

Theorem 3 (Anonymity). *The scheme provides anonymity under the SXDH assumption, the 2^{L+2}-DDHI assumption and the D3DH assumption.*

Proof. We construct the following simulator $\mathcal{S} = (\mathsf{SimCashSetup}, \mathsf{SimSpend})$ which executes the spend protocol without using any users' wallet, private key or state information.

$\mathsf{SimCashSetup}(\lambda)$: produces a simulated Groth-Sahai reference string $\mathsf{params}_{GS} = \{g, h, \vec{u}_1, \vec{u}_2, \vec{v}_1, \vec{v}_2\}$, where $\vec{u}_2 = \vec{u}_2^{\xi_1} \cdot (1, g)^{-1}$ and $\vec{v}_2 = \vec{v}_2^{\xi_2} \cdot (1, h)^{-1}$, for some random $\tau = (\xi_1, \xi_2) \xleftarrow{R} \mathbb{Z}_p^2$ and retains the simulation trapdoor $\tau = (\xi_1, \xi_2)$.

$\mathsf{SimSpend}(\mathsf{params}, pk_{\mathcal{B}}, f, v, pk_{\mathcal{M}}, \mathtt{info})$: conducts the following steps.

1. Generate $\{C_{S,i}\}_{i=1}^3, \{C_{T,i}\}_{i=1}^3, \{C_{\mathcal{U},i}\}_{i=1}^3$ as commitments to $\{1_{\mathbb{G}_2}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2}\}$ and $C_{K_{\mathcal{U}}}$ as a commitment to $1_{\mathbb{G}_2}$. The proof π^{sig} is computed using the NIZK simulator for $\mathsf{SigProve}$. The NIZK proof $\pi_{K_{\mathcal{U}}}$ that $L_{\mathcal{U}} = K_{\mathcal{U}}$ is a real proof and a commitment C_θ to $\theta = 1$ is also generated.
2. Choose a random node x_{coin} at level $\ell = \log v$ and let $\{x_0, \ldots, x_{L-\ell}\}$ denote the path connecting x_{coin} to the root. For $j = 0$ to $L - \ell$ do the following.

a. If $j > 0$, generate a commitment $C_{X_j} = \mathsf{GSCom}(h^{x_j}, open_{x_j})$ to $X_j = h^{x_j}$ and an honestly generated proof that $x_j = 2x_{j-1} + b_j$, for some bit $b_j \in \{0,1\}$. To this end, compute $C_{b_j} = \mathsf{GSCom}(g^{b_j}, open_{b_j})$, $C'_{b_j} = \mathsf{GSCom}(h^{b_j}, open'_{b_j})$ and honestly generate the proof π_{x_j} that C_{X_j} and $C^2_{X_{j-1}} \cdot C'_{b_j}$ open to the same value and the proof that $b_j \in \{0,1\}$. The proof $(\pi_{b_j,1}, \pi_{b_j,2})$ is also generated as a real proof.

b. If $j < L - \ell$, compute C_{Y_j}, $C_{\Phi_{Y_j}}$ and $C_{\Phi_{Y'_j}}$ as commitments to $Y_j = \eta$, $\Phi_{Y'_j} = \eta'$ and $\Phi_{Y''_j} = \eta''$ with $\eta, \eta', \eta'' \xleftarrow{R} \mathbb{G}_1$. Generate the following elements and proofs.

   1. Define $C_{\Phi_{Y_j}}$ as a commitment to $\Phi_{Y_j} = g^{1/x_j}$ and honesty generate a proof that $e(\Phi_{Y_j}, L_{S,1} \cdot X_j) = e(g,h)$.
   2. Pick $\delta_1, \delta_2, \delta_3 \xleftarrow{R} \mathbb{Z}_p$, compute $T_{j,1} = h^{\delta_1}$, $T_{j,2} = e(\eta, h^{\delta_1})$, $T_{j,3} = h^{\delta_2}$, $T_{j,4} = e(\eta', h^{\delta_2})$, $T_{j,5} = h^{\delta_3}$, $T_{j,6} = e(\eta'', h^{\delta_3})$. Compute $C_{\Phi_{j,1}}, C_{\Phi_{j,3}}, C_{\Phi_{j,5}}$ as Groth-Sahai commitments to $\Phi_{j,1} = h^{\delta_1}$, $\Phi_{j,3} = h^{\delta_2}$, $\Phi_{j,5} = h^{\delta_3}$. Honestly generate proofs that $T_{j,2} = e(Y_j, \Phi_{j,1})$, $T_{j,2} = e(g, K_{\mathcal{U}}) \cdot e(\Phi_{Y'_j}, \Phi_{j,3})$, $T_{j,3} = e(g_0, K_{\mathcal{U}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5})$.
   3. Using τ, generate simulated proofs that $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,5}, \Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$. Likewise, generate fake proofs that

$$e(Y_j/\Phi_{Y_j}, h^\theta) = e(Y_j/\Phi_{Y'_j}, h^\theta) = e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}.$$

   The above steps give simulated proofs π_{Y_j}, $\pi_{j,T_1}, \pi_{j,T_3}, \pi_{j,T_5}$.

c. Set $Y_{L-\ell} = \eta$, $Z_{L-\ell} = \eta'^R$ (which will appear in the coin) and $W_{L-\ell} = \eta'$ using randomly chosen $\eta, \eta' \xleftarrow{R} \mathbb{G}_1$. To simulate the non-interactive proofs $\pi_{Y_{L-\ell}}$ and $\pi_{Z_{L-\ell}}$, define $\Phi_{Y_{L-\ell}} = \Phi_{W_{L-\ell}} = g^{1/x_{L-\ell}}$ and generate proofs that $e(\Phi_{Y_{L-\ell}}, L_{S,1} \cdot X_{L-\ell}) = e(g,h)$ and $e(\Phi_{W_{L-\ell}}, L_{T,1} \cdot X_{L-\ell}) = e(g,h)$. Using the trapdoor τ of the fake CRS, simulate fake proofs that $e(Y_{L-\ell}/\Phi_{Y_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(W_{L-\ell}/\Phi_{W_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}$ by trapdoor opening to 0 the commitment to $\theta = 1$. Finally, generate the proof that $e(Z_{L-\ell}, h) = e(W_{L-\ell}, h^R)$ using the real witness $W_{L-\ell}$.

To prove that the outputs of $\mathcal{S}$ are indistinguishable from outputs of the Spend algorithm whenever the latter is honestly executed, we proceed with a sequence of games where the first game is the real attack game where the real oracle $\mathcal{Q}_{\mathsf{Spend}}$ is executed at each spending query. In the last game, the latter is replaced by the simulator $\mathcal{S}$ that executes the spending algorithm without knowing on behalf of which user it is spending the coin.

The sequence of games is described as follows. In Game i, we call E_i the probability that the adversary $\mathcal{A}$, acting as a distinguisher, outputs 1.

Game 1: is the real game where the adversary $\mathcal{A}$ generates the bank's public key $pk_{\mathcal{B}}$. Upon $\mathcal{A}$'s request, $\mathcal{S}$ executes the withdrawal protocol in interaction with $\mathcal{A}$ and obtains a wallet $\mathcal{W}$ of value 2^L. At any time, $\mathcal{A}$ can also ask $\mathcal{S}$ to run the spend protocol for some previously obtained wallet $\mathcal{W}_f$. At each such query, $\mathcal{A}$ chooses a string $\mathtt{info}$, integers $f \in [1, q_w]$ (where q_w denotes the number of past executions of the withdrawal protocol), and an amount $v = 2^\ell$, such that $\ell \leq L$. If f is not the index of a valid wallet $\mathcal{W}_f$, $\mathcal{S}$ outputs $\perp$. Otherwise, it runs $\mathsf{Spend}(\mathsf{params}, pk_{\mathcal{B}}, f, 2^\ell, pk_{\mathcal{M}}, \mathtt{info})$. We assume w.l.o.g. that $\mathcal{A}$ makes q_w withdrawal queries and q_s spend queries for each wallet $\{\mathcal{W}_f\}_{f=1}^{q_w}$ (otherwise, $\mathcal{S}$ can simulate spend queries for itself). It is assumed that $\mathcal{A}$ is legitimate and never asks $\mathcal{S}$ to double-spend a coin. More precisely, if $\mathcal{A}$

requires $\mathcal{S}$ to spend a coin labeled by x_{coin} of wallet $\mathcal{W}_f$, then $\mathcal{A}$ never asks $\mathcal{S}$ to spend a coin associated with any descendant or any ancestor of node x_{coin} in the tree associated to wallet $\mathcal{W}$. At the end, $\mathcal{A}$ outputs $d \in \{0,1\}$ and we call E_1 the event that $d = 1$.

