Cut Down the Tree to Achieve Constant Complexity in Divisible E-Cash

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 Research University, Paris, France

Abstract. Divisible e-cash, proposed in 1991 by Okamoto and Ohta, addresses a practical concern of electronic money, the problem of paying the exact amount. Users of such systems can indeed withdraw coins of a large value N and then divide it into many pieces of any desired values $V \leq N$. Such a primitive therefore allows to avoid the use of several denominations or change issues. Since its introduction, many constructions have been proposed but all of them make use of the same framework: they associate each coin with a binary tree, which implies, at least, a logarithmic complexity for the spendings.
In this paper, we propose the first divisible e-cash system without such a tree structure, and so without its inherent downsides. Our construction is the first one to achieve constant-time spendings while offering a quite easy management of the coins. It compares favorably with the state-of-the-art, while being provably secure in the standard model.

1 Introduction

Electronic payment systems have a strong impact on individual's privacy, and this is often underestimated by the users. Transaction informations, such as payee's identity, date and location, allow a third party (usually, the financial institution) to learn a lot of things about the users: individuals' whereabouts, religious beliefs, health status, etc, which can eventually be quite sensitive.

However, secure e-payment and strong privacy are not incompatible, as shown by Chaum in 1982 [Cha82]: he introduced the concept of electronic cash (*e-cash*), the digital analogue of regular cash, and in particular with its anonymity property. Typically, e-cash systems consider three kinds of parties, the bank, users and merchants. The bank issues coins, which can be withdrawn by users, and then be spent to merchants. Eventually, the merchants deposit the money on their account at the bank. It is better when the spending process does not involve the bank, in which case the e-cash system is said *offline*. Ideally, users and merchants should form a single set, which means that anyone receiving a coin should be able to spend it again without depositing it to the bank. Unfortunately, such a solution, called *transferable* e-cash implies [CP93] coins of growing size which quickly becomes cumbersome.

Although most of the features of regular cash, such as anonymity, can be reproduced by e-cash, there is one fundamental difference between these two systems: the latter can easily by duplicated, as any digital information. This property is a major issue for money, since dishonest users could spend several times the same coin to different merchants. To deter this bad behavior, e-cash systems must enable (1) detection of double-spending (*i.e.* the reuse of a spent coin), or alternatively over-spending (*i.e.* spending more money than withdrawn) and (2) identification of defrauders.

Unfortunately, achieving such properties becomes tricky when anonymity of transactions is required. Indeed, the bank can no longer trace the users' payments and check that, for each of them, the global amount spent remains lower than the amount he withdrew. To enable detection of double-spending/over-spending, most of the e-cash systems then make use of serial numbers: every coin is associated with a unique number, only known to its owner until he spends the coin. The serial number is indeed revealed during the transaction and stored by the bank in a database. The bank can thus detect any reuse of serial numbers and so any double-spending.

1.1 Divisible E-Cash

In 1991, Okamoto and Ohta [OO92] showed that e-cash can do more than simply emulate regular cash. They introduced the notion of *divisible* e-cash, where users withdraw coins of value N and

