This is the Full Version of the Extended Abstract that appears
in the Proceedings of the 13th International Conference on Applied Cryptography and Network Security (ACNS '15)
(2–5 June 2015, New York, USA)
Tal Malkin Ed., Springer-Verlag, LNCS ????, pages ???–???.

Scalable Divisible E-Cash

Sébastien Canard[1], David Pointcheval[2], Olivier Sanders[1,2] and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] CNRS, ENS, INRIA, and PSL, Paris, France

Abstract. Divisible E-cash has been introduced twenty years ago but no construction is both fully secure in the standard model and efficiently scalable. In this paper, we fill this gap by providing an anonymous divisible E-cash construction with constant-time withdrawal and spending protocols. Moreover, the deposit protocol is constant-time for the merchant, whatever the spent value is. It just has to compute and store 2^l serial numbers when a value 2^l is deposited, compared to 2^n serial numbers whatever the spent amount (where 2^n is the global value of the coin) in the recent state-of-the-art paper. This makes a very huge difference when coins are spent in several times.
Our approach follows the classical tree representation for the divisible coin. However we manage to build the values on the nodes in such a way that the elements necessary to recover the serial numbers are common to all the nodes of the same level: this leads to strong unlinkability and anonymity, the strongest security level for divisible E-cash.

1 Introduction

Compared to regular cash, electronic payment systems offer greater convenience for end-users, but usually at the cost of a loss in terms of privacy. Introduced in 1982 by Chaum [11], electronic cash (E-cash) is one solution to reconcile the benefits of both solutions. As with regular cash, users of such systems can withdraw coins from a bank and then spend them to different merchants, while remaining anonymous, with unlinkable transactions. There is however one major difference: if a banknote or a coin can hardly be duplicated, this is on the contrary very easy to copy the series of bits constituting an electronic coin, as for any electronic data. It is therefore necessary, when designing an E-cash system, to provide a way of detecting double-spendings (*i.e.* two spendings using the same coin) and then to allow identification of the underlying defrauder. The challenge is to ensure such features without weakening the anonymity, or the efficiency, of the resulting scheme.

1.1 Related Work

Designing an E-cash system which can handle any amount for a payment (as it is the case for regular cash) is not a trivial task and several kinds of solutions exist in the literature.

One of them is to make use of coins of the smallest possible denomination (*e.g.* one cent), but this raises the problem of storing and spending the thousands of coins which become necessary to handle any amount. In [5], the authors partially address this latter problem by providing a compact E-cash system where users can withdraw wallets of N coins at once and store them efficiently. Unfortunately, each coin must be spent one by one which is unsuitable for practical use.

Another solution is to manage several denominations but, in practice, a user can be unable to make a payment if his wallet does not contain the kind of denomination he needs, since giving change back is not easy. For example, a user may have a wallet which only contains coins of \$10 while having to pay \$8. Such solution does not permit the user to make such payment, while he has enough money! This can be solved by using transferable e-cash systems, which in particular permits money change by the merchant, but at the cost of a larger coin [12].

The last solution to our initial problem has been proposed by Okamoto and Ohta [19] under the name of *divisible E-cash*. Such a system enables users to withdraw a coin C of a large value V, and then to spend it in several transactions, but in such a way that the sum of the amount of these transactions v_i is at most the global amount: $V \geq \sum v_i$. Typically, the coin is of value $V = 2^n$, and one can spend it with transactions of values $v_i = 2^{\ell_i}$, with $\ell_i \in \{0, \ldots, n\}$. This is currently the most relevant solution to solve the above problem and we now focus on this type of E-cash.

Since their introduction, many divisible E-cash schemes have been proposed (*e.g.* [18, 17, 6–8]), most of them sharing the same following idea. Every coin of global value 2^n is associated with a

binary tree with 2^n leaves, each leaf being associated with a unique serial number. When a user spends a value of 2^ℓ, he reveals some information related to an unspent node s of depth $n-\ell$ (and so with 2^ℓ descendant leaves). This allows the bank to recover the 2^ℓ serial numbers associated to the transaction. Such serial numbers, that the bank cannot link to a withdraw, are very convenient to detect defrauders. Indeed, a double-spending implies two transactions involving two nodes with a common subtree and so with common descendant leaves. Therefore, there will be a collision in the list of serial numbers stored by the bank, meaning that there is a double-spending.

Again, the main difficulty is to reconcile this double-spending technique with users' anonymity. The first constructions [19, 18, 9] only offered a weak level of anonymity since several spendings involving the same divisible coin could be linked one to each other. In [17], the first unlinkable system was proposed but the transaction still revealed which part of the coin was spent. Moreover, a trusted authority was necessary to recover defrauders' identity.

The first *truly* anonymous construction was provided in [6] but is rather inefficient. Indeed, this scheme makes use of several groups of different orders, whose generation is very expensive. Moreover, the spending phase requires complex non-interactive zero-knowledge (NIZK) proofs, which make it impractical. An improvement was later proposed in [7], with a much more efficient spending, but the resulting tree construction still suffers from similar downsides. In [2], the authors chose a different approach to construct their binary tree, using cryptographic hash functions. Unfortunately, such functions are not compatible with efficient NIZK proofs so that the authors relied on cut-and-choose protocols to prove the validity of the trees (and so of the coins). The resulting scheme was therefore proved secure under an unconventional security model where the bank is only ensured that it will not loose money on average. The security of all these constructions necessitate the use of the random oracle model (ROM) and the constructions are most of the time incompatible with Groth-Sahai proof methodology [15], and so the ROM cannot be avoided.

A first attempt to construct a divisible E-cash system secure in the standard model is due to Izabachène and Libert [16], but the resulting scheme is impractical, because of no efficient double-spending detection. Indeed, each time a coin is deposited to the bank, the latter has to compare it (by performing several costly computations) with all already deposited coins. This is due to the lack of serial numbers which was identified by the authors as the main cause of the inefficiency of their scheme.

Recently, the first practical E-cash system secure in the standard model was proposed in [8], with constant-time withdrawal and spending protocols. Unlike the previous schemes, where a new tree was generated by the users each time they withdrew coins, this new construction considers only one tree provided in the public parameters. This significantly alleviates the withdrawal and spending protocols since proving the validity of the tree is no longer necessary. However the scheme has two drawbacks, as identified by its authors: First, the public parameters must contain many elements allowing to recover the serial numbers (for double-spending detection), they are thus large; Second, while the deposit protocol is constant-time for the merchant, even for a one-cent deposit, the bank must perform 2^n pairing computations and store the results in a database. This obviously affects the scalability of the proposed scheme.

1.2 Our Contribution

In this paper, we improve the latter solution by fixing these two drawbacks. Although our scheme shares similarities with the one of [8], it differs on the binary tree generation. Indeed, in [8], the elements g_s associated with each node s were randomly and independently generated. This implies that the elements $\widetilde{g}_{s\mapsto f}$, provided to the bank to recover the serial numbers of leaf f from node s, differ according to each node s, leading to the above two issues.

Indeed, an anonymous scheme must reveal no information on the coin used in the transaction. So the bank does not know the involved node s but only its level $|s|$ (since it corresponds to the amount of the transaction). It follows that the bank does not know which elements $\widetilde{g}_{s\mapsto f}$ it should use to compute the underlying serial numbers. It has no other choice than performing the computations with all possible nodes s' of level $|s|$. This ensures that the valid serial numbers will be recovered but at the

cost of many useless computations. This also increases the risk of false double-spending detections, since additional (fake) serial numbers will be stored.

To prevent this problem, we design our tree differently: The nodes are now related in such a way that the elements needed to recover the serial numbers are common to every node at the same level. This reduces the size of the public parameters while avoiding useless computations. Indeed, with our solution, the bank only computes and stores 2^ℓ serial numbers when a value 2^ℓ is deposited, compared to 2^n in [8], whatever the value of the transaction (even for 1 cent).

However, these relations between nodes could also be used to break the strong unlinkability expected from an anonymous divisible E-cash system. To address this problem, we first require that the users encrypt some of the elements they send to the merchant. Unfortunately, the randomness used during the encryption is a problem to recover the deterministic serial numbers. We therefore add some elements in the public parameters which will allow the bank to efficiently cancel the randomness, without endangering the security of our scheme.

These modifications will slightly increase the complexity of the spending protocol but will lead to a much more efficient deposit one. Our solution can then be seen as a way to make the practical divisible E-cash system from [8] highly scalable.

1.3 Organization

In Section 2, we recall some definitions and present the computational assumption our protocol will rely on. Section 3 reviews the syntax of a divisible E-cash system along with informal definitions of the security properties. Section 4 provides a high level description of our construction, while Section 5 goes into the details. An improved fair variant is proposed in Section 6. Because of lack of space, the security analysis is postponed to Appendix A.

2 Preliminaries

2.1 Bilinear Groups

Bilinear groups are a set of three cyclic groups, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, of prime order p, along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. for all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{a \cdot b}$;
2. for any $g \neq 1_{\mathbb{G}_1}$ and $\widetilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$;
3. the map e is efficiently computable.

Galbraith, Paterson, and Smart [13] defined three types of pairings: in type 1, $\mathbb{G}_1 = \mathbb{G}_2$; in type 2, $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists an efficient homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, while no efficient one exists in the other direction; in type 3, $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism exist between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction.

Our construction, as well as the one of [8], requires the use of asymmetric pairings (*i.e.* of type 2 or type 3). For simplicity, we will only consider pairings of type 3 in this work, which is not a strong restriction (see [10]) since these pairings offer the best efficiency.

2.2 Computational Assumption

Besides the classical SXDH and $q-$SDH [4] assumptions in bilinear groups, our construction relies on a new computational assumption, we call EMXDH, since this is an extension of the multi-cross-Diffie-Hellman assumption.