Game 2: is like Game 0 with the difference that the output of the setup procedure CashSetup is replaced by the output of SimCashSetup. Under the SXDH assumption, this change is not noticeable to $\mathcal{A}$ and we have $|\Pr[E_2] - \Pr[E_1]| \in \mathsf{negl}(\lambda)$.

Game 3: is like Game 2 but at each run of the spending algorithm, the execution of SigProve (in step 1 one of Spend) replaces the real proof π^{sig} by a simulated NIZK proof (which $\mathcal{S}$ produces using the trapdoor $\tau = (\xi_1, \xi_2)$ of the fake CRS). This implies that commitments $\{C_{S,i}\}_{i=1}^3, \{C_{T,i}\}_{i=1}^3, \{C_{\mathcal{U},i}\}_{i=1}^3$ are all calculated as commitments to $\{1_{\mathbb{G}_2}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2}\}$. On the other hand, $C_{K_\mathcal{U}}$ is still computed as a GS commitment to $K_\mathcal{U} = h^{sk_\mathcal{U}}$ and the trapdoor τ allows simulating a proof $\pi_{K_\mathcal{U}}$ that $C_{\mathcal{U},1}$ and $C_{K_\mathcal{U}}$ open to the same value (*i.e.*, a proof that relations (1) hold). Since, on a simulated CRS, all commitments are perfectly hiding and NIZK proofs have the same distribution as real proofs, $\mathcal{A}$'s view does not change and we have $\Pr[E_3] = \Pr[E_2]$.

Game 4: is as Game 3 but, at each execution of Spend, NIZK proofs π_{Y_j}, π_{j,T_1}, π_{j,T_3}, π_{j,T_5} (at step 2.b) and $\pi_{Y_{L-\ell}}$, $\pi_{Z_{L-\ell}}$ (at step 2.c) are simulated using τ. More precisely, $\mathcal{S}$ proceeds as follows:

- Simulation of π_{Y_j}: at step b, $\mathcal{S}$ defines $Y_j = \eta_j = g^{1/(s+x_j)}$, for $j = 0$ to $L - \ell - 1$, and computes C_{Y_j} as commitments to Y_j. For $j = 0$ to $L - \ell - 1$, it picks $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and computes

$$T_{j,1} = h^{\delta_{j,1}}, \qquad T_{j,3} = h^{\delta_{j,2}}, \qquad T_{j,5} = h^{\delta_{j,3}},$$
$$T_{j,2} = e(\eta_j, T_{j,1}), \qquad T_{j,4} = pk_\mathcal{U} \cdot e(\eta_j, T_{j,3}), \qquad T_{j,6} = pk_\mathcal{U}^{\log_g(g_0)} \cdot e(\eta_j, T_{j,5}).$$

  Since C_{X_j} is a commitment to $X_j = h^{x_j}$ and $C_{S,1}$ is a commitment to $L_{S,1} = 1_{\mathbb{G}_2}$, setting $\Phi_{Y_j} = g^{1/x_j}$ allows generating a proof that $e(\Phi_{Y_j}, L_{S,1} \cdot X_j) = e(g,h)$ whereas a fake proof that $e(Y_j/\Phi_{Y_j}, h^\theta) = 1_{\mathbb{G}_T}$ can be simulated using the trapdoor τ that allows opening to 0 (and use that opening in the proof of the latter relation) the commitment to $\theta = 1$. We observe that the simulation of π_{Y_j} does not rely on the fact that $\eta_j = g^{1/(s+x_j)}$: it would also work with any $\eta_j \in \mathbb{G}_1$.

- Simulation of π_{j,T_1}, π_{j,T_3} and π_{j,T_5}: $\mathcal{S}$ defines $\Phi_{j,1} = T_{j,1}$, $\Phi_{j,3} = T_{j,3}$ and $\Phi_{j,5} = T_{j,5}$. It also sets $\Phi_{Y'_j} = \Phi_{Y''_j} = \eta_j$. The proof π_{j,T_1} that $T_{j,2} = e(Y_j, T_{j,1})$ is obtained using the witnesses $Y_j = \eta_j$ and $\Phi_{j,1}$ just like the proof π_{j,T_3} that $T_{j,4} = e(g, K_\mathcal{U}) \cdot e(Y_j, \Phi_{j,3})$ and $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$. The proof π_{j,T_5} is also generated as a real proof and so are the proofs that $e(Y_j/\Phi_{Y'_j}, h^\theta) = e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}$.

- Simulation of $\pi_{Y_{L-\ell}}$ and $\pi_{Z_{L-\ell}}$: $\mathcal{S}$ defines $Y_{L-\ell} = \eta_{L-\ell} = g^{1/(s+x_{L-\ell})}$ and computes $Z_{L-\ell}$ as $Z_{L-\ell} = g^{sk_\mathcal{U}} \cdot W_{L-\ell}^R$, where $W_{L-\ell} = \varphi_{L-\ell} = g^{1/(t+x_{L-\ell})}$. Since $C_{T,1}$ is a commitment to $1_{\mathbb{G}_2}$ and $C_{X_{L-\ell}}$ is a commitment to $X_{L-\ell} = h^{x_{L-\ell}}$, the assignment $\Phi_{Y_{L-\ell}} = \Phi_{W_{L-\ell}} = g^{1/x_{L-\ell}}$ allows generating proofs that

$$e(\Phi_{Y_{L-\ell}}, L_{S,1} \cdot X_{L-\ell}) = e(g,h) \qquad \text{and} \qquad e(\Phi_{W_{L-\ell}}, L_{T,1} \cdot X_{L-\ell}) = e(g,h).$$

  As for the fake proofs that $e(Y_{L-\ell}/\Phi_{Y_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(W_{L-\ell}/\Phi_{W_{L-\ell}}, h^\theta) = 1_{\mathbb{G}_T}$, they are simulated by trapdoor opening to 0 the commitment to $\theta = 1$. Finally, the proof for the

relation $e(Z_{L-\ell}, h) = e(g, K_{\mathcal{U}}) \cdot e(W_{L-\ell}, h^R)$ is obtained using the real witnesses $K_{\mathcal{U}}$ and $W_{L-\ell} = \varphi_{L-\ell}$. Note that, while $Y_{L-\ell}$ and $W_{L-\ell}$ are distributed as in the real game, the simulation would still work if they had arbitrary values in $\mathbb{G}_1$.

At this stage, the PRF values values $\{\eta_j\}_{j=0}^{L-\ell}$ and $\varphi_{L-\ell}$ are calculated as in the real game. However, the proofs that PRFs were correctly evaluated are all replaced by simulated proofs. Since these proofs are perfectly indistinguishable from proofs generated using real witnesses, $\mathcal{A}$'s view is not affected and we have $\Pr[E_4] = \Pr[E_3]$.

In the following, we consider a sequence of $q_w + 1$ hybrid games where, for each wallet $\mathcal{W}_f$ that $\mathcal{S}$ obtained after an execution of the withdrawal protocol, PRF outputs are gradually replaced by uniformly random values of the same group.

Game $5.k$**:** $(0 \leq k \leq q_w)$ is defined as follows:

- All spend queries involving wallet $\mathcal{W}_f$ (for $f \leq k$), the pseudorandom values $\eta_j = g^{1/(s+x_j)}$ and $\varphi_{L-\ell} = g^{1/(t+x_{L-\ell})}$ (where $j \in [0, L-\ell]$ and $i \in [1, L]$) are replaced by truly random values $\eta_j \xleftarrow{R} \mathbb{G}_1$ and $\varphi_{L-\ell} \xleftarrow{R} \mathbb{G}_1$. In the case of $\{\eta_j\}_{j=0}^{L-\ell}$, the replacement is done consistently in that, if the same value $\eta_j \in \mathbb{G}_1$ has to be involved (in the pairs $(T_{j,1}, T_{j,2}) = (T_{j,1}, e(\eta_j, T_{j,2}))$ and $(T_{j,3}, T_{j,4}) = (T_{j,3}, pk_{\mathcal{U}} \cdot e(\eta_j, T_{j,3}))$ in two distinct spend queries, $\mathcal{S}$ re-uses the same random value $\eta_j \in_R \mathbb{G}_1$.
- In contrast, all PRF values η_j and $\varphi_{L-\ell}$ for other wallets $\mathcal{W}_f$ (for $f > k$) remain pseudorandom and are still computed as $\eta_j = g^{1/(s+x_j)}$ and $\varphi_{L-\ell} = g^{1/(t+x_{L-\ell})}$, respectively.