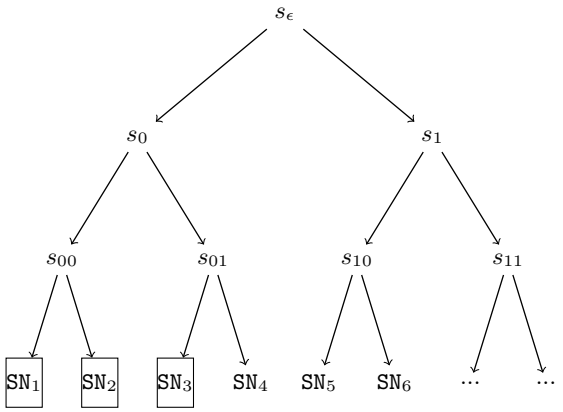

Fig. 1. Tree-based divisible coin

have the ability of dividing it into many pieces of any desired values $V_i \leq N$ such that $\sum_i V_i = N$. Such a property enables the user to pay the exact amount whatever the amount of the initially withdrawn coin was, which was a problem for traditional e-cash (and regular cash) systems. The authors proposed a framework representing each coin of value $N = 2^n$ by a binary tree where each leaf is associated with a serial number, and so with a value 1. When a user wants to spend a value $2^\ell \leq N$, he reveals an information related to a node s of depth $n - \ell$, allowing the bank to recover the 2^ℓ serial numbers associated with the leaves descending from s. The benefit of this tree structure is to provide a partial control on the amount of serial numbers the user reveals. The latter can indeed send them by batches of 2^ℓ, for any $0 \leq \ell \leq n$, which is much more efficient than sending them one by one, while ensuring that no information on serial numbers which do not descend from the spent nodes will leak.

Following this seminal work, a large number of constructions (including for example the following papers [CG07,ASM08,CG10,CPST15a,Mär15,CPST15b]) have been proposed, all of them making use of this framework, with a binary tree. In 2007, Canard and Gouget [CG07] proposed the first anonymous construction in the random oracle model, and recently, Canard *et al* [CPST15a] showed that both anonymity and efficiency can be achieved in the standard model.

However, this binary tree structure has a major downside: it is tailored to spend powers of 2. Unfortunately, such an event is unlikely in real life. In practice, to pay a value V, the users must write $V = \sum_i b_i \cdot 2^i$, for $b_i \in \{0, 1\}$ and then repeat the `Spend` protocol v times, where $v = \sum_i b_i$. Therefore, the *constant-time* property claimed by several constructions is somewhat misleading: spendings can be performed in constant-time as long as V is a power of 2 but not in the general case, and in the worst case the complexity is logarithmic.

Moreover, this structure makes the coin management slightly more difficult. Indeed, let us consider the case illustrated by the Figure 1, where a user has already spent a value $V_1 = 3$ and so revealed the first three serial numbers $\mathtt{SN}_1, \mathtt{SN}_2$ and $\mathtt{SN}_3$. Now assume that the user wants to spend a value $V_2 = 2$. He cannot use the node s_{01}, since $\mathtt{SN}_3$ has already been revealed and so must use s_{10} or s_{11}. This means that the serial number $\mathtt{SN}_4$ will remain isolated, and the user will have to spend it later as a unit. It is then necessary to maintain a list of unspent serial numbers and try to avoid the presence of several "holes" in the tree, which thereafter restricts a lot the value that can be spent at once.

1.2 Our Contribution

In this work, we aim at a greater simplicity and a better efficiency, and propose the first divisible e-cash system which truly achieves constant-time spendings. The main novelty of our construction it that we get rid of the tree structure and so of its inherent downsides that we have described above. Our scheme enables users to reveal, by sending a constant number of elements, the sequence of V serial numbers $\mathtt{SN}_j, \ldots, \mathtt{SN}_{j+V-1}$, for any j and V of their choice (provided that $j + V - 1 \leq N$), even if V is not a power of 2. If we reconsider the previous

example, this means that the user can now reveal, with a constant complexity, $\mathtt{SN}_4, \ldots, \mathtt{SN}_{4+V_2-1}$, for any value V_2.

We start from [CPST15a], which introduced the idea of a unique coin's structure, but make several changes to achieve constant-time spendings. The most important one is that we generate the public parameters in such a way that a same element can be used for spendings of any possible amount. This stands in sharp contrast with previous constructions where each element was associated with a node of the tree and so with a unique amount. More specifically, we use bilinear groups (*i.e.* 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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$) and set the N serial numbers of a coin as $\mathtt{SN}_j = e(s, \widetilde{g})^{x \cdot y^j}$, for $j = 1, \ldots, N$, where x is the coin's secret and $(y, s, \widetilde{g}) \in \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{G}_2$ are global parameters of the system (not all public). These parameters additionally contain the elements $s_j = s^{y^j} \in \mathbb{G}_1$, for $j = 1, \ldots, N$ and $\widetilde{g}_j = \widetilde{g}^{y^j} \in \mathbb{G}_2$, for $j = 1, \ldots, N-1$. The relations between all these elements (namely the fact that they all depend on y) are at the heart of the efficiency of our construction but have a strong impact on anonymity. Indeed, (1) they could be used by an adversary to link transactions together and (2) they make the anonymity property much more difficult to prove.