Definition 1 (SXDH assumption). *For $k \in \{1, 2\}$, the DDH assumption is hard in $\mathbb{G}_k$ if, given $(g, g^x, g^y, g^z) \in \mathbb{G}_k^4$, it is hard to distinguish whether $z = x \cdot y$ or z is random. The SXDH assumption holds if DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$*

Definition 2 ($q-$SDH assumption). *Given $(g, g^x, g^{x^2}, ..., g^{x^q}) \in \mathbb{G}_1$, it is hard to output a pair $(m, g^{\frac{1}{x+m}})$.*

Definition 3 (EMXDH Assumption). *Given* $(g, g^a, g^x, g^t, \widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$, $(\{g^{y^i}\}_{i=1}^{i=n}, \{g^{t \cdot y^i}\}_{i=1}^{i=n}, \{g^{x \cdot y^i}\}_{i=1}^{i=n-1}, \{g^{x \cdot t \cdot y^i}\}_{i=1}^{i=n-1}, \{\widetilde{g}^{1/y^i}\}_{i=1}^{i=n}) \in \mathbb{G}_1^{4n-2} \times \mathbb{G}_2^n$, *as well as* $(g^{z_1}, g^{z_2}) \in \mathbb{G}_1$, *it is hard to distinguish whether* $(z_1, z_2) = (x \cdot y^n/a, x \cdot t \cdot y^n/a)$ *or* (z_1, z_2) *is random.*

We discuss the hardness of the problem related to this assumption in Appendix B. We stress however that this assumption, as the SXDH one, would clearly be wrong with a symmetric pairing since the test $e(g^{z_1}, g^a) = e(g^x, g^{y^n})$ would allow distinguishing a random z_1 from a valid one.

This assumption will underlie the anonymity of our construction. However, as explained in Section 6, one can rely on a weaker assumption if a weaker level of anonymity is enough.

2.3 Digital Signature Scheme

A digital signature scheme Σ is defined by three algorithms:

- the key generation algorithm $\Sigma.\mathtt{Keygen}$ which outputs a pair of signing and verification keys $(\mathsf{sk}, \mathsf{pk})$ – we assume that sk always contains pk;
- the signing algorithm $\Sigma.\mathtt{Sign}$ which, on input the signing key sk and a message m, outputs a signature σ;
- and the verification algorithm $\Sigma.\mathtt{Verify}$ which, on input m, σ and pk, outputs 1 if σ is a valid signature on m under pk, and 0 otherwise.

The standard security notion for a signature scheme is *existential unforgeability under chosen message attacks* (EUF-CMA) [14] which means that it is hard, even given access to a signing oracle, to output a valid pair (m, σ) for a message m never asked to the oracle. In this paper we will also use two variants. The first one is the security against *selective chosen message attacks* (SCMA), which limits the oracle queries to be asked before having seen the key pk. The second one is a *strong unforgeability* (SUF) where the adversary must now output a valid pair (m, σ) which was not returned by the signing oracle (a new signature for an already signed message is a valid forgery) but can only ask one query to the signing oracle (OTS, for One-Time Signature).

2.4 Groth-Sahai Proof Systems

In [15], Groth and Sahai propose a non-interactive proofs system, in the common reference string (CRS) model, which captures most of the relations for bilinear groups. There are two types of CRS that yields either perfect soundness or perfect witness indistinguishability. These two types of CRS are computationally indistinguishable (under the SXDH assumption in our setting).

To prove that some variables satisfy a set of relations, the prover first commits to them (by using the elements from the CRS) and then computes one proof element per relation. Efficient non-interactive witness undistinguishable proofs are available for pairing-product equations or multi-exponentiation equations. The former are of the type:

$$\prod_{i=1}^{n} e(A_i, \widetilde{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, \widetilde{X}_j)^{a_{i,j}} = t_T$$

for variables $\{X_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{X}_i\}_{i=1}^n \in \mathbb{G}_2$ and constant $t_T \in \mathbb{G}_T$, $\{A_i\}_{i=1}^n \in \mathbb{G}_1$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}_p$.

The latter are of the type:

$$\prod_{i=1}^{n} A_i^{y_i} \prod_{j=1}^{n} X_j^{b_j} \prod_{i=1}^{n} \prod_{j=1}^{n} X_j^{y_i \cdot a_{i,j}} = T$$

for variables $\{X_i\}_{i=1}^n \in \mathbb{G}_k$, $\{y_i\}_{i=1}^n \in \mathbb{Z}_p$ and constant $T \in \mathbb{G}_k$, $\{A_i\}_{i=1}^n \in \mathbb{G}_k$, $\{b_i\}_{i=1}^n \in \mathbb{Z}_p$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}_p$ for $k \in \{1, 2\}$.

Multi-exponentiation equations also admit non-interactive zero-knowledge (NIZK) proofs at no additional cost.

3 Divisible E-cash System

For consistency, we recall the syntax of a divisible E-cash system described in [8]. For simplicity, we borrow their notations.

Syntax. A divisible e-cash system is defined by the following algorithms, that involve at least three entities: the bank $\mathcal{B}$, a user $\mathcal{U}$ and a merchant $\mathcal{M}$.

- $\texttt{Setup}(1^k, V)$: On inputs a security parameter k and an integer V, this probabilistic algorithm outputs the public parameters $p.p.$ for divisible coins of global value V. We assume that $p.p.$ are implicit to the other algorithms, and that they include k and V. They are also an implicit input to the adversary, we will then omit them.
- $\texttt{BKeygen}()$: This probabilistic algorithm executed by the bank $\mathcal{B}$ outputs a key pair $(\mathsf{bsk}, \mathsf{bpk})$. It also sets L as an empty list, that will store all deposited coins. We assume that bsk contains bpk.
- $\texttt{Keygen}()$: This probabilistic algorithm executed by a user $\mathcal{U}$ (resp. a merchant $\mathcal{M}$) outputs a key pair $(\mathsf{usk}, \mathsf{upk})$ (resp. $(\mathsf{msk}, \mathsf{mpk})$). We assume that usk (resp. msk) contains upk (resp. mpk).
- $\texttt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: This is an interactive protocol between the bank $\mathcal{B}$ and a user $\mathcal{U}$. At the end of this protocol, the user gets a divisible coin C of value V or outputs $\perp$ (in case of failure) while the bank stores the transcript $\texttt{Tr}$ of the protocol execution or outputs $\perp$.
- $\texttt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, v), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, v))$: This is an interactive protocol between a user $\mathcal{U}$ and a merchant $\mathcal{M}$. At the end of the protocol the merchant gets a master serial number Z of value v (the amount of the transaction they previously agreed on) along with a proof of validity Π or outputs $\perp$. $\mathcal{U}$ either updates C or outputs $\perp$.
- $\texttt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (v, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: This is an interactive protocol between a merchant $\mathcal{M}$ and the bank $\mathcal{B}$. $\mathcal{B}$ checks that Π is valid on v and Z and that (v, z, Π) has never been deposited (corresponding to the case of a cheating merchant). $\mathcal{B}$ then recovers the m (for some $m \geq v$) serial numbers $z_1, \ldots, z_m$ corresponding to this transaction and checks whether, for some $1 \leq i \leq m$, $z_i \in L$. If none of the serial numbers is in L, then the bank credits $\mathcal{M}$'s account of v, stores (v, Z, Π) and appends $\{z_1, \ldots, z_m\}$ to L. Else, there is at least an index $i \in \{1, \ldots, m\}$ and a serial number z' in L such that $z' = z_i$. The bank then recovers the tuple (v', Z', Π') corresponding to z' and publishes $[(v, Z, \Pi), (v', Z', \Pi')]$.
- $\texttt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2), \mathsf{bpk})$: On inputs two different valid transcripts (v_1, Z_1, Π_1) and (v_2, Z_2, Π_2), this deterministic algorithm outputs a user's public key upk if there is a collision between the serial numbers derived from Z_1 and from Z_2, and $\perp$ otherwise.

Security Model. Besides correctness, the authors of [8] formally defined 3 security properties that a secure divisible e-cash system must achieve. The first one is *traceability* which requires that no coalition of users can spend more than they have withdrawn without revealing one of their identities. The second one is *exculpability* which requires that no user can be falsely accused of double-spending, even by a coalition of the bank, users and merchants. Eventually, the last property expected by such schemes is *anonymity* which means that no one can learn anything about a spending except the information already available from the environment (such as the date, the value of the spending,...).

However, they also describe two variants of anonymity that they called *unlinkability* and *strong unlinkability*. The former requires that two spendings from the same coin cannot be linked except by revealing which part of the coin is spent. The latter strengthens the level of anonymity by forbidding this additional leakage of information. A divisible, strongly unlinkable, e-cash system can be made anonymous by providing a way to identify double-spenders using only public information and so without the help of a trusted entity.

As explained in [8], a divisible e-cash system which is just unlinkable cannot achieve the anonymity property. In this paper, we improve on [8] by reducing the storage and computation of the anonymous version (strongly unlinkable).

The formal security games for traceability, exculpability, anonymity are recalled in Appendix C.

4 Our Construction

Notation. Let $\mathcal{S}_n$ be the set of bitstrings of size smaller than n and $\mathcal{F}_n$ be the set of bitstrings of size exactly n. For every $s \in \mathcal{S}_n$, $|s|$ denotes the length of s, and we define the set $\mathcal{F}_n(s)$ as $\{f \in \mathcal{F}_n : s \text{ is a prefix of } f\}$. For any $i \in \{0, \ldots, n\}$, we set $\mathcal{L}(i)$ as $\{b_{i+1}...b_n : b_j \in \{0,1\}\}$, *i.e.* the set of bitstrings of size $n-i$, indexed by $i+1, \ldots, n$. Therefore, $\mathcal{L}(n)$ only contains the empty string, while $\mathcal{L}(0) = \mathcal{F}_n$.

In the following, each node s of a tree of depth n (defining a coin of value 2^n) will refer to an element of $\mathcal{S}_n$. The root will then be associated with the empty string ϵ and a leaf with an element of $\mathcal{F}_n$. For all $s \in \mathcal{S}_n \setminus F_n$, the left child (resp. the right child) of s will refer to $s||0$ (resp. $s||1$).

4.1 High Level Description

The initial construction [8] works in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where g (resp. $\widetilde{g}$) is a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) and $G = e(g, \widetilde{g})$. The core idea of their construction is to define one single tree in the public parameters which is common to all the coins. Each node s (resp. leaf f) of this tree is associated with an element $g_s \leftarrow g^{r_s} \in \mathbb{G}_1$ (resp. $\chi_f \leftarrow G^{y_f}$) for some random scalar r_s (resp. y_f). Each (divisible) coin is associated to a secret scalar x which implicitly defines its serial numbers as $\{G^{x \cdot y_f}\}_{f \in \mathcal{F}_n}$. To allow the bank to detect double-spendings, the public parameters contain, for each $s \in \mathcal{S}_n$ and each $f \in \mathcal{F}_n(s)$, the element $\widetilde{g}_{s \mapsto f} \leftarrow \widetilde{g}^{y_f / r_s} \in \mathbb{G}_2$. Indeed, by using them and the element $t_s = g_s^x$ provided by the user during the spending, the bank is able to recover the serial numbers $\{G^{x \cdot y_f}\}_{f \in \mathcal{F}_n(s)}$ since $e(t_s, \widetilde{g}_{s \mapsto f}) = G^{x \cdot y_f}$.