The pseudorandomness of the Dodis-Yampolskiy PRF in $\mathbb{G}_1$ guarantees that, Game $5.k$ is indistinguishable from Game $5.(k-1)$ if we define Game 5.0 to be identical to Game 4. Under the 2^{L+2}-DDHI assumption in $\mathbb{G}_1$ (since from Game $5.k$ to Game $5.(k-1)$, we make at most 2^{L+2} replacements) where L is the maximal height of the tree involved in the k-th withdrawal query and by an hybrid argument, we have $|\Pr[E_{5.q_w}] - \Pr[E_4]| \in \mathsf{negl}(\lambda)$.

In Game $5.q_w$, two distinct spend queries for a given wallet still involve pairs

$$\begin{aligned}
(T_{j,1}, T_{j,2}) &= \big(h^{\delta_{j,1}}, e(\eta_j, T_{j,1})\big),\\
(T_{j,3}, T_{j,4}) &= \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta_j, T_{j,1})\big)\\
(T_{j,5}, T_{j,6}) &= \big(h^{\delta_{j,3}}, pk_{\mathcal{U}}^{\log_g(g_0)} \cdot e(\eta_j, T_{j,5})\big)
\end{aligned}$$

that depend on the same random η_j and we have to make sure that $\mathcal{A}$ is not able to link them. To this end, we need to gradually replace the random values η_j in $T_{j,2}$, $T_{j,4}$ and $T_{j,6}$ by other random values that are chosen independently of pairs $(T_{j,1}, T_{j,2})$, $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$ that were previously generated for the same node. For simplicity, we assume w.l.o.g. that the adversary $\mathcal{A}$ makes exactly q_s $\mathcal{Q}_{\mathsf{Spend}}$-queries for each withdrawn wallet. We thus define a sub-sequence of hybrid games starting with Game $6.(0,0,0)$, which is identical to $5.q_w$, and ending with Game $6.(q_w, q_s, L)$.

Game $6.(f,k,l)''$ **(**$1 \leq f \leq q_w$**,** $1 \leq k \leq q_s$**,** $1 \leq l \leq L$**):** is a hybrid game where all spend queries involving the f-th withdrawn wallet $\mathcal{W}_f$ are treated in a way that depends on the index $q \in [1, q_s]$ of the query among those involving $\mathcal{W}_f$. More precisely, consider the q-th Q_{Spend}-query that

involves the ζ-th withdrawn wallet. Let x_{coin} be the label of the coin and let $\{x_0, \ldots, x_{L-\ell}\}$ be the path from the root to $x_{L-\ell} = x_{coin}$ (note that $L - \ell$ may be as large as L).

1. If $\zeta \in [f+1, q_w]$ or $q \in [k+1, q_s]$, for each node $j \in \{0, \ldots, L-\ell\}$, $\mathcal{S}$ generates pairs

$$\begin{aligned}(T_{j,1}, T_{j,2}) &= \big(h^{\delta_{j,1}}, e(\eta_j, T_{j,1})\big),\\ (T_{j,3}, T_{j,4}) &= \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta_j, T_{j,1})\big)\\ (T_{j,5}, T_{j,6}) &= \big(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta_j, T_{j,5})\big)\end{aligned}$$

   for random $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and using a persistent random value $\eta_j \in \mathbb{G}_1$ which is assigned once-and-for-all to node j at the beginning of the game.
2. If $\zeta \in [1, f-1]$ or $q \in [1, k-1]$, for each $j \in \{0, \ldots, L-\ell\}$, $\mathcal{S}$ rather computes

$$(T_{j,1}, T_{j,2}) = \big(h^{\delta_{j,1}}, e(\eta, T_{j,1})\big), \qquad (T_{j,3}, T_{j,4}) = \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta', T_{j,1})\big),$$
$$(T_{j,5}, T_{j,6}) = \big(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta'', T_{j,5})\big)$$

   for uniformly chosen $\eta, \eta', \eta'' \xleftarrow{R} \mathbb{G}_1$ that are freshly chosen for each node j.
3. If $\zeta = f$ and $q = k$, $\mathcal{S}$ considers the path $\{x_0, \ldots, x_{L-\ell}\}$ and, for each $j \in \{0, \ldots, L-\ell\}$, the way to compute $(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6})$ depends on j:
   - If $j < l$, $\mathcal{S}$ computes $(T_{j,1}, T_{j,2})$, $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$ using newly chosen uniformly chosen $\eta, \eta', \eta'' \xleftarrow{R} \mathbb{G}_1$ as in case 2.
   - If $j > l$, $\mathcal{S}$ proceeds as in case 1 and computes $(T_{j,1}, T_{j,2})$, $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$ using the persistent random values $\eta_j \in \mathbb{G}_1$ chosen at the beginning of the game.
   - If $j = l$, $\mathcal{S}$ computes

$$\begin{aligned}(T_{j,1}, T_{j,2}) &= \big(h^{\delta_{j,1}}, e(\eta, T_{j,1})\big),\\ (T_{j,3}, T_{j,4}) &= \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta, T_{j,1})\big)\\ (T_{j,5}, T_{j,6}) &= \big(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta, T_{j,5})\big)\end{aligned}$$

     for a random $\eta \xleftarrow{R} \mathbb{G}_1$ which is chosen independently of η_j.

Game $6.(f,k,l)'$**:** exactly as Game $6.(f,k,l)''$ except that if $\zeta = f$, $q = k$ and $j = l$, $\mathcal{S}$ computes

$$\begin{aligned}(T_{j,1}, T_{j,2}) &= \big(h^{\delta_{j,1}}, e(\eta, T_{j,1})\big),\\ (T_{j,3}, T_{j,4}) &= \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta, T_{j,1})\big)\\ (T_{j,5}, T_{j,6}) &= \big(h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta', T_{j,1})\big)\end{aligned}$$

using completely random $\eta, \eta' \xleftarrow{R} \mathbb{G}_1$.

Game $6.(f,k,l)$**:** exactly as in previous game except that if $\zeta = f$, $q = k$ and $j \leq l$, the simulator $\mathcal{S}$ now computes

$$\begin{aligned}(T_{j,1}, T_{j,2}) &= \big(h^{\delta_{j,1}}, e(\eta, T_{j,1})\big),\\ (T_{j,3}, T_{j,4}) &= \big(h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta', T_{j,1})\big)\\ (T_{j,5}, T_{j,6}) &= \big(h^{\delta_{j,2}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta'', T_{j,1})\big)\end{aligned}$$

using independent random $\eta, \eta', \eta'' \xleftarrow{R} \mathbb{G}_1$ that are not used anywhere else.

Note that, in Game 6.(q_w, q_s, L), all spend queries are answered by outputting pairs $(T_{j,1}, T_{j,2})$ that are completely uncorrelated to the values $(T_{j,1}, T_{j,2})$ appearing in other occurrences of the same node. The same holds for pairs $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$. In addition, for each j, the relationship between $T_{j,1}$ and $T_{j,2}$ is independent from the relationship between $T_{j,3}$ and $T_{j,4}$, which is itself independent from how $T_{j,6}$ relates to $T_{j,5}$.

Lemma 1. *If the D3DH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, for each $f \in \{1, \ldots, q_w\}$, $k \in \{1, \ldots, q_s\}$ and $l \in \{1, \ldots, L\}$, no PPT adversary can tell apart Game 6.$(f, k, l-1)$ and Game 6.$(f, k, l)''$*

Proof. Let $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, \Gamma \stackrel{?}{=} g^{abc})$ be a D3DH instance. Assuming that an adversary $\mathcal{A}$ can distinguish Game 6.$(f, k, l-1)$ and Game 6.$(f, k, l)''$ with non-negligible advantage, we construct a distinguisher $\mathcal{D}$ for the D3DH problem. Namely, $\mathcal{D}$ interacts with $\mathcal{A}$ as the simulator $\mathcal{S}$ does in withdrawal protocols in Game 5.q_w. For each pair $(\zeta, q) \in [1, q_w] \times [1, q_s]$, the q-th spend query related to wallet $\mathcal{W}_\zeta$ is answered as follows. The distinguisher $\mathcal{D}$ considers the path $\{x_0, \ldots, x_{L-\ell}\}$, where $x_{L-\ell} = x_{coin}$ is the label of the coin to be spent. For $j = 0$ to $L - \ell$, $\mathcal{D}$ considers the node x_j on the path $\{x_0, \ldots, x_{L-\ell}\}$ and does the following.