Regarding (2), the problem comes from the fact that the reduction in the anonymity proof must take all these relations into account while being able to reveal the non-critical serial numbers $\{e(s, \widetilde{g})^{x \cdot y^j}\}_{j=1}^{j^*-1} \cup \{e(s, \widetilde{g})^{x \cdot y^j}\}_{j=j^*+V^*}^{N}$ and to insert the challenge serial numbers in $\{e(s, \widetilde{g})^{x \cdot y^j}\}_{j=j^*}^{j^*+V^*-1}$, for any $j^*, V^* \in [1, N]$. Nonetheless, we manage to prove the anonymity of our construction under an assumption which, albeit new and rather complex, does not depend on either j^* and V^*. We stress that the latter point was far from obvious. We also investigate in Appendix C another way of generating the public parameters which allows to rely on a more classical assumption but at the cost of significant increase of the complexity (which nevertheless remains constant).

Regarding (1), we must pay attention to the way the serial numbers $\mathtt{SN}_i$, for $i = j, \ldots, j + V - 1$, are revealed during a spending of value V. For example, we show in Section 4.1 that the solution from [CPST15a] (namely sending s_j^x) would trivially be insecure in our setting. The user will then rather send s_j^x encrypted in a way that prevents anyone from testing relations between spendings while ensuring that only a specific amount of serial numbers can be recovered from it.

Our `Spend` protocol is then quite efficient: it mostly consists in sending an encryption of s_j^x along with a proof of well-formedness. As illustrated on Figure 3 of Section 5.2, it outperforms the state-of-the-art [Mär15,CPST15b], whose complexity logarithmically depends on the spent value V. Since spending is the operation subject to the strongest time constraints (for example, it should be performed in less than 300ms in a public transport system [GSM12]) we argue that our construction makes all the features of e-cash systems much more accessible.

1.3 Organization

In Section 2, we recall some definitions and present the computational assumptions underlying the security of our scheme. Section 3 reviews the syntax of a divisible E-cash system along with security properties definitions. We provide in Section 4 a high level description of our construction and a more detailed presentation in Section 5. The latter additionally contains a comparison with state-of-the-art. The security analysis is performed in Section 6. Eventually, we describe an instantiation of our divisible e-cash system in Appendix A and propose in Appendix C an alternative scheme which is less efficient, but whose anonymity relies on a more classical assumption. The hardness of our new assumption is proven in the generic bilinear group model in Appendix B.

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 $g \neq 1_{\mathbb{G}_1}$ 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 [GPS08] 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 exists between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction.

Although Type-1 pairings were mostly used in the early-age of pairing-based cryptography, they have been gradually discarded in favour of Type-3 pairings. Indeed, the latter offer a better efficiency and are compatible with several computational assumptions, such as the SXDH and the $N - \mathsf{MXDH}'$ ones we present below, which do not hold in the former.

2.2 Computational Assumptions

Our security analysis makes use of the SXDH, $q-\mathsf{SDH}$ [BB08] and $N-\mathsf{BDHI}$ [BB04] assumptions which have been considered reasonable for Type-3 pairings.

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 $(m, g^{\frac{1}{x+m}}) \in \mathbb{Z}_p \times \mathbb{G}_1$.*

Definition 3 ($N-$BDHI assumption). *Given $(\{g^{y^i}\}_{i=0}^N, \{\widetilde{g}^{y^i}\}_{i=0}^N) \in \mathbb{G}_1^{N+1} \times \mathbb{G}_2^{N+1}$, it is hard to compute $G = e(g, \widetilde{g})^{1/y} \in \mathbb{G}_T$.*

However, the anonymity of our construction relies on a new assumption, that we call $N-\mathsf{MXDH}'$. To provide more confidence in the latter, we first introduced a weaker variant, called $N-\mathsf{MXDH}$, that holds (as we prove it in Appendix B) in the generic bilinear group model for Type-3 pairings and next prove that both variants are actually related as stated in Theorem 6.