Limitations. However, this solution has two drawbacks. First, it implies public parameters of significant size since they must contain $(n+1) \cdot 2^n$ elements $\widetilde{g}_{s \mapsto f}$. Second, each of these elements depends on a node s, so that the bank needs to know the spent node s^* to select the correct $\widetilde{g}_{s^* \mapsto f}$ and compute the associated serial numbers. Unfortunately, to achieve the strong unlinkability or the anonymity properties, a divisible E-cash must not reveal this node s^*. Therefore, the only way for the bank to detect double-spendings is to compute, for every node s of the same level than s^* and for every $f \in \mathcal{F}_n(s)$, the pairings $e(t_{s^*}, \widetilde{g}_{s \mapsto f})$. For a deposit of one cent, the bank must then perform 2^n pairings to get the 2^n potential serial numbers, only one of them being valid. This additionally increases the risk of false positive.

Our Approach. In this work, we construct our parameters in such a way that the elements used to compute the serial numbers do no longer depend on the specific nodes, but only on the levels of the nodes in the tree (and so only on the spent values, which are publicly known). More precisely, for each level i, we provide 2^{n-i} pairs of elements of $\mathbb{G}_2$ which will be used by the bank each time a node of this level is deposited. Therefore, the bank will no longer need to perform useless computations and so will only have to compute V serial numbers when a value V will be deposited. Moreover, it decreases the size of the public parameters since only $2^{n+2} - 2$ elements (instead of $(n+1) \cdot 2^n$) of $\mathbb{G}_2$ are necessary.

Description. Informally, we associate the root ϵ of our tree with an element $g_\epsilon \in \mathbb{G}_1$, and each level i, from 1 to n, with two random scalars $y_{i,0}, y_{i,1} \xleftarrow{\$} \mathbb{Z}_p$. Given a node s associated with an element $g_s \in \mathbb{G}_1$ we can compute the element $g_{s||0} \leftarrow g_s^{y_{|s|+1,0}}$ associated with its left child and the element $g_{s||1} \leftarrow g_s^{y_{|s|+1,1}}$ associated with its right child. Therefore, as illustrated on Figure 1, each node $s = b_1 \ldots b_{|s|}$ is associated with an element $g_s \leftarrow g^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}}$.

To allow the bank to compute the serial numbers associated to this node, we provide, for all $i = 0, \ldots, n$, and for each $f = b_{i+1} \ldots b_n \in \mathcal{L}(i)$, the value $\widetilde{g}_{i,f} \leftarrow \widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}$. The point here is that $\widetilde{g}_{i,f}$ is common to every node of level i and so will be used by the bank each time a deposit of value 2^{n-i} is made. As illustrated on Figure 2 (which shows the generic tree without the secret value x), the serial numbers of a coin associated with the secret x are then implicitly defined as $\{G^{x \cdot y_\epsilon \prod_{i=1}^{n} y_{i,b_i}}\}_{b_1 \ldots b_n \in \mathcal{F}_n}$.

Unfortunately, we cannot provide $t_s = g_s^x$ during a spending as in [8]. Revealing this element indeed breaks the anonymity of our new scheme. For example, if s is a node of level $n-1$, then a

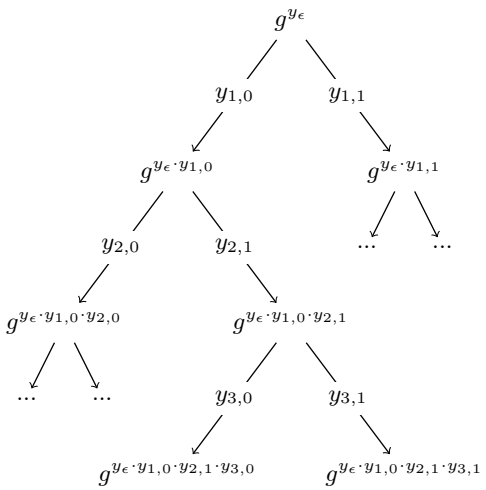

Fig. 1. Divisible coin

spending involving its left child $s||0$ and a spending involving its right child $s||1$ should be unlinkable. However, this is not true when we reveal $g^x_{s||0}$ and $g^x_{s||1}$, since one can simply check whether the equality $e(g^x_{s||0}, \widetilde{g}_{n-1,1}) = e(g^x_{s||1}, \widetilde{g}_{n-1,0})$ holds. Indeed:

$$e(g^x_{s||0}, \widetilde{g}_{n-1,1}) = e(g^x_s, \widetilde{g})^{y_{n,0} \cdot y_{n,1}} = e(g^x_{s||1}, \widetilde{g}_{n-1,0}).$$

To overcome this problem, at the spending time, the user will just send an ElGamal encryption of $t_s = g^x_s$ under the public key $k_{|s|}$, *i.e.* a pair $(g^{r_1}, t_s \cdot k^{r_1}_{|s|})$ for some random $r_1 \xleftarrow{\$} \mathbb{Z}_p$. This will slightly increase the number of elements and the complexity of the proof that the user must produce during a spending, but it will ensure the strong unlinkability of our scheme. For the same reasons, we cannot reveal the security tag $\mathsf{upk}^R \cdot h^x_s$ (where R is obtained by hashing some public information related to the transaction), which is used in [8] to identify a double-spender. We will just provide an ElGamal encryption of it under the same public key.

Despite the randomness used in the ciphertexts, the bank must remain able to compute the deterministic serial numbers. We will then provide some additional elements $\widetilde{h}_{i,f}$ (see Section 4.2) in the public parameters to cancel this randomness without endangering the anonymity of our scheme.

Security Analysis. All the differences between our solution and the one of [8] lead to a new proof of anonymity. Indeed, in the latter, the elements g_s are chosen randomly which enables the reduction $\mathcal{R}$ to embed a DDH challenge in one node s^* without affecting the other ones, but with a few additional inputs. Therefore, $\mathcal{R}$ can handle any query involving the other nodes as long as its guess on s^* is correct. This is no longer the case with our solution since the elements g_s are now related, hence the stronger EMXDH assumption described in Section 2.2.

4.2 Setup

Public Parameters. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be the description of bilinear groups of prime order p, elements g, h, u_1, u_2, w be generators of $\mathbb{G}_1$, and $\widetilde{g}$ be a generator of $\mathbb{G}_2$. We denote $G = e(g, \widetilde{g})$. A trusted authority generates $(y_\epsilon, a_0) \xleftarrow{\$} \mathbb{Z}_p^2$ and, for $i = 1, \ldots, n$, $(y_{i,0}, y_{i,1}, a_i) \xleftarrow{\$} \mathbb{Z}_p^3$.

The trusted authority computes $(g_\epsilon, h_\epsilon) \leftarrow (g^{y_\epsilon}, h^{y_\epsilon})$ and for any node $s = b_1 \ldots b_{|s|}$, $(g_s, h_s) \leftarrow (g^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}}, h^{y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i}})$. Eventually, it computes, for $i = 0, \ldots, n$:

- $k_i \leftarrow g^{a_i}$, the ElGamal encryption key;
- $(\widetilde{g}_{i,f}, \widetilde{h}_{i,f}) \leftarrow (\widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}, \widetilde{g}^{-a_i \prod_{j=i+1}^{n} y_{j,b_j}})$, for every $f = b_{i+1} \ldots b_n \in \mathcal{L}(i)$.

As said in [8], the bank and a set of users can cooperatively generate such parameters, avoiding the need of such trusted entity. The public parameters $p.p.$ are set as the bilinear groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with

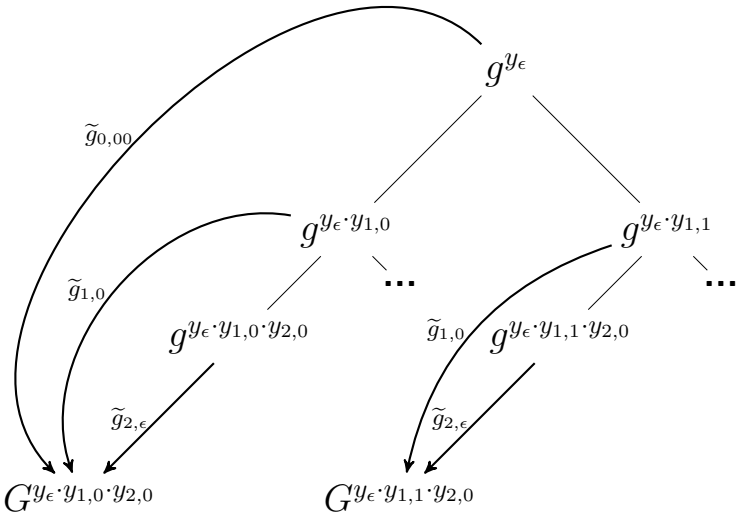

Fig. 2. Computing serial numbers

the generators g, h, u_1, u_2, w and $\widetilde{g}$, a collision-resistant full-domain hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, as well as all the above elements $\{(g_s, h_s), s \in \mathcal{S}_n\}$, $\{k_i, i = 0, \ldots, n\}$ and $\{(\widetilde{g}_{i,f}, \widetilde{h}_{i,f}), i = 0, \ldots, n, f \in \mathcal{L}(i)\}$.

One can remark that the elements $(\widetilde{g}_{i,f}, \widetilde{h}_{i,f})$ will be used to cancel the ElGamal encryption at level i, for any leaf f:

$$e(k_i, \widetilde{g}_{i,f}) = e(g^{a_i}, \widetilde{g}^{\prod_{j=i+1}^{n} y_{j,b_j}}) = (g, \widetilde{g}^{a_i \prod_{j=i+1}^{n} y_{j,b_j}}) = e(g, \widetilde{h}_{i,f})^{-1}.$$

As a consequence, it is important to note that ElGamal encryptions under k_i are not semantically secure because of these elements, but the one-wayness will be enough for our purpose.

Although our construction is compatible with both the random oracle and the standard models, we will only describe, for the sake of clarity, a protocol with provable security in the standard model. We must therefore add to the public parameters the description of a common reference string (CRS) for the perfect soundness setting of the Groth-Sahai [15] proofs system and a one-time signature scheme Σ_{ots} (*e.g.* the one of [4]).