Case $\zeta > f$ or $q > k$ or ($\zeta = f$, $q = k$ and $j > l$): $\mathcal{D}$ first checks if a value $\mu_j \in \mathbb{Z}_p$ was previously defined for node x_j. If so, it recalls the previously defined value. Otherwise, $\mathcal{D}$ chooses $\mu_j \xleftarrow{R} \mathbb{Z}_p$ and retains it for later queries involving the same node. It also picks $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and computes $T_{j,1} = h^{\delta_{j,1}}$, $T_{j,2} = e(g^a, h^b)^{\delta_{j,1} \cdot \mu_j}$ as well as $T_{j,3} = h^{\delta_{j,2}}$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(g^a, h^b)^{\delta_{j,2} \cdot \mu_j}$ and $T_{j,5} = h^{\delta_{j,3}}$, $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(g^a, h^b)^{\delta_{j,3} \cdot \mu_j}$, which implicitly defines $\eta_j = g^{ab \cdot \mu_j}$.

To simulate the proofs π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, the distinguisher $\mathcal{D}$ computes C_{Y_j}, $C_{\Phi_{j,1}}$, $C_{\Phi_{j,3}}$, $C_{\Phi_{j,5}}$ using the assignment $Y_j = \Phi_{Y'_j} = \Phi_{Y''_j} = g^{a\mu_j}$, $\Phi_{j,1} = h^{b \cdot \delta_{j,1}}$, $\Phi_{j,3} = h^{b \cdot \delta_{j,2}}$, $\Phi_{j,5} = h^{b \cdot \delta_{j,3}}$ which satisfies the relations $T_{j,2} = e(Y_j, \Phi_{j,1})$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3})$ and $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5})$.

Then, the trapdoor τ of the fake params_{GS} allows generating proofs that $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$ by using the witness $\theta = 0$ for the latter three relations and the witness $\theta = 1$ in other relations. We note that the proof π_{Y_j} can still be simulated by defining $\Phi_{Y_j} = g^{1/x_j}$ as in Game 4.

Case $(\zeta, q, j) = (f, k, l)$: $\mathcal{D}$ checks if a value $\mu_j \in \mathbb{Z}_p$ was previously defined for node x_j. If yes, it recalls the previously defined $\mu_j \in \mathbb{Z}_p$ and chooses it at random otherwise. Then, $\mathcal{D}$ picks $\tilde{\delta}_{j,1}, \tilde{\delta}_{j,2} \xleftarrow{R} \mathbb{Z}_p$ and sets

$$T_{j,1} = (h^c)^{\tilde{\delta}_{j,1}}, \qquad T_{j,3} = (h^c)^{\tilde{\delta}_{j,2}}, \qquad T_{j,5} = (h^c)^{\tilde{\delta}_{j,3}},$$

as well as

$$T_{j,2} = e(\Gamma, h)^{\tilde{\delta}_{j,1} \cdot \mu_j}, \qquad T_{j,4} = pk_{\mathcal{U}} \cdot e(\Gamma, h)^{\tilde{\delta}_{j,2} \cdot \mu_j}, \qquad T_{j,6} = e(g, h^{sk_{\mathcal{U}}}) \cdot e(\Gamma, h)^{\tilde{\delta}_{j,3} \cdot \mu_j}. \tag{3}$$

In order to simulate the proofs π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, $\mathcal{D}$ generates commitments to the variable assignment

$$Y_j = \Phi_{Y'_j} = \Phi_{Y''_j} = \Gamma^{\mu_j}, \qquad \Phi_{j,1} = h^{\tilde{\delta}_{j,1}}, \qquad \Phi_{j,3} = h^{\tilde{\delta}_{j,2}}, \qquad \Phi_{j,5} = h^{\tilde{\delta}_{j,5}}.$$

Since $T_{j,2} = e(Y_j, \Phi_{j,1})$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3})$ and $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5})$, this assignment suffices to simulate proofs since the trapdoor τ allows computing convincing proofs that

$e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$ by trapdoor opening to 0 the commitment to $\theta = 1$. The proof π_{Y_j} is simulated by setting $\Phi_{Y_j} = g^{1/x_j}$ as in Game 4.

Case $(\zeta, q, j) < (f, k, l)$: the distinguisher $\mathcal{D}$ randomly picks $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and $\eta, \eta', \eta'' \xleftarrow{R} \mathbb{G}_1$. Then, it computes pairs

$$\begin{aligned}(T_{j,1}, T_{j,2}) &= (h^{\delta_{j,1}}, e(\eta, T_{j,1})),\\ (T_{j,3}, T_{j,4}) &= (h^{\delta_{j,2}}, pk_{\mathcal{U}} \cdot e(\eta', T_{j,1})),\\ (T_{j,5}, T_{j,6}) &= (h^{\delta_{j,3}}, e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta'', T_{j,5})).\end{aligned}$$

In order to generate the proofs π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, $\mathcal{D}$ first commits to the variables $Y_j = \eta$, $\Phi_{Y'_j} = \eta'$, $\Phi_{Y''_j} = \eta''$ $\Phi_{j,1} = h^{\delta_{j,1}}$, $\Phi_{j,3} = h^{\delta_{j,2}}$ and $\Phi_{j,5} = h^{\delta_{j,3}}$. Given that $T_{j,2} = e(Y_j, \Phi_{j,1})$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3})$ and since it also holds that $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5})$, the proofs for $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$ can be honestly generated. As for the false statements $e(Y_j/\Phi_{Y'_j}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}$, they can be proved using the trapdoor τ.

It is easy to see that, if $\eta = g^{abc}$, then $\eta_j = g^{ab \cdot \mu_j}$, so that $\mathcal{D}$ is playing Game 6.$(f, k, l-1)$ with $\mathcal{A}$. If $\eta \in_R \mathbb{G}_1$, we can write $\eta = g^{c \cdot z}$ for some $z \in_R \mathbb{Z}_p$ such that $z \neq ab$ with overwhelming probability. Then, relations (3) imply that $\eta_j = z \cdot \mu_j$ for each j. In this case, $\mathcal{D}$ is playing Game 6.$(f, k, l)''$ with $\mathcal{A}$. □

Lemma 2. *If the D3DH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$*, for each* $f \in \{1, \ldots, q_w\}$*,* $k \in \{1, \ldots, q_s\}$ *and* $l \in \{1, \ldots, L\}$*, no PPT adversary can distinguish Game* 6.$(f, k, l)''$ *and Game* 6.$(f, k, l)'$.

Proof. Let $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, \Gamma \stackrel{?}{=} g^{abc})$ be a D3DH instance taken as input by $\mathcal{D}$. Assuming that $\mathcal{A}$ can distinguish Game 6.$(f, k, l)''$ and Game 6.$(f, k, l)'$, we construct $\mathcal{D}$ a distinguisher for the D3DH problem. Namely, $\mathcal{D}$ interacts with $\mathcal{A}$ as $\mathcal{S}$ does in withdrawal protocols in Game 5.q_w. For all pairs $(\zeta, q) \in [1, q_w] \times [1, q_s]$, $\mathcal{D}$ handles the q-th spend query related to wallet $\mathcal{W}_\zeta$ in the following way. It considers to path $\{x_0, \ldots, x_{L-\ell}\}$ that connects the expandable coin $x_{coin} = x_{L-\ell}$ to the root. For $j = 0$ to $L - \ell$, it computes the following values.