Definition 4. *$\forall N \in \mathbb{N}^*$, we define $C = N^3 - N^2$, $S = C + 1$, $E = N^2 - N$, $D = S + E$ and $P = D + C$, along with the following assumptions.*

- **($N-$MXDH assumption).** *Given $\{(g^{\gamma^k})_{k=0}^P, (g^{\alpha \cdot \delta \cdot \gamma^{-k}})_{k=0}^E, (g^{\chi \cdot \gamma^k})_{k=D+1}^P, (g^{\alpha \cdot \gamma^{-k}}, g^{\chi \cdot \gamma^k / \alpha})_{k=0}^C\} \in \mathbb{G}_1^{P+E+3S+1}$, as well as $(\widetilde{g}^{\gamma^k}, \widetilde{g}^{\alpha \cdot \gamma^{-k}})_{k=0}^C \in \mathbb{G}_2^{2S}$ and an element $g^z \in \mathbb{G}_1$, it is hard to decide whether $z = \delta + \chi \gamma^D / \alpha$ or z is random.*
- **($N-\mathsf{MXDH}'$ assumption).** *Given $\{(g^{\gamma^k}, h^{\gamma^k})_{k=0}^P, (g^{\alpha \cdot \delta \cdot \gamma^{-k}}, h^{\alpha \cdot \delta \cdot \gamma^{-k}})_{k=0}^E, (g^{\chi \cdot \gamma^k}, h^{\chi \cdot \gamma^k})_{k=D+1}^P$, $(g^{\alpha \cdot \gamma^{-k}}, g^{\chi \cdot \gamma^k / \alpha}, h^{\chi \cdot \gamma^k / \alpha})_{k=0}^C\} \in \mathbb{G}_1^{2P+5S+2E+2}$, as well as $(\widetilde{g}^{\gamma^k}, \widetilde{g}^{\alpha \cdot \gamma^{-k}})_{k=0}^C \in \mathbb{G}_2^{2S}$ and a pair $(g^{z_1}, h^{z_2}) \in \mathbb{G}_1^2$, it is hard to decide whether $z_1 = z_2 = \delta + \chi \gamma^D / \alpha$ or (z_1, z_2) is random.*

In Appendix C, we present another divisible e-cash protocol whose proof relies on a more classical assumption, but at the cost of larger public parameters and more complex (but still constant-size) protocols.

Regarding the $N - \mathsf{MXDH}$ assumption, the core idea is that the elements provided in an instance allow to compute the sets $\mathcal{S}_1 = \{e(g, \widetilde{g})^{\chi \cdot \gamma^k}\}_{k=0}^{S-1}$ and $\mathcal{S}_2 = \{e(g, \widetilde{g})^{\chi \cdot \gamma^k}\}_{k=D+1}^{P+C}$ but no element of $\mathcal{S}_3 = \{e(g, \widetilde{g})^{\chi \cdot \gamma^k}\}_{k=S}^{D}$. In the security proof, we will manage to force the V^* "challenge" serial numbers $\mathtt{SN}_{j^*}, \ldots, \mathtt{SN}_{j^*+V^*-1}$ (where V^* is the amount of the challenge transaction, *i.e* the one where the adversary tries to identify the spender) to belong to $\mathcal{S}_3$ while ensuring that the other ones belong to $\mathcal{S}_1 \cup \mathcal{S}_2$ and so can be simulated. This requires a great flexibility from the assumption, since the number V^* and the index j^* are adaptively chosen by the adversary. If N is the amount of the divisible coin, this means that it must be possible, for any $(j^*, V^*) \in [1, N]^2$, to insert $j^* - 1$ serial numbers in $\mathcal{S}_1$, V^* in $\mathcal{S}_3$ and $N + 1 - (j^* + V^*)$ in $\mathcal{S}_2$, all of these sets being constant. We show in Section 6.3 that this is the case when the integers C, S, E, D and P are chosen as in the above definition.