5 Our Divisible E-Cash System

In this section, we provide an extended description of our new protocol. Then, we discuss its efficiency.

5.1 The protocol

– `BKeygen`(): The bank has to sign two different kinds of messages and so selects two signature schemes denoted Σ_0 and Σ_1.
  - The former will be used to compute signatures τ_s on pairs (g_s, h_s) for every node s of the tree. Such signatures will allow users to prove during a spending that they use a valid pair (g_s, h_s) without revealing it.
  - The latter will be used by the bank during the `Withdraw` protocol to certify the secret values associated with the withdrawn coin.

  Both schemes has to allow signatures on elements of $\mathbb{G}_1^2$ while being compatible with Groth-Sahai [15] proofs. We will therefore instantiate them with the structure preserving signature scheme proposed in [1], since it was proven to be optimal for type 3 pairings.

The bank generates the pair $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \Sigma_1.\mathtt{Keygen}(p.p.)$ and the pairs $(\mathsf{sk}_0^{(i)}, \mathsf{pk}_0^{(i)}) \leftarrow \Sigma_0.\mathtt{Keygen}(p.p.)$, for each level $i = 0, \ldots, n$ of the tree, and computes, for every node $s \in \mathcal{S}_n$, $\tau_s \leftarrow \Sigma_0.\mathtt{Sign}(\mathsf{sk}_0^{(|s|)}, (g_s, h_s))$. Eventually, it will set bsk as sk_1 and bpk as $(\{\mathsf{pk}_0^{(i)}\}_i, \mathsf{pk}_1, \{\tau_s\}_{s\in\mathcal{S}_n})$. A way to reduce the size of this public key is described in Remark 5.

- $\mathtt{Keygen}()$: Each user (resp. merchant) selects a random $\mathsf{usk} \leftarrow \mathbb{Z}_p$ (resp. msk) and gets $\mathsf{upk} \leftarrow g^{\mathsf{usk}}$ (resp. $\mathsf{mpk} \leftarrow g^{\mathsf{msk}}$). In the following we assume that upk (resp. mpk) is public, meaning that anyone can get an authentic copy of it.
- $\mathtt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: To withdraw a divisible coin, the user must jointly compute with the bank a random scalar x (thus without control on it), and then get a certificate. In practice, the scalar x is computed as $x = x_1 + x_2$ where x_1 is chosen and kept secret by the user and x_2 is chosen by the bank and given to the user. It is also necessary to bind this secret value to the user's identity to allow identification of double-spenders.
  The user then computes u_1^{usk} and $u_2^{x_1}$, sends them along with upk to the bank, and proves knowledge of x_1 and usk (in a zero-knowledge way, such as the Schnorr's interactive protocol [20]). If the proof is valid, the bank chooses a random x_2, and checks that $u = u_2^{x_1} \cdot u_2^{x_2}$ was not previously used, the bank computes $\sigma \leftarrow \Sigma_1.\mathtt{Sign}(\mathsf{sk}_1, (u_1^{\mathsf{usk}}, u))$ and sends it to the user, together with x_2. The user computes $x = x_1 + x_2$, and sets $C \leftarrow (x, \sigma)$. σ is thus a signature on the pair $(u_1^{\mathsf{usk}}, u_2^x)$, which strongly binds the user to the randomly chosen x.
- $\mathtt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, 2^\ell), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, 2^\ell))$: To spend a value 2^ℓ, the user selects an unspent node s of level $n-\ell$ and two random scalars $r_1, r_2 \leftarrow \mathbb{Z}_p$ and computes $R \leftarrow H(info)$, $t_s \leftarrow (g^{r_1}, g_s^x \cdot k_{n-\ell}^{r_1})$ and $v_s \leftarrow (g^{r_2}, \mathsf{upk}^R \cdot h_s^x \cdot k_{n-\ell}^{r_2})$, where *info* is some information related to the transaction (date, amount, merchant's public key, ...). Actually, t_s and v_s are ElGamal encryptions of g_s^x, the identifier of the node, and of $\mathsf{upk}^R \cdot h_s^x$, the security tag used to identify double-spenders. But of course, he must additionally prove that the plaintexts in t_s and v_s are valid, *i.e.* they are related to the values certified during a withdrawal. To do so, he will provide Groth-Sahai proof of knowledge of σ, and proof of existence of τ_s to attest the validity of the pair (g_s, h_s).
  However, Groth-Sahai [15] proofs can be re-randomized. This can be a problem in our case, since a dishonest merchant could re-randomize a valid transcript and to deposit it again. This would lead an honest user to be accused of double-spending, and so would break the exculpability property. To prevent this bad behavior, the user first generates a one-time signature key pair $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots}) \leftarrow \Sigma_{ots}.\mathtt{Keygen}(1^k)$ and certifies the public key into $\mu \leftarrow w^{\frac{1}{\mathsf{usk}+H(\mathsf{pk}_{ots})}}$. This key pair will then be used to sign the transcript (including the proofs).
  Next, the user computes Groth-Sahai commitments to $\mathsf{usk}, x, r_1, r_2, g_s, h_s, \tau_s, \sigma, \mu$, $U_1 = u_1^{\mathsf{usk}}$, and $U_2 = u_2^x$, and provides a NIZK proof π that the committed values satisfy:

$$t_s = (g^{r_1}, g_s^x \cdot k_{n-\ell}^{r_1}) \quad \wedge \quad v_s = (g^{r_2}, (g^R)^{\mathsf{usk}} \cdot h_s^x \cdot k_{n-\ell}^{r_2})$$

$$U_2 = u_2^x \quad \wedge \quad U_1 = u_1^{\mathsf{usk}} \quad \wedge \quad \mu^{(\mathsf{usk}+H(\mathsf{pk}_{ots}))} = w$$

  along with a NIWI proof π' that the committed values satisfy:

$$1 = \Sigma_0.\mathtt{Verify}(\mathsf{pk}_0^{(n-\ell)}, (g_s, h_s), \tau_s) \quad \wedge \quad 1 = \Sigma_1.\mathtt{Verify}(\mathsf{pk}_1, (U_1, U_2), \sigma).$$

  Finally, the user computes $\eta \leftarrow \Sigma_{ots}.\mathtt{Sign}(\mathsf{sk}_{ots}, H(R||t_s||v_s||\pi||\pi'))$ and sends it to $\mathcal{M}$ along with $\mathsf{pk}_{ots}, t_s, v_s, \pi, \pi'$.
  The merchant then checks the validity of the proofs and of the signatures, and accepts the transaction if everything is correct. In such a case, he stores $(2^\ell, Z, \Pi)$ where $Z \leftarrow (t_s, v_s)$ and $\Pi \leftarrow (\pi, \pi', \mathsf{pk}_{ots}, \eta)$.
- $\mathtt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (2^\ell, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: Upon receiving the tuple $(2^\ell, Z = (t_s, v_s), \Pi = (\pi, \pi', \mathsf{pk}_{ots}, \eta))$, the bank first checks the validity of the proofs and signatures. Then, it checks that it was not previously deposited. To this aim, it parses t_s as $(t_s[1], t_s[2])$, and for each $f \in \mathcal{L}(n-\ell)$, it computes the serial numbers $z_f \leftarrow e(t_s[2], \widetilde{g}_{n-\ell,f}) \cdot e(t_s[1], \widetilde{h}_{n-\ell,f})$ and checks whether $z_f \in L$. If none of them is in L, which means that none was already spent, the bank adds these elements to this list L and associates them with the transcript $(2^\ell, Z, \Pi)$. Else, there is an element $z' \in L$

such that for some f, $z_f = z'$. The bank recovers the corresponding transcript $(2^{\ell'}, Z', \Pi')$ and outputs $[(2^{\ell}, Z, \Pi), (2^{\ell'}, Z', \Pi')]$.
As remarked above, $e(k_i, \widetilde{g}_{i,f}) \cdot e(g, \widetilde{h}_{i,f}) = 1$, for any level i and any leaf f, and so $z_f = e(g_s^x, \widetilde{g}_{n-\ell,f})$, and is thus independent of r_1.
As noticed in [8], the bank does not actually have to store and compare the elements $z_i \in \mathbb{G}_T$ but only their fingerprints, that can be small hash values for some collision-resistant hash function.

– $\texttt{Identify}((2^{\ell_1}, Z_1, \Pi_1), (2^{\ell_2}, Z_2, \Pi_2), \mathsf{bpk})$: To recover the identity of a double-spender from such a pair of transcripts, one first checks the validity of both transcripts and returns $\perp$ if one of them is not correct. One then parses Z_i as (t_{s_i}, v_{s_i}) and computes the lists $S_i \leftarrow \{e(t_{s_i}[2], \widetilde{g}_{n-\ell_i,f}) \cdot e(t_{s_i}[1], \widetilde{h}_{n-\ell_i,f}), \forall f \in \mathcal{L}(n-\ell_i)\}$, for $i = 1, 2$. One returns $\perp$ if there is no collision between S_1 and S_2. In case of collision, there are $f_1 \in \mathcal{L}(n-\ell_1)$ and $f_2 \in \mathcal{L}(n-\ell_2)$ such that

$$\begin{aligned} e(g_{s_1}^{x_1}, \widetilde{g}_{n-\ell_1,f_1}) &= e(t_{s_1}[2], \widetilde{g}_{n-\ell_1,f_1}) \cdot e(t_{s_1}[1], \widetilde{h}_{n-\ell_1,f_1}) \\ &= e(t_{s_2}[2], \widetilde{g}_{n-\ell_2,f_2}) \cdot e(t_{s_2}[1], \widetilde{h}_{n-\ell_2,f_2}) = e(g_{s_2}^{x_2}, \widetilde{g}_{n-\ell_2,f_2}). \end{aligned}$$

But then, since the x's values are mutually chosen by the user and the bank, they are random, while all the other elements are fixed from the setup. This is thus quite unlikely this equality holds for different random x and different paths in the tree: such a collision is a double-spending of a leaf $f \in \mathcal{L}(0) = \mathcal{F}_n$ in the tree parametrized by $x = x_1 = x_2$.
In addition, because of the signature σ, and the soundness of the NIWI in the transcript, the same user is necessarily associated to the two coins: it is quite unlikely two users come up with the same x. Then, if one lets T_i be $e(v_{s_i}[2], \widetilde{g}_{n-\ell_i,f_i}) \cdot e(v_{s_i}[1], \widetilde{h}_{n-\ell_i,f_i})$, for $i = 1, 2$, since we also have $e(h_{s_1}^{x_1}, \widetilde{g}_{n-\ell_1,f_1}) = e(h_{s_2}^{x_2}, \widetilde{g}_{n-\ell_2,f_2})$, as above, we have the simplification $T_i = e(\mathsf{upk}_i^{R_i} \cdot h_{s_i}^{x_i}, \widetilde{g}_{n-\ell_i,f_i})$, and even:

$$T_1/T_2 = e(\mathsf{upk}^{R_1}, \widetilde{g}_{n-\ell_1,f_1})/e(\mathsf{upk}^{R_2}, \widetilde{g}_{n-\ell_2,f_2}) = e(\mathsf{upk}, \widetilde{g}_{n-\ell_1,f_1}^{R_1}/\widetilde{g}_{n-\ell_2,f_2}^{R_2}).$$

In order to trace the double-spender, one has to compute, for each public key upk_i, the value $e(\mathsf{upk}_i, \widetilde{g}_{n-\ell_1,f_1}^{R_1}/\widetilde{g}_{n-\ell_2,f_2}^{R_2})$ until one gets a match with T_1/T_2, in which case the algorithm outputs upk_i.