Case $\zeta > f$ or $q > k$ or ($q = k$ and $j > l$): $\mathcal{D}$ proceeds analogously to the similar case in the proof of Lemma 1. Namely, $\mathcal{D}$ first checks if a value μ_j was previously defined for the node x_j. If yes, $\mathcal{D}$ recalls the previously defined $\mu_j \in \mathbb{Z}_p$. Otherwise, $\mathcal{D}$ picks $\mu_j \xleftarrow{R} \mathbb{Z}_p$ and stores it for later use. It also picks $\delta_{j,1}, \delta_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and computes the elements $T_{j,1} = h^{\delta_{j,1}}$, $T_{j,2} = e(g, h)^{\delta_{j,1} \cdot \mu_j}$, $T_{j,3} = h^{\delta_{j,2}}$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(g, h)^{\delta_{j,2} \cdot \mu_j}$, $T_{j,5} = h^{\delta_{j,3}}$ and finally $T_{j,6} = (g_0, h^{sk_{\mathcal{U}}}) \cdot e(g, h)^{\delta_{j,3} \cdot \mu_j}$, which implicitly defines the random function in such a way that $\eta_j = g^{\mu_j}$. To simulate the proofs π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, $\mathcal{D}$ computes commitments C_{Y_j}, $C_{\Phi_{j,1}}$, $C_{\Phi_{j,3}}$ and $C_{\Phi_{j,5}}$ to the variables

$$Y_j = \Phi_{Y'_j} = \Phi_{Y''_j} = g^{\mu_j}, \qquad \Phi_{j,1} = h^{\delta_{j,1}}, \qquad \Phi_{j,3} = h^{\delta_{j,2}}, \qquad \Phi_{j,5} = h^{\delta_{j,5}},$$

which satisfy the equalities

$$T_{j,2} = e(Y_j, \Phi_{j,1}), \qquad T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3}), \qquad T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5}).$$

Then, $\mathcal{D}$ can generate real proofs that $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$.

Case $(\zeta, q, j) = (f, k, l)$: $\mathcal{D}$ picks $\delta_{j,1}, \delta_{j,2}, \tilde{\delta}_{j,3} \xleftarrow{R} \mathbb{Z}_p$ and sets

$$T_{j,1} = h^{\delta_{j,1}}, \qquad T_{j,3} = h^{\delta_{j,2}}, \qquad T_{j,5} = (h^c)^{\tilde{\delta}_{j,3}},$$
$$T_{j,2} = e(g^a, h^b)^{\delta_{j,1}}, \qquad T_{j,4} = pk_{\mathcal{U}} \cdot e(g^a, h^b)^{\delta_{j,2}}, \qquad T_{j,6} = pk_{\mathcal{U}} \cdot e(\Gamma, h)^{\tilde{\delta}_{j,3}}.$$

In order to simulate π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, $\mathcal{D}$ generates commitments to the variable assignment $Y_j = \Phi_{Y'_j} = g^a$, $\Phi_{Y''_j} = \Gamma$, $\Phi_{j,1} = h^{b\delta_{j,1}}$, $\Phi_{j,3} = h^{b\delta_{j,2}}$, $\Phi_{j,5} = h^{\tilde{\delta}_{j,3}}$. Since this variable assignment satisfies $T_{j,2} = e(Y_j, \Phi_{j,1})$, $T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3})$ and $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5})$, $\mathcal{D}$ just has to simulate proofs that $e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$ by opening to 0 the commitment to $\theta = 1$.

Case $(\zeta, q, j) < (f, k, l)$: this case is handled exactly as in the proof of Lemma 1.

We observe that, if $\eta = g^{abc}$, $\mathcal{D}$ is playing Game $6.(f, k, l)''$. In the situation where $\eta \in_R \mathbb{G}_1$, $\mathcal{D}$ is playing Game $6.(f, k, l)'$. □

Lemma 3. *For each $k \in \{1, \ldots, q_s\}$ and $l \in \{1, \ldots, L\}$, no PPT adversary can distinguish Game $6.(f, k, l)'$ from Game $6.(f, k, l)$ if the D3DH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Proof. Let $(g, g^a, g^b, g^c, h, h^a, h^b, h^c, \Gamma \stackrel{?}{=} g^{abc})$ be a D3DH instance taken as input by the distinguisher $\mathcal{D}$. We show that, if $\mathcal{A}$ can distinguish Game $6.(f, k, l)'$ and Game $6.(f, k, l)$, $\mathcal{D}$ is a successful D3DH distinguisher. The latter interacts with the adversary $\mathcal{A}$ in the same way as in the proofs of Lemmas 1 and 2. For all pairs (η, q), the q-th spend query involving the wallet $\mathcal{W}_\zeta$ is handled as follows. The distinguisher $\mathcal{D}$ considers the coin to be spent x_{coin} and the path $\{x_0, \ldots, x_{L-\ell}\}$, where $x_{L-\ell} = x_{coin}$. For each $j \in \{0, \ldots, L - \ell\}$, it does the following.

Case $\zeta > f$ or $q > k$ or ($q = k$ and $j > l$): $\mathcal{D}$ proceeds exactly as in the similar case in the proof of Lemma 2.

Case $(\zeta, q, j) = (f, k, l)$: $\mathcal{D}$ picks $\delta_{j,1}, \tilde{\delta}_{j,2}, \delta_{j,3} \xleftarrow{R} \mathbb{Z}_p$ as well as $\eta'' \xleftarrow{R} \mathbb{G}_1$ and sets

$$T_{j,1} = h^{\delta_{j,1}}, \qquad T_{j,3} = (h^c)^{\tilde{\delta}_{j,2}}, \qquad T_{j,5} = h^{\delta_{j,3}},$$
$$T_{j,2} = e(g^a, h^b)^{\delta_{j,1}}, \qquad T_{j,4} = pk_{\mathcal{U}} \cdot e(\Gamma, h)^{\tilde{\delta}_{j,2}}, \qquad T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta'', h)^{\delta_{j,3}}.$$

To simulate proofs π_{j,T_1}, π_{j,T_3} and π_{j,T_5}, $\mathcal{D}$ generates commitments to the variable assignment

$$Y_j = g^a, \qquad \Phi_{Y'_j} = \Gamma, \qquad \Phi_{Y''_j} = \eta'',$$
$$\Phi_{j,1} = h^{b \cdot \delta_{j,1}}, \qquad \Phi_{j,3} = h^{\tilde{\delta}_{j,2}}, \qquad \Phi_{j,5} = h^{\delta_{j,3}}.$$

Since this variable assignment satisfies the equalities

$$T_{j,2} = e(Y_j, \Phi_{j,1}), \qquad T_{j,4} = pk_{\mathcal{U}} \cdot e(\Phi_{Y'_j}, \Phi_{j,3}), \qquad T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\Phi_{Y''_j}, \Phi_{j,5}),$$

$\mathcal{D}$ only needs to simulate fake proofs for the relations $e(Y_j/\Phi_{Y'_j}, h^\theta) = e(Y_j/\Phi_{Y''_j}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,1}/\Phi_{j,1}, h^\theta) = 1_{\mathbb{G}_T}$, $e(T_{j,3}/\Phi_{j,3}, h^\theta) = 1_{\mathbb{G}_T}$ and $e(T_{j,5}/\Phi_{j,5}, h^\theta) = 1_{\mathbb{G}_T}$ by opening to 0 the commitment to $\theta = 1$.

- **Case** $(\zeta, q, j) < (f, k, l)$: this case is dealt as in the proof of lemma 2.

It is easy to see that, if $\eta = g^{abc}$, $\mathcal{D}$ is playing Game 6.$(f, k, l)'$ with the adversary. In the case $\eta \in_R \mathbb{G}_1$, $\mathcal{D}$ is rather playing Game 6.(f, k, l) with $\mathcal{A}$. □

In Game 6.(q_w, q_s, L), $\mathcal{A}$ only gets to see a sequence of uncorrelated values at each spend query. Moreover, whenever $\mathcal{A}$ observes pairs $(T_{j,1}, T_{j,2})$, $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$, elements $T_{j,4}$ and $T_{j,6}$ have the form $T_{j,4} = pk_{\mathcal{U}} \cdot e(\eta', T_{j,3})$, $T_{j,6} = e(g_0, h^{sk_{\mathcal{U}}}) \cdot e(\eta'', T_{j,3})$ for completely random $\eta', \eta'' \xleftarrow{R} \mathbb{G}_1$ that do not appear anywhere else. It means that $(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6})$ perfectly hide the user's public key $pk_{\mathcal{U}}$. For this reason, changing the commitment $C_{K_{\mathcal{U}}}$ into a commitment to $K_{\mathcal{U}} = 1_{\mathbb{G}_2}$ and computing $T_{j,4}$ and $T_{j,6}$ as $e(\eta', T_{j,3})$ and $e(\eta'', T_{j,5})$, respectively, does not alter $\mathcal{A}$'s view since all these values have the same distribution either way.