Theorem 5. *The $N-\mathsf{MXDH}$ assumption holds in the generic bilinear group model: after q_G group and pairing oracle queries, no adversary can solve the $N-\mathsf{MXDH}$ problem with probability greater than $2N^3 \cdot (7N^3+q_G)^2/p$.*

The proof, that is quite classical, can be found in Appendix B. It is worthy to note that the integer N will represent the amount of a divisible coin and so will remain negligible compared to p. For example, a typical value for N is 1000 which allows users to withdraw coins of value 10$, if the basic unit is the cent.

Theorem 6. *The $N-\mathsf{MXDH}'$ assumption holds if both the DDH assumption in $\mathbb{G}_1$ and the $N-\mathsf{MXDH}$ assumption hold.*

Proof. Let $\mathcal{A}$ be an adversary against the $N-\mathsf{MXDH}'$ assumption with a non-negligible advantage

$$\mathtt{Adv}(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z = \delta + \chi \cdot \gamma^D/\alpha] - \Pr[\mathcal{A}(\mathcal{S}, g^{z_1}, h^{z_2}) | z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p],$$

where $\mathcal{S}$ refers to the set of all elements, except g^{z_1} and h^{z_2}, provided in an $N-\mathsf{MXDH}'$ challenge. We define hybrid distributions:

$$\mathtt{Adv}_1(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z = \delta + \chi \cdot \gamma^D/\alpha] - \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z \xleftarrow{\$} \mathbb{Z}_p]$$
$$\mathtt{Adv}_2(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z \xleftarrow{\$} \mathbb{Z}_p] - \Pr[\mathcal{A}(\mathcal{S}, g^{z_1}, h^{z_2}) | z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p],$$

we then have: $\mathtt{Adv}(\mathcal{A}) = \mathtt{Adv}_1(\mathcal{A}) + \mathtt{Adv}_2(\mathcal{A})$.

Since $\mathtt{Adv}(\mathcal{A})$ is non-negligible, at least $\mathtt{Adv}_1(\mathcal{A})$ or $\mathtt{Adv}_2(\mathcal{A})$ is non-negligible.

In the former case, $\mathcal{A}$ can be used to break the $N-\mathsf{MXDH}$ assumption: from an $N-\mathsf{MXDH}$ instance, one can generate an $N-\mathsf{MXDH}'$ instance with a random scalar c and setting $h = g^c$. By running $\mathcal{A}$ on this instance, it gives a valid guess for it if and only if this would be a valid guess for the $N-\mathsf{MXDH}$ instance. The advantage is thus the same.

In the latter case, $\mathcal{A}$ can be used to break the DDH assumption in $\mathbb{G}_1$. Indeed, let (g, g^{z_1}, h, h^{z_2}) be a DDH challenge. One can compute a valid set $\mathcal{S}$ from g and h by using random (known) scalars α, γ and δ, and then run $\mathcal{A}$ on $(\mathcal{S}, g^{z_1}, h^{z_2})$. □

One can note that the $N-\mathsf{MXDH}$ and $N-\mathsf{MXDH}'$ assumptions would actually be equivalent if the former implied the DDH assumption in $\mathbb{G}_1$ (which does not seem to be true). Nevertheless, this theorem shows that the $N-\mathsf{MXDH}'$ assumption is not much stronger than the $N-\mathsf{MXDH}$ one, since the DDH assumption can be considered reasonable.

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) [GMR88] 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 variants, first with *selective chosen-message attacks* (SCMA) which restricts means for the adversary by limiting the oracle queries to be asked before having seen the key pk; or with *one-time signature* (OTS), which limits the adversary to ask one query only to the signing oracle; and with *strong unforgeability* (SUF) which relaxes the goal of the adversary which must now output a valid pair (m, σ) that was not returned by the signing oracle (a new signature for an already signed message is a valid forgery).