Remark 4. We stress that a collision on the serial numbers (or on their fingerprints) means that for the secret values x_1, x_2, and for the secret nodes s_1, s_2 at public levels ℓ_1, ℓ_2, there are $f_1 \in \mathcal{L}(n-\ell_1)$ and $f_2 \in \mathcal{L}(n-\ell_2)$ verifying:

$$x_1 \cdot y_\epsilon \prod_{i=1}^{n-\ell_1} y_{i,b_{1,i}} \prod_{j=n-\ell_1+1}^{n} y_{j,b_{1,j}} = x_2 \cdot y_\epsilon \prod_{i=1}^{n-\ell_2} y_{i,b_{2,i}} \prod_{j=n-\ell_2+1}^{n} y_{j,b_{2,j}}$$

Since the secret x_1 and x_2 are randomly chosen after the values $y_{i,b}$ have been randomly fixed, the collision is quite unlikely if $x_1 \neq x_2$. Similarly, because of the random choice of the values $y_{i,b}$, it is quite unlikely there exist two disctint sequences $(b_{1,i}) \neq (b_{2,i})$ such that $\prod_{i=1}^{n} y_{i,b_{1,i}} = \prod_{i=1}^{n} y_{i,b_{2,i}}$. As a consequence, the two sequences are equal, which means that $s_1||f_1 = s_2||f_2$: a collision corresponds to two spendings involving the same path $s_1||f_1$ in the tree parametrized by $x = x_1 = x_2$, with overwhelming probability.

On the other hand, one can easily check that a double-spending automatically leads to a collision.

Remark 5. The appropriate combination of the node s and the final path f into $s||f$ from the root to the leaf in all the serial number computations comes from the signature τ_s that involves the key at the appropriate public level (since this is related to the value of the coin). Our approach consists in generating $(n+1)$ public keys $\mathsf{pk}_0^{(i)}$ (one for each level i) and a signature τ_s for every node $s \in \mathcal{S}_n$, all in the public key bpk of the bank. Proving that (g_s, h_s) is valid for a spending of 2^ℓ can then be achieved by proving knowledge of a signature τ_s such that $\Sigma_0.\texttt{Verify}(\mathsf{pk}_0^{(n-\ell)}, (g_s, h_s), \tau_s) = 1$.

Unfortunately, this solution implies a bank public key of significant size since it must contain $(n+1)$ public keys (one for each level) and $2^{n+1} - 1$ structure preserving signatures (one for each node).

Schemes	Canard-Gouget [7]	Canard *et al* [8]	Our work
Standard Model	no	yes	yes
Public Parameters	$2^{n+3}\|q\| + 1\ pk$	$(n+2)\ pk$ $+ (1+(n+1)2^n)\ \mathbb{G}_2$ $+ (2^{n+2}+3)\ \mathbb{G}_1$ $+ (2^{n+1}-1)\ \|\texttt{Sign}\|$	$2\ pk$ $+ (2^{n+2}-1)\ \mathbb{G}_2$ $+ (2^{n+2}+n+4)\ \mathbb{G}_1$ $+ 2^n\ \|\texttt{Sign}\|$
`Withdraw` Computations	$(2^{n+3}+2^{n+2}-5)\texttt{exp}$ $+ (n+2)$ `Sign`	1 `Sign`	1 `Sign`
Coin Size	$(2^{n+2}+n+1)\ \|q\|$ $+ (n+2)\ \|\texttt{Sign}\|$	$2\ \|p\| + \|\texttt{Sign}\|$	$2\ \|p\| + \|\texttt{Sign}\|$
`Spend` Computations	$NIZK\{$ 3 $\texttt{exp}^*$ $+$ 2 `Sign` $+$ 2 `Pair` $\}$ $+$ 1 `exp`	$NIZK\{$ 2 `exp` $+$ 2 `Sign` $\}$ $+$ 3 `exp` $+$ 1 `Sign`	$NIZK\{$ 4 `exp` $+$ 2 `Sign` $+$ 2 `Pair` $\}$ $+$ 7 `exp` $+$ 1 `Sign`
Transfer size of `Spend`	$3\ \|q\| + \|NIZK\|$	$2\ \mathbb{G}_1 + 1\ \|\texttt{Sign}\|$ $+\|NIZK\|$	$4\ \mathbb{G}_1 + 1\ \|\texttt{Sign}\|$ $+ 1\ \mathbb{G}_2 + \|NIZK\|$
`Deposit` Computations	2^{l+1}`exp`	2^n `Pair`	2^{l+1} `Pair`
`Deposit` size	$2^l\ \|q\| + \|\texttt{Spend}\|$	$2^n\ \mathbb{G}_T + \|\texttt{Spend}\|$	$2^l\ \mathbb{G}_T + \|\texttt{Spend}\|$

Fig. 3. Efficiency comparison between related works and our construction for coins of value 2^n and `Spend` and `Deposit` of value 2^l ($l \leq n$). The space and times complexities are given from the user's point of view. `exp` refers to an exponentiation, `pair` to a pairing computation, `Sign` to the cost of the signature issuing protocol whose public key is pk. $NIZK\{\texttt{exp}\}$ denotes the cost of a $NIZK$ proof of a multi-exponentiation equation, $NIZK\{\texttt{pair}\}$ the one of a pairing product equation and $NIZK\{\texttt{Sign}\}$ the one of a valid signature. $NIZK\{\texttt{exp}^*\}$ refers to the cost of a proof of equality of discrete logarithms in groups of different orders.

Another way of proving the validity of the pair (g_s, h_s) is to notice that, for every $f \in \mathcal{L}(|s|)$, there is a leaf $\ell \in \mathcal{F}_n$ such that $s||f = \ell$. Therefore, $e(g_s, \widetilde{g}_{|s|,f}) = e(g_\ell, \widetilde{g})$ and $e(h_s, \widetilde{g}_{|s|,f}) = e(h_\ell, \widetilde{g})$. Since the element $\widetilde{g}_{|s|,f}$ is common to every node of level $|s|$, it can be revealed by the user so only the validity of the pair (g_ℓ, h_ℓ) remains to be proved. Therefore, the bank can generate only one key pair $(\mathsf{sk}_0, \mathsf{pk}_0)$ (instead of $n+1$ such key pairs) and provide 2^n signatures $\tau_\ell \leftarrow \Sigma_0.\texttt{Sign}(\mathsf{sk}_0, (g_\ell, h_\ell))$ for all the leaves (instead of $2^{n+1}-1$, for all the nodes). This slightly increases the size of the proof since the user will have to commit to (g_ℓ, h_ℓ) and prove statements on them, but this allows to significantly reduce the size of bpk.

The security of our divisible e-cash system is stated by the following theorem, proved in Appendix A.

Theorem 6. *In the standard model, assuming that the hash function H is collision-resistant, our divisible e-cash system is anonymous under the* SXDH *and the* EMXDH *assumptions, traceable if Σ_0 is an EUF-SCMA secure signature scheme and Σ_1 is an EUF-CMA secure signature scheme, and achieves the exculpability property under the $q-$* SDH *assumption if Σ_{ots} is a SUF-secure one-time signature scheme.*

5.2 Efficiency

We compare in Figure 3 the efficiency of our construction (including the remark 5) with the one of [8], which is the most efficient scheme regarding the `Withdraw` and the `Spend` protocols, and with the one from [7], whose `Deposit` protocol is less expensive but which is only compatible with the random oracle model.

For proper comparison, we add the elements $\widetilde{g}_{s\mapsto f}$ (see Section 4.1) to the public parameters of [8].

For a 128-bits security level, we have (see [13]) $|q| = |\mathbb{G}_T| = 3072$, $|p| = |\mathbb{G}_1| = 256$ and $|\mathbb{G}_2| = 512$ by using Barreto-Naehrig curves [3]. Therefore, for $n = 10$ (allowing to divide the coin in 1024 parts), the public parameters of [7] require 3.1 MBytes of storage space, those of [8] require 1.1 MBytes while ours only require 525 KBytes.

Compared to [8], the main advantage of our solution lies in the `Deposit` protocol. Indeed, in the former solution, the bank has to compute and store 2^n serial numbers z_i (most of them being invalid) for each transaction, no matter which value was spent. Considering the number of transactions that

a payment system may have to handle, this may become cumbersome. Our solution significantly alleviates the computing and storage needs of the bank since the number of z_i that it must recover is exactly the same as the spent value. However, this improvement implies a slight increase of the complexity of the `Spend` protocol but we argue that the trade-off remains reasonable. Moreover, it is possible to get rid of the 2 `Pair` in the $NIZK$ proof, at the cost of increasing the public parameters, by not taking into account the Remark 5.

6 Fair Divisible E-Cash System

The protocol described above achieves the strongest level of anonymity where users cannot be identified as long as they remain honest. However, it may be necessary for legal reasons to allow some entity (*e.g.* the police) to identify any spender. Our construction can be modified to add such an entity, that we will call an *opener*, leading to a *fair* divisible E-cash system. The point is that these modifications will decrease the complexity of our construction and weaken the assumption underlying its anonymity.