Game 7: is as Game 6.(q_w, q_s, L) but the commitment $C_{K_{\mathcal{U}}}$ is now computed as a commitment to $K_{\mathcal{U}} = 1_{\mathbb{G}_2}$ at step 1 of each spending query. In addition, for each $j \in \{0, \ldots, L - \ell\}$, the pairs $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$ are now computed as $T_{j,4} = e(\eta', T_{j,3})$ and $T_{j,6} = e(\eta'', T_{j,5})$, for completely random $\eta', \eta'' \xleftarrow{R} \mathbb{G}_1$. Since the distribution of $C_{K_{\mathcal{U}}}$, $(T_{j,3}, T_{j,4})$ and $(T_{j,5}, T_{j,6})$ does not change, we necessarily have $\Pr[E_{6.(q_w,,q_s,L)}] = \Pr[E_7]$.

In Game 7, $\mathcal{S}$ simulates $\mathcal{A}$'s view without using users' private keys at any time and without the PRF seeds contained in any wallet. To render $\mathcal{S}$ completely independent of wallets $\mathcal{W}_1, \ldots, \mathcal{W}_{q_w}$ however, we still have to eliminate the use of state for each wallet.

Game 8: is as Game 7 with the difference that, at each spending query, instead of determining x_{coin} as a function of the internal state state of the wallet, $\mathcal{S}$ arbitrarily chooses x_{coin} as a random node at height ℓ in the tree. At step 2, of the simulated execution of spend, commitments C_{X_j} are then calculated as commitments to nodes $x_j \in \{x_0, \ldots, x_{coin}\}$ and proofs π_{b_j} are honestly generated. Since all commitments C_{X_i} are perfectly hiding and proofs π_{b_j} are perfectly WI, the view of $\mathcal{A}$ does not change and we must have $\Pr[E_8] = \Pr[E_7]$.

The whole sequence of games thus implies that, under the SXDH, 2^{L+2}-DDHI and D3DH assumptions, we have $|\Pr[E_1] - \Pr[E_8]| \in \mathsf{negl}(\lambda)$, which implies that the simulated spending oracle is computationally indistinguishable from the real spending oracle. □

A.2 Balance

Theorem 4 (Balance). *The scheme satisfies the balance property under the q_w-HSDH, and TDH assumptions in $(\mathbb{G}_1, \mathbb{G}_2)$, where q_w is the number of $\mathcal{Q}_{\mathsf{withdraw}}$ queries.*

Proof. Let $\mathcal{A}$ be a successful adversary in the *Balance* experiment and $\mathcal{B}$ the simulator which simulates oracle queries. We gradually modify the view of $\mathcal{A}$ in each of the following games.

Game 0: this is the real security game.

Game 1: in this Game, $\mathcal{B}$ modifies the simulation of oracle $\mathcal{Q}_{\mathsf{withdraw}}$ so as to control the coin values $(s, t, sk_{\mathcal{U}})$ that are signed in each wallet. Recall that, at each $\mathcal{Q}_{\mathsf{withdraw}}$ query, the executed interactive protocol consists of three parts: the first two steps combine the contributions of the bank and the user to construct s, t and the last part is an execution of the $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ protocol between $\mathcal{B}$ and $\mathcal{A}$ who acts as the adversarially-controlled user. In the first step of

the withdrawal, $\mathcal{A}$ provides $\mathcal{B}$ with proofs of knowledge of $(s', t', sk_{\mathcal{U}})$ and openings of commitments $C_{s'}$, $C_{t'}$ and $C_{sk_{\mathcal{U}}}$. At this step, $\mathcal{B}$ rewinds the prover $\mathcal{A}$ so as to extract the values $(s', t', sk_{\mathcal{U}})$ in each interactive proof. Then, $\mathcal{B}$ picks $s'', t'' \xleftarrow{R} \mathbb{Z}_p$ which are sent to $\mathcal{A}$. Next, $\mathcal{B}$ runs the signing algorithm Sign of the F-unforgeable to obtain a signature $(\sigma_1, \sigma_2, \sigma_3)$ on $\vec{m} = (s' + s'', t' + t'', sk_{\mathcal{U}})$. In the two-party protocol $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$, $\mathcal{B}$ simulates $\mathcal{A}$'s view and makes sure that the protocol ends with $\mathcal{A}$ obtaining $(\sigma_1, \sigma_2, \sigma_3)$. Then, $\mathcal{B}$ adds firstly $(s' + s'', t' + t'', sk_{\mathcal{U}})$ to a wallet database DB_W and secondly the user's public key $pk_{\mathcal{U}}$ in the user database $\mathsf{T}_{\mathcal{U}}$ (which is initially empty). Clearly, $\mathcal{A}$'s view is identical to her view in Game 0.

Game 2: in this game, $\mathcal{B}$ modifies the simulation of oracle $\mathcal{Q}_{\mathsf{deposit}}$, such that when receiving query $(\mathsf{params}, pk_{\mathcal{B}}, pk_{\mathcal{M}}, coin, 2^{\ell}, \mathrm{DB}_{\mathcal{B}})$, $\mathcal{B}$ parses the coin as

$$coin = \Big(\{C_{S,i}\}_{i=1}^{3}, \{C_{T,i}\}_{i=1}^{3}, \{C_{\mathcal{U},i}\}_{i=1}^{3},\ C_{K_{\mathcal{U}}},\ \pi_{K_{\mathcal{U}}},\ \pi^{sig},\ \{C_{X_j},\ C_{b_j},\ C'_{b_j},\ \pi_{x_j},\ \pi_{b_j,1}\ \pi_{b_j,2}\}_{j=1}^{L-\ell},$$
$$\{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}),\ C_{Y_j},\ C_{\Phi_{Y_j}},\ C'_{\Phi_{Y_j}},\ C''_{\Phi_{Y_j}},\ \{C_{\Phi_{j,k}}\}_{k\in\{1,3,5\}},\ \pi_{Y_j},\ \pi_{j,T_1},$$
$$\pi_{j,T_3},\ \pi_{j,T_5}\}_{j=0}^{L-\ell-1},\ Y_{L-\ell},\ Z_{L-\ell},\ C_{\Phi_{Y_{L-\ell}}}, C_{W_{L-\ell}},\ C_{\Phi_{W_{L-\ell}}},\ \pi_{Y_{L-\ell}},\ \pi_{Z_{L-\ell}},\ \mathtt{info}\Big)$$

for some wallet $\mathcal{W}$. We assume that all proofs properly verify (otherwise, $\mathcal{B}$ sets the proper flag) and let $\mathrm{DB}_{\mathcal{B}}$ be the database of deposited coins. Then, $\mathcal{B}$ uses the trapdoor of the perfectly sound Groth-Sahai CRS to extract the commitments $\{C_{S,i}\}_{i=1}^{3}, \{C_{T,i}\}_{i=1}^{3}, \{C_{\mathcal{U},i}\}_{i=1}^{3}$ (recall that extracting these commitments amounts to compute Elgamal decryptions in $\mathbb{G}_1$ or $\mathbb{G}_2$) and extracts the underlying components $\sigma = \{\sigma_\tau\}_{\tau=1}^{3}$ of the P-signature appearing in committed form as part of π^{sig}. At this step, $\mathcal{B}$ considers the extracted triples $(L_{S,1}, L_{S,2}, L_{S,3}) = (h^s, u^s, \tilde{V}_1^s)$, $(L_{T,1}, L_{T,2}, L_{T,3}) = (h^t, u^t, \tilde{V}_2^t)$ and $(L_{\mathcal{U},1}, L_{\mathcal{U},2}, L_{\mathcal{U},3}) = (h^{sk_{\mathcal{U}}}, u^{sk_{\mathcal{U}}}, \tilde{V}_3^{sk_{\mathcal{U}}})$. It checks whether the database DB_W contains a triple $(s, t, sk_{\mathcal{U}})$ such that $(h^s, h^t, h^{sk_{\mathcal{U}}}) = (L_{S,1}, L_{T,1}, L_{\mathcal{U},1})$ and aborts if no such triple is found.

If the above event occurs, it is easy to see that the F-unforgeability of the P-signature is broken. Under the HSDH and TDH assumptions, Game 2 and Game 1 are thus indistinguishable.