2.4 Groth-Sahai Proof Systems

In [GS08], Groth and Sahai proposed a non-interactive proof system, in the common reference string (CRS) model, which captures most of the relations for bilinear groups. There are two types of setup for the CRS that yield either perfect soundness or perfect witness indistinguishability, while being 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, for variables $\{X_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{X}_i\}_{i=1}^n \in \mathbb{G}_2$ and constant $t_T \in \mathbb{G}_T$, $\{A_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{B}_i\}_{i=1}^n \in \mathbb{G}_2$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}_p$:

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

- or multi-exponentiation equations, 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 $k \in \{1, 2\}$:

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

Multi-exponentiation equations and pairing-product equations such that $t_T = 1_{\mathbb{G}_T}$ also admit non-interactive zero-knowledge (NIZK) proofs at no additional cost.

3 Divisible E-cash System

We recall in this section the syntax and the security model of a divisible e-cash system, as described in [CPST15a].

3.1 Syntax

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

- $\texttt{Setup}(1^k, N)$: On input a security parameter k and an integer N, this probabilistic algorithm outputs the public parameters pp for divisible coins of global value N. We assume that pp are implicit to the other algorithms, and that they include k and N. 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 N 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 $\perp$.

$\mathtt{Exp}^{tra}_{\mathcal{A}}(1^k, N)$ – Traceability Security Game

1. $pp \leftarrow \mathtt{Setup}(1^k, N)$
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 $\sum_{i=1}^{u} V_i > m \cdot N$ 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

$\mathtt{Exp}^{excu}_{\mathcal{A}}(1^k, N)$ – Exculpability Security Game

1. $pp \leftarrow \mathtt{Setup}(1^k, N)$
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

$\mathtt{Exp}^{anon-b}_{\mathcal{A}}(1^k, N)$ – Anonymity Security Game

1. $pp \leftarrow \mathtt{Setup}(1^k, N)$
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_i is not registered for $i \in \{0, 1\}$, then return 0
5. If $c_{\mathsf{upk}_i} > m_{\mathsf{upk}_i} \cdot N - V$ 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. $c_{\mathsf{upk}_{1-b}} \leftarrow c_{\mathsf{upk}_{1-b}} + V$
8. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Corrupt}, \mathcal{O}\mathtt{AddCorrupt}, \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}, \mathcal{O}\mathtt{Spend}}()$
9. If upk_i has been corrupted for $i \in \{0, 1\}$, then return 0
10. Return $(b = b^*)$

Fig. 2. Security Games for Anonymous Divisible E-Cash

- $\mathtt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (V, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: This is an interactive protocol between a merchant $\mathcal{M}$ and the bank $\mathcal{B}$. $\mathcal{B}$ first checks the validity of the transcript (V, Z, Π) and that it has not already been deposited. If one of these conditions is not fulfilled, then $\mathcal{B}$ aborts and outputs $\perp$. At the end of the protocol $\mathcal{B}$ stores the V serial numbers $\mathtt{SN}_1, \ldots, \mathtt{SN}_V$ derived from Z in L or returns a transcript (V', Z', Π') such that $\mathtt{SN}_i$ is also a serial number derived from Z', for some $i \in [1, V]$.
- $\mathtt{Identify}((v_1, Z_1, \Pi_1), (v_2, Z_2, \Pi_2), \mathsf{bpk})$: 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.

3.2 Security Model

Informally, to reconcile the interests of all parties, a divisible e-cash system should (1) ensure detection of double-spending/over-spending and identification of the defrauders, (2) preserve privacy of its users, (3) ensure that none of them can be falsely accused of fraud. Regarding the first point, we recall that reuse of money cannot be prevented (since digital coin can always be duplicated) but the guarantee of being identified should constitute a strong incentive not to cheat. The third point implicitly ensures that a coin can only be spent by its owner.