Let us consider the `Setup` algorithm defined in Section 4.2. A trusted authority was needed to generate the scalars $(a_0, y\epsilon, \{a_i, y_{i,0}, y_{i,1}\}_{i=1}^{i=n})$ involved in the construction of the tree. Indeed, an entity knowing them can easily break the anonymity of the scheme (but not its exculpability or its traceability).

Let us assume that the `Setup` algorithm is run by the opener. Since every transcript of a `Spend` protocol contains a pair $t_s = (g^r, g_s^x \cdot k_{n-|s|}^r)$ for some coin secret x and some random scalar r, the opener can recover:

$$g^x \leftarrow (g_s^x \cdot k_{n-|s|}^r \cdot (g^r)^{-a_{|s|}})^{(y_\epsilon \prod_{i=1}^{|s|} y_{i,b_i})^{-1}}$$

which only depends on the coin secret x. It then only remains to link this value with the user's identity. One way to achieve this is to slightly modify the `Withdraw` protocol by requiring that users also send g^x, prove that it is well formed and send a signature on it under `upk`. These elements will then be stored by the bank in a public register that will be used by the opener to identify spenders (the signature will ensure the exculpability property).

This new way of identifying spenders allows to alleviate our construction. Regarding the `Spend` protocol, computing the pair v_s (and proving statement about it) is no longer necessary since it was only useful to identify double-spenders. Regarding the public parameters, the elements h_s can be discarded since they were only involved in the computation of v_s. Finally, the `Identify` algorithm which suffers from a linear cost in the number of users is replaced by the constant-time computation of g^x described above.

Another benefit of this fair divisible E-cash system is that its anonymity now relies on the following assumption, which is clearly weaker than the EMXDH assumption (see Definition 3):

Definition 7 (Weak EMXDH Assumption). *Given* $(g, g^a, g^x, \widetilde{g}, \widetilde{g}^a) \in \mathbb{G}_1^3 \times \mathbb{G}_2^2$, $(\{g^{y^i}\}_{i=1}^{i=n}, \{g^{x \cdot y^i}\}_{i=1}^{i=n-1}, \{\widetilde{g}^{1/y^i}\}_{i=1}^{i=n}) \in \mathbb{G}_1^{2n-1} \times \mathbb{G}_2^n$, *as well as* $g^z \in \mathbb{G}_1$, *it is hard to distinguish whether* $z = x \cdot y^n/a$ *or* z *is random.*

7 Conclusion

In this work we have proposed a new divisible E-cash system which improves the state-of-the art paper [8] by addressing its two downsides, namely the storage and computational costs of the deposit protocol and the size of the public parameters. Our solution relies on a new way of constructing the binary tree which induces several modifications compared to [8] leading to the first efficient and scalable divisible E-cash system secure in the standard model.

Acknowledgments

This work was supported in part by the French ANR Projects ANR-12-INSE-0014 SIMPATIC and ANR-11-INS-0013 LYRICS, and in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

1. Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, August 2011.
2. Man Ho Au, Willy Susilo, and Yi Mu. Practical anonymous divisible e-cash from bounded accumulators. In Gene Tsudik, editor, *FC 2008*, volume 5143 of *LNCS*, pages 287–301. Springer, January 2008.
3. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, August 2005.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
5. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, May 2005.
6. Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 482–497. Springer, May 2007.
7. Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 82–97. Springer, January 2010.
8. Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible e-cash made practical. In Jonathan Katz, editor, *PKC '15*, volume 9020 of *Lecture Notes in Computer Science*, pages 77–100. Springer, 2015. Cryptology ePrint Archive, Report 2014/785, http://eprint.iacr.org/.
9. Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. Easy come - easy go divisible cash. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 561–575. Springer, May / June 1998.
10. Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.
11. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.
12. David Chaum and Torben P. Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 390–407. Springer, May 1992.
13. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
14. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
15. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.
16. Malika Izabachène and Benoît Libert. Divisible E-cash in the standard model. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 314–332. Springer, May 2012.
17. Toru Nakanishi and Yuji Sugiyama. Unlinkable divisible electronic cash. In Josef Pieprzyk, Eiji Okamoto, and Jennifer Seberry, editors, *Information Security, Third International Workshop, ISW 2000, Wollongong, NSW, Australia, December 20-21, 2000, Proceedings*, volume 1975 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 2000.
18. Tatsuaki Okamoto. An efficient divisible electronic cash scheme. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 438–451. Springer, August 1995.
19. Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, August 1991.
20. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, August 1989.

A Security Proofs

In [8], the security notions were formally defined through games that we recall in Appendix C. In all of them, the adversary $\mathcal{A}$ can add new users (either honest or corrupted) to the system, corrupt existing ones (in which case $\mathcal{A}$ recovers every secret known by them) or ask them to spend any value. This is modelled by queries to $\mathcal{O}\texttt{Add}$, $\mathcal{O}\texttt{AddCorrupt}$, $\mathcal{O}\texttt{Corrupt}$ and $\mathcal{O}\texttt{Spend}$ oracles. According to each security notion, $\mathcal{A}$ has also access to additional oracles that we define below.

A.1 Proof of Traceability

The goal of the adversary $\mathcal{A}$ is to output, after q_w withdraws, u valid transcripts $\{(2^{\ell_i}, Z_i, \Pi_i)\}_{i=1}^{u}$ such that $\texttt{Identify}((2^{\ell_i}, Z_i, \Pi_i), (2^{\ell_j}, Z_j, \Pi_j)) = \perp$ for all $i \neq j$ and $\sum_{i=1}^{u} 2^{\ell_i} > q_w \cdot 2^n$. This models a coalition of malicious users which manages to spend more than it has withdrawn without incriminating one of its members. To succeed, $\mathcal{A}$ has also access to a $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ oracle which executes the bank's side of the `Withdraw` protocol.

Let us consider a successful adversary $\mathcal{A}$. We distinguish the three following cases:

- Type-1 Forgeries: $\exists i$ such that Π_i contains commitments to a pair (g_s, h_s) which was never signed by the bank.
- Type-2 Forgeries: $\exists i$ such that Π_i contains commitments to a pair $(u_1^{\mathsf{usk}}, u_2^x)$ which was never signed by the bank during a $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ query.
- Type-3 Forgeries: $\forall 1 \leq i \leq u$, $\exists \tau_s$ in bpk which is a valid signature on the pair (g_s, h_s) committed in Π_i and the pairs $(u_1^{\mathsf{usk}}, u_2^x)$ involved in this transcript were signed by the bank during a $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ query.

The Remark 4 implies that type-3 forgeries can only occur with negligible probability. Indeed, let $x_1,...,x_{q_w}$ be the secret values associated with coins withdrawn by $\mathcal{A}$ during the $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ queries. Since an amount of $\sum_i 2^{\ell_i}$ has been deposited, the bank has computed $\sum_i 2^{\ell_i}$ serial numbers z_i. All of them are included in $\{G^{x_i \cdot y_\epsilon} \prod_{j=1}^{n} y_{j,b_j}\}_{b_1,...,b_n \in \mathcal{F}_n, 1 \leq i \leq q_w}$ (otherwise $\mathcal{A}$ would be a Type-2 forger) which contains $q_w \cdot 2^n < \sum_i 2^{\ell_i}$ elements. Therefore, there are at least $i \neq j$ such that $z_i = z_j$. Such serial numbers are associated with the same secret value x (see Remark 4) and so with the same public key upk which would have been returned by the $\texttt{Identify}$ algorithm.

Lemma 8. *Any Type-1 forger $\mathcal{A}$ succeeding with probability ϵ can be converted into an adversary against the EUF-SCMA security of Σ_0 succeeding with probability $\epsilon/n + 1$.*

Proof. The reduction $\mathcal{R}$ generates the public parameters as in the $\texttt{Setup}$ algorithm and selects $i^* \in [0, n]$. He then sends (g_s, h_s) for every $s \in \mathcal{S}_n$ such that $|s| = i^*$ to the challenger of the EUF-SCMA security experiment which returns the signatures τ_s along with a public key pk. $\mathcal{R}$ then generates the other key pairs $(\mathsf{sk}_0^{(i)}, \mathsf{pk}_0^{(i)})$ for $i \neq i^*$ and uses them to sign the other nodes. Finally, it sets $\mathsf{pk}_0^{(i^*)} = \mathsf{pk}$ and publishes $\{\mathsf{pk}_0^{(i)}\}_i$ along with the signatures τ_s. Since the keys $\mathsf{sk}_0^{(i)}$ are no longer involved in the protocol, $\mathcal{R}$ is able to answer any query.

The game ends when $\mathcal{A}$ outputs u transcripts such that one of them, $(2^\ell, Z, \Pi)$, contains commitments to a pair (g_s, h_s) which was never signed by the bank. The soundness of the proof Π implies that it also contains commitments to an element τ_s such that $\Sigma_0.\texttt{Verify}((g_s, h_s), \tau_s, \mathsf{pk}_0^{n-\ell}) = 1$. If $\ell \neq i^*$ then $\mathcal{R}$ aborts. Else, τ_s is a valid forgery on (g_s, h_s) under pk which can be used to break the EUF-SCMA security of Σ_0.

Lemma 9. *Any Type-2 forger $\mathcal{A}$ succeeding with probability ϵ can be converted into an adversary against the EUF-CMA security of Σ_1 with the same advantage.*

Proof. The reduction $\mathcal{R}$ generates the public parameters and its public key as usual excepts that it sets pk_1 as pk, the public key received from the challenger $\mathcal{C}$ of the EUF-CMA security. $\mathcal{R}$ can then directly answer all the queries except the $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ ones for which it will forward the pairs $(u_1^{\mathsf{usk}}, u_2^x)$ to $\mathcal{C}$ and return the resulting signature σ to $\mathcal{A}$.

The game ends when $\mathcal{A}$ outputs u transcripts such that one of them, $(2^\ell, Z, \Pi)$, contains commitments to a pair $(u_1^{\mathsf{usk}}, u_2^x)$ which was never signed by the bank during a $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}$ query. Again, the soundness of the proof implies that it also contains commitments to an element σ such that $\Sigma_1.\texttt{Verify}((u_1^{\mathsf{usk}}, u_2^x), \sigma, \mathsf{pk}_1) = 1$. Such a forgery can then be used to break the EUF-CMA security of Σ_1.