Game 3: is identical to Game 2 except that $\mathcal{B}$ aborts if the overall amount of deposited coins is more than the overall withdrawn amount. In Game 3, it is easy to see that, the global deposited amount is necessarily bounded by $q_w \cdot 2^L$ and no over-spent is possible.

It comes that the balance property is satisfied as long as the HSDH and TDH assumptions hold in $(\mathbb{G}_1, \mathbb{G}_2)$. □

A.3 Identification

Theorem 5 (Identification). *The scheme guarantees the identification of double-spenders assuming that H is a collision-resistant hash function and that the q_w-HSDH, TDH, SXDH, D3DH and 2^{L+1}-DDHI assumptions all hold in $(\mathbb{G}_1, \mathbb{G}_2)$, where q_w denotes the number of $\mathcal{Q}_{\mathsf{withdraw}}$ queries.*

Proof. Towards a contradiction, we assume that $\mathcal{A}$ is an adversary that can double-spend coins and escape identification with non-negligible probability. We show how one could $\mathcal{A}$ to construct algorithms that emulate the bank $\mathcal{B}$ and break either the security of the P-signature scheme or the pseudo-randomness of the Dodis-Yampolskiy PRF in $\mathbb{G}_1$. The proofs proceeds with the following sequence of games.

Game 0: this is the real security game.

Game 1: in this Game, $\mathcal{B}$ changes the simulation of $\mathcal{Q}_{\mathsf{withdraw}}$. Recall that, at each $\mathcal{Q}_{\mathsf{withdraw}}$ query, the withdrawal protocol consists of two steps: the first one combines the contributions of $\mathcal{B}$ and $\mathcal{U}$ to construct $(s,t) = (s'+s'', t'+t'')$ and the second part is a run of the $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ where $\mathcal{B}$ interacts with the malicious user $\mathcal{A}$. In the first step of the withdrawal protocol, the adversary $\mathcal{A}$ generates commitments $(C_{s'}, C_{t'}, C_{sk_{\mathcal{U}}})$ and proves knowledge of openings (which include $(s', t', sk_{\mathcal{U}})$) of these commitments. At this step, $\mathcal{B}$ rewinds the prover $\mathcal{A}$ so as to extract $(s', t', sk_{\mathcal{U}})$. Then, $\mathcal{B}$ sends random values (s'', t'') to $\mathcal{A}$ and runs algorithm Sign of the signature for $\vec{m} = (s'+s'', t'+t'', sk_{\mathcal{U}})$ and obtains a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$. Then, $\mathcal{B}$ simulates $\mathcal{A}$'s view in the execution of $\mathsf{SigIssue} \leftrightarrows \mathsf{SigObtain}$ and makes sure that $\mathcal{A}$ eventually obtains σ. Then, $\mathcal{B}$ stores $(s, t, sk_{\mathcal{U}})$ in its wallet database DB_W and adds the user public key $pk_{\mathcal{U}}$ in the database $\mathsf{T}_{\mathcal{U}}$ (which is initially empty). As in the proof of theorem 4, $\mathcal{A}$'s view is not altered by these changes.

Game 2: we bring another modification to oracle $\mathcal{Q}_{\mathsf{withdraw}}$. Remember that, at step 1 the withdrawal protocol, the adversary $\mathcal{A}$ provides commitments $(C_{s'}, C_{t'}, C_{sk_{\mathcal{U}}})$ and interactively proves that: (1) he knows how to open these commitments; (2) $C_{sk_{\mathcal{U}}}$ is a commitment to $sk_{\mathcal{U}}$ such that $pk_{\mathcal{U}} = e(g,h)^{sk_{\mathcal{U}}}$. We raise a failure event F_2 and let the challenger $\mathcal{B}$ abort if the extracted value $sk_{\mathcal{U}}$ – which $\mathcal{B}$ obtains by running the knowledge extractor of the proof system as in Game 1 – is such that $pk_{\mathcal{U}} \neq e(g,h)^{sk_{\mathcal{U}}}$. Clearly, Game 2 and Game 1 are identical until F_2 occurs and, in this situation, $\mathcal{A}$ must be able to break the soundness of the interactive proof system. As long as the latter is sound, Game 2 is thus indistinguishable from Game 1.

Game 3: we modify the simulation of oracle $\mathcal{Q}_{\mathsf{deposit}}$ as follows. At each $\mathcal{Q}_{\mathsf{deposit}}$ query, $\mathcal{B}$ parses the coin as

$$\begin{aligned} coin = \Big(& \{C_{S,i}\}_{i=1}^{3}, \{C_{T,i}\}_{i=1}^{3}, \{C_{\mathcal{U},i}\}_{i=1}^{3},\ C_{K_{\mathcal{U}}},\ \pi_{K_{\mathcal{U}}},\ \pi^{sig},\ \{C_{X_j},\ C_{b_j},\ C'_{b_j},\ \pi_{x_j},\ \pi_{b_j,1}\ \pi_{b_j,2}\}_{j=1}^{L-\ell}, \\ & \{(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}),\ C_{Y_j},\ C_{\Phi_{Y_j}},\ C'_{\Phi_{Y_j}},\ C''_{\Phi_{Y_j}},\ \{C_{\Phi_{j,k}}\}_{k\in\{1,3,5\}},\ \pi_{Y_j},\ \pi_{j,T_1}, \\ & \pi_{j,T_3},\ \pi_{j,T_5}\}_{j=0}^{L-\ell-1},\ Y_{L-\ell},\ Z_{L-\ell},\ C_{\Phi_{Y_{L-\ell}}}, C_{W_{L-\ell}},\ C_{\Phi_{W_{L-\ell}}},\ \pi_{Y_{L-\ell}},\ \pi_{Z_{L-\ell}},\ \mathtt{info}\Big). \end{aligned}$$

Then, $\mathcal{B}$ uses the trapdoor of the extractable Groth-Sahai CRS to extract the commitments $\{C_{S,i}\}_{i=1}^{3}$, $\{C_{T,i}\}_{i=1}^{3}$, $\{C_{\mathcal{U},i}\}_{i=1}^{3}$ and the committed P-signature components $\{C_{\sigma_\tau}\}_{\tau=1}^{3}$ which are part of π^{sig}. ¿From these, $\mathcal{B}$ obtains the underlying P-signature $\{\sigma_\tau\}_{\tau=1}^{3}$. At this step, $\mathcal{B}$ aborts if these extracted elements $(h^s, h^t, h^{sk_{\mathcal{U}}})$ contradict the security of the P-signature as in Game 2 of the proof of theorem 4. Under the HSDH and TDH assumptions, Game 3 and Game 2 are thus indistinguishable. These assumptions imply that, in particular, the adversary $\mathcal{A}$ cannot produce a coin for which $K_{\mathcal{U}}$ does not contain $h^{sk_{\mathcal{U}}}$ for a value of $sk_{\mathcal{U}}$ that was not certified using the P-signature at some $\mathcal{Q}_{\mathsf{withdraw}}$ query. Moreover, unless the failure event F_2 occurs, this underlying $sk_{\mathcal{U}}$ must always correspond to the public key $pk_{\mathcal{U}} = e(g,h)^{sk_{\mathcal{U}}}$ that $\mathcal{B}$ stored in $T_{\mathcal{W}}$ at the end of that particular $\mathcal{Q}_{\mathsf{withdraw}}$ query.