These security properties were formally defined as *traceability*, *anonymity* and *exculpability* by the authors of [CPST15a]. For consistency, we recall the associated security games, in Figure 2, which make use of the following oracles:

- $\mathcal{O}\mathtt{Add}()$ is an oracle used by the adversary $\mathcal{A}$ to register a new honest user (resp. merchant). The oracle runs the $\mathtt{Keygen}$ algorithm, stores usk (resp. msk) and returns upk (resp. mpk) to $\mathcal{A}$. In this case, upk (resp. mpk) is said *honest*.
- $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ is an oracle used by $\mathcal{A}$ to corrupt an honest user (resp. merchant) whose public key is upk (resp. mpk). The oracle then returns the corresponding secret key usk (resp. msk) to $\mathcal{A}$ along with the secret values of every coin withdrawn by this user. From now on, upk (resp. mpk) is said *corrupted*.

– $\mathcal{O}\texttt{AddCorrupt}(\texttt{upk}/\texttt{mpk})$ is an oracle used by $\mathcal{A}$ to register a new corrupted user (resp. merchant) whose public key is upk (resp. mpk). In this case, upk (resp. mpk) is said *corrupted*. The adversary could use this oracle on a public key already registered (during a previous $\mathcal{O}\texttt{Add}$ query) but for simplicity, we do not consider such case as it will gain nothing more than using the $\mathcal{O}\texttt{Corrupt}$ oracle on the same public key.
– $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}(\texttt{upk})$ is an oracle that executes the user's side of the Withdraw protocol. This oracle will be used by $\mathcal{A}$ playing the role of the bank against the user with public key upk.
– $\mathcal{O}\texttt{Withdraw}_{\mathcal{B}}(\texttt{upk})$ is an oracle that executes the bank's side of the Withdraw protocol. This oracle will be used by $\mathcal{A}$ playing the role of a user whose public key is upk against the bank.
– $\mathcal{O}\texttt{Spend}(\texttt{upk}, V)$ is an oracle that executes the user's side of the Spend protocol for a value V. This oracle will be used by $\mathcal{A}$ playing the role of the merchant $\mathcal{M}$.

In the experiments, users are denoted by their public keys upk, $c_{\texttt{upk}}$ denotes the amount already spent by user upk during $\mathcal{O}\texttt{Spend}$ queries and $m_{\texttt{upk}}$ the number of divisible coins that he has withdrawn. This means that the total amount available by a user upk is $m_{\texttt{upk}} \cdot N$. The number of coins withdrawn by all users during an experiment is denoted by m.

In the anonymity security game, we differ a little bit from [CPST15a]: while $c_{\texttt{upk}_b}$ is increased by V at step 6 during the Spend protocol, $c_{\texttt{upk}_{1-b}}$ is also increased by V at step 7 to avoid $\mathcal{A}$ trivially wins by trying to make one of the two players to overspend money.

Let $\mathcal{A}$ be a probabilistic polynomial adversary. A divisible E-cash system is:

– *traceable* if $\texttt{Succ}^{tra}(\mathcal{A}) = \Pr[\texttt{Exp}^{tra}_{\mathcal{A}}(1^k, V) = 1]$ is negligible for any $\mathcal{A}$;
– *exculpable* if $\texttt{Succ}^{excu}(\mathcal{A}) = \Pr[\texttt{Exp}^{excu}_{\mathcal{A}}(1^k, V) = 1]$ is negligible for any $\mathcal{A}$;
– *anonymous* if $\texttt{Adv}^{anon}(\mathcal{A}) = \Pr[\texttt{Exp}^{anon-1}_{\mathcal{A}}(1^k, V)]$ - $\Pr[\texttt{Exp}^{anon-0}_{\mathcal{A}}(1^k, V)]$ is negligible for any $\mathcal{A}$.

4 Our construction

4.1 High Level Description

Our Approach. We start from [CPST15a,CPST15b], in order to keep the quite easy and efficient withdrawal procedure (which mostly consists in certifying secret scalars). But we would like to improve on the spending procedure, and namely to get everything really constant (both in time and in size). Indeed, the user should be able to send only one information revealing the serial numbers, corresponding to the amount to be spent. But he should also be able to choose the sequence he discloses. For example, if he wants to pay a value V with a coin whose $(j-1)$ first serial numbers have already been used, then he should be able to send an element $\phi_{V,j}$ revealing the V serial numbers $\texttt{SN}_j, \ldots, \texttt{SN}_{j+V-1}$.