A.2 Proof of Exculpability

An adversary $\mathcal{A}$ against the exculpability property must output two transcripts (v_1, Z_1, Π_1) and (v_2, Z_2, Π_2) which accuse an honest user upk of double-spending, *i.e.* such that $\texttt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2)) = \mathsf{upk}$. To succeed, $\mathcal{A}$ has access to a $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$ oracle which executes the user's side of the $\texttt{Withdraw}$ protocol. This oracle will be used by $\mathcal{A}$ playing the role of the bank against an honest user.

Let us consider a successful adversary $\mathcal{A}$. At least one of the two transcripts that it outputs was not produced by the reduction $\mathcal{R}$ during a $\mathcal{O}\texttt{Spend}$ query (otherwise $\mathcal{R}$ would have double-spent its own coins) but was forged by it. We distinguish the two following cases:

– Type-1 forgeries: the public key pk_{ots} of the one-time signature scheme used in this forged transcript is one of those used by $\mathcal{R}$ to answer $\mathcal{O}\mathtt{Spend}$ queries.
– Type-2 forgeries: pk_{ots} was never used by $\mathcal{R}$.

The SUF-security of the OTS Σ_{ots} implies that Type-1 forgeries can only occur with negligible probability. Indeed, recall that the one-time signature scheme is used to sign all the elements involved in the transcript. Since the forged transcript is different from those returned by $\mathcal{R}$, it requires a new one-time signature η, valid under pk_{ots}, which is therefore a forgery. Any Type-1 forger can thus be converted into an adversary against the SUF-security of Σ_{ots}.

Lemma 10. *Let q_s (resp. q_a) be a bound on the number of $\mathcal{O}\mathtt{Spend}$ queries (resp. $\mathcal{O}\mathtt{Add}$ queries). Any Type-2 forger $\mathcal{A}$ succeeding with probability ϵ can be converted into an adversary against the $q_s - \mathsf{SDH}$ assumption succeeding with probability ϵ/q_a.*

Let $(g, g^{\alpha}, \ldots, g^{\alpha_{q_s}})$ be a $q_s - \mathsf{SDH}$ challenge, $\mathcal{R}$ will make a guess on the user upk^* framed by $\mathcal{A}$ and will act as if its secret key was α. Therefore, it selects $1 \leq i^* \leq q_a$ and generates the public parameters as in the $\mathtt{Setup}$ algorithm except that it sets u_1 as g^z for some random $z \in \mathbb{Z}_p$). Next, it computes q_s key pairs $(\mathsf{sk}_{ots}^{(i)}, \mathsf{pk}_{ots}^{(i)}) \leftarrow \Sigma_{ots}.\mathtt{Keygen}(1^k)$ and sets w as $g^{\prod_{i=1}^{q_s}(\alpha + H(\mathsf{pk}_{ots}^{(i)}))}$ (which is possible using the $q_s - \mathsf{SDH}$ challenge [4], since the exponent is a polynomial in α of degree q_s). The reduction will answer the oracle queries as follows.

– $\mathcal{O}\mathtt{Add}()$ queries: When the adversary makes the i-th $\mathcal{O}\mathtt{Add}$ query to register a user, $\mathcal{R}$ runs the $\mathtt{Keygen}$ algorithm if $i \neq i^*$ and set $\mathsf{upk}^* \leftarrow g^{\alpha}$ otherwise.
– $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ queries: $\mathcal{R}$ returns the secret key if $\mathsf{upk} \neq \mathsf{upk}^*$ and aborts otherwise.
– $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$ queries: $\mathcal{R}$ stores the public key which is now considered as registered.
– $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$ queries: $\mathcal{R}$ acts normally if $\mathsf{upk} \neq \mathsf{upk}^*$ and simulates the interactive proof of knowledge of α otherwise.
– $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, 2^{\ell})$ queries: $\mathcal{R}$ acts normally if $\mathsf{upk} \neq \mathsf{upk}^*$. Else, to answer the $j-$th query on upk^*, it computes $\mu \leftarrow g^{\prod_{i=1, i\neq j}^{q_s}(\alpha + H(\mathsf{pk}_{ots}^{(i)}))}$ which verifies $\mu = w^{\frac{1}{\alpha + H(\mathsf{pk}_{ots}^{(j)})}}$ and uses $\mathsf{sk}_{ots}^{(j)}$ as in the $\mathtt{Spend}$ protocol.

The adversary then outputs two valid transcripts $(2^{\ell_1}, Z_1, \Pi_1)$ and $(2^{\ell_2}, Z_2, \Pi_2)$ which accuse upk of double-spending. If $\mathsf{upk} \neq \mathsf{upk}^*$ then $\mathcal{R}$ aborts which will occur with probability $1/q_a$. Else, the soundness of the proof implies that the forged transcript was signed using a key sk_{ots} and so that the proof involves an element $\mu = w^{\frac{1}{\alpha + H(\mathsf{pk}_{ots})}}$. Since here we consider Type-2 attacks, $\mathsf{pk}_{ots} \notin \{\mathsf{pk}_{ots}^{(i)}\}_i$, so $\mathcal{R}$ extracts from the proof the element μ which can be used to break the $q_s - \mathsf{SDH}$ assumption in $\mathbb{G}_1$ (as in [4]).

A.3 Proof of Anonymity

An adversary $\mathcal{A}$ against the anonymity property must distinguish, among two honest users of its choice, who was involved in a spending. To succeed, it has also access to the $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ oracle defined above. Let us consider a successful adversary $\mathcal{A}$ with advantage ϵ. We construct a reduction $\mathcal{R}$ using $\mathcal{A}$ against the EMXDH assumption.

Let $(g, g^a, g^t, \{g^{y^i}\}_{i=1}^{i=n}, \{g^{t\cdot y^i}\}_{i=1}^{i=n}, \{g^{x\cdot y^i}\}_{i=0}^{i=n-1}, \{g^{x\cdot t\cdot y^i}\}_{i=0}^{i=n-1}, \{\widetilde{g}^{y^{-i}}\}_{i=0}^{i=n}, \widetilde{g}^a) \in \mathbb{G}_1^{4n+3} \times \mathbb{G}_2^{n+2}$ and $(g^{z_1}, g^{z_2}) \in \mathbb{G}_1^2$ be the corresponding challenge. The reduction $\mathcal{R}$ selects a random depth d^*, and a random node $s^* = b_1^* \ldots b_{|s^*|}^*$ at depth $|s^*| = d^*$. It will act as if $y_{\epsilon} = y^n$, $a_{d^*} = a$ and, for $1 \leq i \leq d^*$, $y_{i,b_i^*} = u_{i,b_i^*} \xleftarrow{\$} \mathbb{Z}_p$ and $y_{i,\bar{b}_i^*} = y^{-1} \cdot u_{i,\bar{b}_i^*}$ with $u_{i,\bar{b}_i^*} \xleftarrow{\$} \mathbb{Z}_p$. We define, for each $s = b_{\ell} \ldots b_{|s|}$, the integer $d_s = |\{\ell \leq i \leq min(|s|, d^*) : b_i \neq b_i^*\}|$. To generate the public parameters, the challenger

– sets (h, u_1, u_2) as (g^t, g^{d_1}, g^{d_2}) for random $d_1, d_2 \xleftarrow{\$} \mathbb{Z}_p$
– sets g_{ϵ} as g^{y^n}
– selects, for $1 \leq i \neq d^* \leq n$ $a_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $g_i \leftarrow g^{a_i}$
– sets g_{d^*} as g^a

– selects, for $1 \leq i \leq n$, $u_{i,0}, u_{i,1} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$(g_s, h_s) \leftarrow ((g^{y^{n-d_s}})^{\prod_{i=1}^{|s|} u_{i,b_i}}, (g^{t \cdot y^{n-d_s}})^{\prod_{i=1}^{|s|} u_{i,b_i}}), \quad \text{for each node } s = b_1 \ldots b_{|s|}$$

– computes, for every $0 \leq i \neq d^* \leq n$ and every $f \in \mathcal{L}_i$,

$$(\widetilde{g}_{i,f}, \widetilde{h}_{i,f}) \leftarrow ((\widetilde{g}^{y^{-d_f}})^{\prod_{j=i+1}^{n} u_{j,b_j}}, \widetilde{g}_{i,f}^{-a_i})$$

– computes, for every $f \in \mathcal{L}_{|s^*|}$,

$$(\widetilde{g}_{d^*,f}, \widetilde{h}_{d^*,f}) \leftarrow (\widetilde{g}^{\prod_{j=d^*+1}^{n} u_{j,b_j}}, (\widetilde{g}^{a})^{-\prod_{j=d^*+1}^{n} u_{j,b_j}})$$

Finally, $\mathcal{R}$ also generates a common reference string for the perfect witness indistinguishability setting. Such a CRS is indistinguishable from the one generated in the `Setup` algorithm under the SXDH assumption.

The point is that, for each s which is not a prefix or a descendant of s^*, we have $d_s > 0$. Therefore, the associated element g_s (resp. h_s) is equal to $g^{y^j \cdot r_s}$ (resp. $g^{t \cdot y^j \cdot r_s}$) for some $j < n$ and some $r_s \in \mathbb{Z}_p$ which is known to $\mathcal{R}$. The reduction can then use s to answer a $\mathcal{O}$`Spend` query since the challenge contains the pairs $(g^{x \cdot y^i}, g^{x \cdot t \cdot y^i})$ for $i < n$.

We denote by c_{upk} the value spent by the user whose public key is upk during $\mathcal{O}$`Spend` queries and by m_{upk} the number of coins that he withdrew. Let q_w be a bound on the number of $\mathcal{O}$`Withdraw` queries, $\mathcal{R}$ randomly selects i^* from $[0,q_w]$ and answers to the oracle queries as follows:

– $\mathcal{O}$`Add`() queries: $\mathcal{R}$ runs the `Keygen` algorithm and returns upk (or mpk).
– $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$ queries: When the adversary asks the i^{th} $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ query, the reduction acts normally if $i \neq i^*$ and as if the secret value of the coin is x otherwise (by sending $(g^x)^{d_2}$ and simulating the proof of knowledge, since x is not known by $\mathcal{R}$). The chosen public key corresponding to this last case will be denoted upk^*.
– $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ queries: $\mathcal{R}$ acts normally if the query was not made on upk^*. Else, it aborts the experiment.
– $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: $\mathcal{R}$ stores the public key which is now considered as registered.
– $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, 2^\ell)$ queries: $\mathcal{R}$ is able to deal with any of these queries if $\mathsf{upk} \neq \mathsf{upk}^*$. Else, the reduction is able to answer as long as $c_{\mathsf{upk}^*} \leq m_{\mathsf{upk}^*} \cdot 2^n - 2^{n-d^*} - 2^\ell$ (and aborts otherwise) since this condition means that there is at least one unspent node s (of depth $n - \ell$) which is not a prefix or a descendant of s^*. Therefore, $d_s > 0$ and so, as explained above, computing g_s^x and h_s^x is possible. The reduction can then return a valid pair $t_s \leftarrow (g^{r_1}, g_s^x \cdot g_{n-\ell}^{r_1})$ and a valid pair $v_s \leftarrow (g^{r_2}, (\mathsf{upk}^*)^R \cdot h_s^x \cdot g_{n-\ell}^{r_2})$ where $R \leftarrow H(info)$ and simulates the non-interactive proof (which is possible since we use a simulated CRS).