Game 4: is as Game 3 but we add a new failure event. Recall that the game ends with the adversary outputting coins $coin_a$ and $coin_b$ of values $v_a = 2^{\ell_a}$, $v_b = 2^{\ell_b}$ for which $\mathcal{B}$ detects a double spending (note that these coins must involve the same sub-wallet index $i \in [1, L]$). The

challenger $\mathcal{B}$ parses these coins as

$$coin_\kappa = \Big(\{C_{S,i}^{(\kappa)}\}_{i=1}^3, \{C_{T,i}^{(\kappa)}\}_{i=1}^3, \{C_{\mathcal{U},i}^{(\kappa)}\}_{i=1}^3,\ C_{K_\mathcal{U}}^{(\kappa)},\ \ldots,$$
$$\{\ldots, C_{Y_j}^{(\kappa)},\ \ldots\}_{j=0}^{L-\ell_\kappa-1},\ Y_{L-\ell_\kappa}^{(\kappa)},\ Z_{L-\ell_\kappa}^{(\kappa)}\ \ldots,\ \mathtt{info}_\kappa\Big).$$

for each $\kappa \in \{a, b\}$. At this step, $\mathcal{B}$ uses the extraction trapdoor to extract $h^{sk_\mathcal{U}^{(\kappa)}}$, $h^{s^{(\kappa)}}$, $h^{t^{(\kappa)}}$ and $\{Y_j^{(\kappa)}\}_{j=0}^{L-\ell_\kappa-1}$ from their commitments $C_{K_\mathcal{U}}^{(\kappa)}$, $C_{S,1}^{(\kappa)}$, $C_{T,1}^{(\kappa)}$, $\{C_{Y_j}^{(\kappa)}\}_{j=0}^{L-\ell_\kappa-1}$ for each $\kappa \in \{a, b\}$. Since $\mathsf{SameCoin}(coin_a, coin_b, v_a, v_b) = 1$, due to the perfect soundness of the proof system, it holds that either: (a) $v_a = v_b$ and $Y_{L-\ell_a}^{(a)} = Y_{L-\ell_b}^{(b)}$; (b) $v_a > v_b$ and $Y_{L-\ell_a}^{(a)} = Y_{L-\ell_a}^{(b)}$. We force $\mathcal{B}$ to abort in the event F_4 that $h^{s^{(a)}} \neq h^{s^{(b)}}$. We claim that event F_4 occurs with negligible probability as long as the Dodis-Yampolskiy PRF is pseudorandom in $\mathbb{G}_1$. Indeed, recall that the seeds $s^{(a)}, s^{(b)}$ are uniformly distributed in $\mathbb{Z}_p$ as they are generated by combining the contributions of $\mathcal{A}$ and $\mathcal{B}$. When two PRFs are seeded by distinct and uniformly random keys $s^{(a)}$, $s^{(b)}$ and when they have domains and ranges of polynomial size, the two ranges can only intersect with negligible probability. Otherwise, it is possible to construct a reduction that breaks their pseudorandomness without knowing the seeds as argued in [5][Appendix F]. To this end, the reduction would have to appeal to the simulator of the proof of anonymity in order to simulate proofs of the correctness of PRFs. Under the SXDH, D3DH and 2^{L+1}-DDHI assumptions, Game 4 is thus indistinguishable from Game 3.

Game 5: is as Game 4 but $\mathcal{B}$ also aborts if $h^{s^{(a)}} = h^{s^{(b)}}$ but $h^{sk_\mathcal{U}^{(a)}} \neq h^{sk_\mathcal{U}^{(b)}}$ or $h^{t^{(a)}} \neq h^{t^{(b)}}$. This event is easily seen to occur with negligible probability. Since the seeds $t^{(a)}$ and $t^{(b)}$ are uniformly distributed in $\mathbb{Z}_p$, two distinct wallet can only end up with the same $s^{(a)} = s^{(b)}$ with negligible chance. Game 5 is thus statistically close to Game 4.

Game 6: is as Game 5 but we add a last failure event and let $\mathcal{B}$ abort in the event that the two coins $coin_a$ and $coin_b$ are such that $(\mathtt{info}_a, pk_{\mathcal{M}_a}) \neq (\mathtt{info}_b, pk_{\mathcal{M}_b})$ but result in the same hash value $R_a = H(\mathtt{info}_a, pk_{\mathcal{M}_a}) = H(\mathtt{info}_b, pk_{\mathcal{M}_b}) = R_b$. As long as the hash function is collision-resistant, Game 6 is indistinguishable from Game 5.

In Game 6, $\mathcal{A}$ has no way to covertly double-spend coins since we ruled out all the possibilities allowing him to cheat without revealing his identity.

It comes that double-spenders are always identified provided the HSDH, TDH, D3DH and 2^{L+1}-DDHI assumptions hold and the hash function H is collision-resistant. □

A.4 Weak Exculpability

Theorem 6 (Weak Exculpability). *The scheme provides Weak Exculpability under the SXDH assumption, the D3DH assumption and the 2^L-DDHI assumption.*

Proof. The proof proceeds similarly to the proof of weak exculpability in [5] but relies on the Computational Bilinear Diffie-Hellman assumption according to which $e(g, h)^{abc}$ is infeasible to compute given $(g, g^a, g^b, g^c, h, h^a, h^b, h^c)$. We note that this assumption is implied by the D3DH assumption.

The proof proceeds using the sequence of games hereafter. At each step of this sequence, the adversary generates the public key $pk_\mathcal{B}$ on behalf of the dishonest bank.

Game 1: is the real attack game.

Game 2: we modify the treatment of $\mathcal{Q}_{\mathsf{GetKey}}$ queries. At the outset of the game, the challenger $\mathcal{D}$ defines values $a, b \xleftarrow{R} \mathbb{Z}_p$. At the k-th $\mathcal{Q}_{\mathsf{GetKey}}$ query, $\mathcal{D}$ replies by choosing $\gamma_k \xleftarrow{R} \mathbb{Z}_p$ and outputs $pk_{\mathcal{U}_k} = e(g^a, h^b)^{\gamma_k}$, which implicitly defines $sk_{\mathcal{U}_k} = ab\gamma_k$. Note that $pk_{\mathcal{U}_k}$ is uniformly distributed in the space of public keys and Game 2 is thus perfectly indistinguishable from Game 1.

Game 3: is as Game 2 but, instead of running the real setup algorithm of the e-cash system, the challenger $\mathcal{D}$ runs the simulated setup algorithm $\mathsf{SimCashSetup}$ described in the proof of theorem 3 and retains the simulation trapdoor τ. Under the SXDH assumption, Game 3 cannot be distinguished from Game 2.

Game 4: is as Game 3 but, at each $\mathcal{Q}_{\mathsf{withdraw}}$ query, $\mathcal{D}$ uses the ZK simulator to generate the interactive proof that $C_{sk_{\mathcal{U}}}$ is a commitment to the private key corresponding to $pk_{\mathcal{U}_k}$. Thanks to the zero-knowledge property of the interactive proof system, Game 4 is indistinguishable from Game 3.

Game 5: proceeds as Game 4 but, at steps 1 of each $\mathcal{Q}_{\mathsf{withdraw}}$ query, the challenger $\mathcal{D}$ generates the commitment $C_{sk_{\mathcal{U}}}$ as a commitment to a random value instead of a commitment to the actual private key $sk_{\mathcal{U}_k}$. Since the commitment is perfectly hiding, the adversary $\mathcal{A}$ does not see the difference between Game 4 and Game 5.

Game 6: is identical to Game 5 but, at each $\mathcal{Q}_{\mathsf{Spend}}$ query, the challenger $\mathcal{D}$ uses the simulator $\mathsf{SimSpend}$ described in the proof of theorem 3. The latter implies that, under the SXDH, q-DDHI and D3DH assumptions, the adversary $\mathcal{A}$ cannot notice the difference between Game 6 and Game 5. Moreover, in Game 6, the challenger $\mathcal{D}$ does not make use of the private key $sk_{\mathcal{U}_k} = ab\gamma_k$ of user k at any time.

Game 7: is like Game 6 with the difference that $\mathcal{D}$ aborts if the adversary manages to output coins $coin_a$ and $coin_b$ for which the identification algorithm points to some honest user $\mathcal{U}_k$. We claim that, in this situation, the collusion between the dishonest users and the bank $\mathcal{A}$ provides the challenger $\mathcal{D}$ with a way to solve the Bilinear Diffie-Hellman problem since the two coins $coin_a$ and $coin_b$ do not only reveal $pk_{\mathcal{U}_k}$, but also either:

- $g^{sk_{\mathcal{U}_k}} = g^{\gamma_k ab}$ if the double-spending occurs for the same node (and coins of the same value) of the sub-wallet i. This obviously reveals $e(g,h)^{abc} = e(g^{sk_{\mathcal{U}_k}}, h^c)^{1/\gamma_k}$ to $\mathcal{D}$.
- $e(h_0, g^{ab\gamma_k}) = T_{j_b,6}/e(Y_{i-\ell_a}, T_{j_b,5})$ – which equals $e(g,g)^{abc\gamma_k}$ if g_0 is defined to be $g_0 = g^c$ – if one of the two nodes is a descendant of the other one (we assumed w.l.o.g. that $v_b > v_a$). In this case, $\mathcal{D}$ also obtains $e(g,h)^{abc}$.

Under the aforementioned assumptions, the scheme thus provides weak exculpability. □