During the challenge phase, $\mathcal{A}$ outputs $\{\mathsf{upk}_0, \mathsf{upk}_1\}$ along with a value 2^ℓ. Of course, it is assumed that none of these users has spent more than $m_{\mathsf{upk}_b} \cdot 2^n - 2^\ell$. If $\mathsf{upk}^* \notin \{\mathsf{upk}_0, \mathsf{upk}_1\}$ or $\ell \neq n - d^*$ (*i.e.* s^* is not associated with the right value 2^ℓ) then $\mathcal{R}$ aborts. Else, $\mathcal{R}$ selects two random scalars $r, r' \xleftarrow{\$} \mathbb{Z}_p$ and returns, along with simulated proofs,

$$((g^{z_1})^{-\prod_{i=1}^{|s^*|} u_{i,b_i^*}} \cdot g^r, g^{a \cdot r}) \text{ and } ((g^{z_2})^{-\prod_{i=1}^{|s^*|} u_{i,b_i^*}} \cdot g^{r'}, (\mathsf{upk}^*)^R \cdot g^{a \cdot r'})$$

– in the case that $(z_1, z_2) = (x \cdot y^n/a, x \cdot t \cdot y^n/a)$ then
  • the first pair is $(g^{r_1}, g_{s^*}^x \cdot g_{|s^*|}^{r_1})$ with $r_1 = r - z_1 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*}$:

$$\begin{aligned} g_{s^*}^x \cdot g_{|s^*|}^{r_1} &= g^{(x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g^{a \cdot (r - z_1 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*})} \\ &= g^{(x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g^{a \cdot r} \cdot g^{(-x \cdot y^n) \prod_{i=1}^{|s*|} u_{i,b_i^*}} = g^{a \cdot r} \end{aligned}$$

- the second pair is $(g^{r_2}, (\mathsf{upk}^*)^R \cdot h_{s^*}^x \cdot g_{|s^*|}^{r_2})$ with $r_2 = r' - z_2 \cdot t \prod_{i=1}^{|s*|} u_{i,b_i^*}$:

$$\begin{aligned} h_{s^*}^x \cdot g_{|s^*|}^{r_2} &= h_{s^*}^x \cdot g_{|s^*|}^{-z_2 \cdot t \prod_{i=1}^{|s*|} u_{i,b_i^*}} \cdot g_{|s^*|}^{r'} \\ &= (g_{s^*}^x \cdot g_{|s^*|}^{-z_2 \cdot \prod_{i=1}^{|s*|} u_{i,b_i^*}})^t \cdot g^{a \cdot r'} = g^{a \cdot r'} \end{aligned}$$

  Therefore, the pairs returned by $\mathcal{R}$ are valid.
- in the random case, these pairs are respectively $(g^\alpha \cdot g^r, g^{a \cdot r})$ and $(g^\beta \cdot g^{r'}, (\mathsf{upk}^*)^R \cdot g^{a \cdot r'})$, and perfectly hide upk^*.

The bit returned by $\mathcal{A}$ can then be used against the EMXDH assumption.

The reduction will not abort if it correctly guessed the value spent during the challenge phase (*i.e.* $\ell = n - d^*$) and if the coin withdrawn during the i^{*th} query belongs to upk_0 or upk_1. The advantage of $\mathcal{R}$ in breaking the assumption is then greater than $2\epsilon/((n+1) \cdot q_w)$.

B Security of the EMXDH Assumption

First, let us consider the weak-EMXDH assumption defined in Section 6. Informally, this assumption relies on the fact that, since $g^z \in \mathbb{G}_1$, we can combine it with every element of $\mathbb{G}_2$ (but not $\mathbb{G}_1$ since we use an asymmetric bilinear map) provided by the challenge. Therefore, the assumption holds if $e(g^z, \widetilde{g}^a)$ and $e(g^z, \widetilde{g}^{1/y^j})$, for $0 \leq j \leq n$, are undistinguishable from random elements of $\mathbb{G}_T$, given all the inputs from the challenge.

If $z = x \cdot y^n/a$ then $e(g^z, \widetilde{g}^a) = G^{x \cdot y^n}$ and $e(g^z, \widetilde{g}^{y^{-j}}) = G^{x \cdot y^{n-j}/a}$, for $G = e(g, \widetilde{g})$. While all the other combinations lead to G, G^{a^2}, G^x, $\{G^{y^i}\}_{i=-n}^{i=n}$, $\{G^{ay^i}\}_{i=-n}^{i=n}$, $\{G^{xy^i}\}_{i=-n}^{i=n-1}$, $\{G^{axy^i}\}_{i=0}^{i=n-1}$. No linear combination of the latter can help to distinguish the former.

The elements provided in the challenge allow (see Appendix A.3) to generate the public parameters g_s for each node $s \in \mathcal{S}_n$ as well as the pairs $(\widetilde{g}_{i,f}, \widetilde{h}_{i,f})$ for every $0 \leq i \leq n$ and $f \in \mathcal{F}_n$ but also to compute the element g_s^x involved in the pair t_s sent during the Spend protocol. This is enough for the fair divisible E-cash system sketched in Section 6. Unfortunately, the anonymous version described in Section 5 also requires to provide, for each $s \in \mathcal{S}_n$, the elements $h_s = g_s^t$ for some $t \in \mathbb{Z}_p$. Such elements are used during a spending to compute the element $h_s^x = (g_s^x)^t$ involved in the pair v_s necessary to identify double-spenders without the help of an opener.

The problem is that the reduction $\mathcal{R}$ of Appendix A.3 cannot know the value t. Therefore, it is necessary to provide for most elements w of $\mathbb{G}_1$ of the challenge, the associated ones w^t, hence the EMXDH assumption.

C Security Games

We recall, for consistency, the security games defined in [8]. Each one of them makes use of the oracles described in Appendix A.

C.1 Traceability

The traceability experiment $\mathtt{Exp}_{\mathcal{A}}^{tra}(1^k, V)$ is defined on Figure 4. A divisible E-cash system is traceable if, for any probabilistic polynomial adversary $\mathcal{A}$, the advantage $\mathtt{Adv}_{\mathcal{A}}^{tra}(1^k, V) = \Pr[\mathtt{Exp}_{\mathcal{A}}^{tra}(1^k, V) = 1]$ is negligible.

C.2 Exculpability

The exculpability experiment $\mathtt{Exp}_{\mathcal{A}}^{excu}(1^k, V)$ is defined on Figure 5. A divisible e-cash system is *exculpable* if, for any probabilistic polynomial adversary $\mathcal{A}$, the advantage $\mathtt{Adv}_{\mathcal{A}}^{excu}(1^k, V) = \Pr[\mathtt{Exp}_{\mathcal{A}}^{excu}(1^k, V) = 1]$ is negligible.

$\mathtt{Exp}^{tra}_{\mathcal{A}}(1^k, V)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k, V)$
2. $(\mathsf{bsk}, \mathsf{bpk}) \leftarrow \mathtt{BKeygen}()$
3. $[(v_1, Z_1, \Pi_1), \ldots, (v_u, Z_u, \Pi_u)] \xleftarrow{\$} \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{B}}, \mathcal{O}\mathtt{Spend}}(\mathsf{bpk})$
4. If $\sum_{i=1}^{u} v_i > m \cdot V$ and $\forall i \neq j, \mathtt{Identify}((v_i, Z_i, \Pi_i), (v_j, Z_j, \Pi_j)) = \perp$, then return 1
5. Return 0

Fig. 4. Traceability Security Game

$\mathtt{Exp}^{excu}_{\mathcal{A}}(1^k, V)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k, V)$
2. $\mathsf{bpk} \leftarrow \mathcal{A}()$
3. $[(v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2)] \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}, \mathcal{O}\mathtt{Spend}}()$
4. If $\mathtt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2), \mathsf{bpk}) = \mathsf{upk}$ and upk not corrupted, then return 1
5. Return 0

Fig. 5. Exculpability Security Game

C.3 Anonymity

The anonymity experiment $\mathtt{Exp}^{anon-b}_{\mathcal{A}}(1^k, V)$, for $b \in \{0, 1\}$, is described on Figure 6. The notation c_{upk} and m_{upk} is defined in Appendix A.3. A divisible e-cash system is *anonymous* if, for any probabilistic polynomial adversary $\mathcal{A}$, the advantage $\mathtt{Adv}^{anon}_{\mathcal{A}}(1^k, V) = \Pr[\mathtt{Exp}^{anon-1}_{\mathcal{A}}(1^k, V)] - \Pr[\mathtt{Exp}^{anon-0}_{\mathcal{A}}(1^k, V)]$ is negligible.

$\mathtt{Exp}^{anon-b}_{\mathcal{A}}(1^k, V)$

1. $p.p. \leftarrow \mathtt{Setup}(1^k, V)$
2. $\mathsf{bpk} \leftarrow \mathcal{A}()$
3. $(v, \mathsf{upk}_0, \mathsf{upk}_1, \mathsf{mpk}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}, \mathcal{O}\mathtt{Spend}}()$
4. If upk_0 or upk_1 is not registered, then return 0
5. If $c_{\mathsf{upk}_i} > m_{\mathsf{upk}_i} \cdot V - v$ for $i \in \{0, 1\}$, then return 0
6. $(v, Z, \Pi) \leftarrow \mathtt{Spend}(\mathcal{C}(\mathsf{usk}_b, C, \mathsf{mpk}, v), \mathcal{A}())$
7. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}, \mathcal{O}\mathtt{Spend}^*}()$
8. If upk_0 or upk_1 has been corrupted, then return 0
9. Return $(b = b^*)$

Fig. 6. Anonymity Security Game