Description. All the serial numbers have the same structure, and are just customized by a random secret scalar x which constitutes the secret of the coin (our withdrawals are thus similar to the ones of [CPST15a,CPST15b]). More specifically, the public parameters contain the N values $s_j = s^{y^j}$ (for $j = 1, \ldots, N$), with a public group element $s \in \mathbb{G}_1$, and some secret scalar $y \xleftarrow{\$} \mathbb{Z}_p$: for any coin's secret x, this defines the serial numbers $\texttt{SN}_j = e(s, \widetilde{g})^{x \cdot y^j}$.

The critical point is to find a way to construct the unique $\phi_{V,j}$ and to decide which elements should be provided in the public parameters pp to enable the bank to compute the serial numbers (all the expected ones, but not more).

First Attempt. One could define $\phi_{V,j}$ as s_j^x, in which case pp should contain the set $\mathcal{S} = \{\widetilde{g}_k = \widetilde{g}^{y^k}\}_{k=0}^{N-1}$. Indeed, a user with a fresh coin (*i.e.* never involved in a spending) must be able to spend a value N by revealing s_1^x and so the bank needs to know $\mathcal{S}$ to recover $\texttt{SN}_i \leftarrow e(s_1^x, \widetilde{g}_{i-1})$, for $i = 1, \ldots, N$. One can note that $\mathcal{S}$ is actually enough for any spending, since, for any $j \in [1, N]$, recovering $\texttt{SN}_j, \ldots, \texttt{SN}_{j+V-1}$ from $\phi_{V,j}$ still requires elements from $\{\widetilde{g}_k\}_{k=0}^{V-1}$.

However, there is an obvious problem with this solution. Once $\mathcal{S}$ is published, nothing prevents the bank from computing more serial numbers than the amount V of the transaction. For

example, if a user with a fresh coin spends a value 1, then the bank is still able to recover all the serial numbers from $\phi_{1,1} = s_1^x$.

Our Solution. It is therefore necessary to provide a way, for the user, to control the amount of serial numbers which can be recovered from the element s_j^x. To this end, we define N (one for each possible value $V \in [1, N]$) ElGamal [ElG84] public keys $h_V = g^{a_V}$ and add the sets $\mathcal{S}_V = \{\widetilde{g}_k^{-a_V}\}_{k=0}^{V-1}$, for $V = 1, \ldots, N$, to pp. To reveal V serial numbers from s_j^x, the user now encrypts it under h_V, which defines $\phi_{V,j}$ as $(c_0 = g^r, c_1 = s_j^x \cdot h_V^r)$, for some $r \in \mathbb{Z}_p$. By using the elements from $\mathcal{S}_V$, the bank is still able to compute the V serial numbers since:

$$\begin{aligned} e(c_1, \widetilde{g}_k) \cdot e(c_0, \widetilde{g}_k^{-a_V}) &= e(s_j^x \cdot h_V^r, \widetilde{g}_k) \cdot e(g^r, \widetilde{g}_k^{-a_V}) \\ &= e(s_j^x, \widetilde{g}_k) \cdot e(h_V^r, \widetilde{g}_k) \cdot e(g^r, \widetilde{g}_k^{-a_V}) \\ &= e(s^{y^j \cdot x}, \widetilde{g}^{y^k}) \cdot e(g^{a_V \cdot r}, \widetilde{g}_k) \cdot e(g^{-a_V \cdot r}, \widetilde{g}_k) \\ &= e(s, \widetilde{g})^{x \cdot y^{j+k}} = \mathtt{SN}_{j+k}, \end{aligned}$$

for $k = 0, \ldots, V-1$. But now, it can no longer derive additional serial numbers because $\mathcal{S}_V$ only contains V elements. Moreover, the elements of the other sets $\mathcal{S}_{V'}$, for $V' \neq V$, are useless since they correspond to other public keys.

One can note that ElGamal encryption was also used in [CPST15b] but to prevent an adversary from testing relations across the different levels of the tree. We here use it to enable a total control on the amount of revealed serial numbers. A same element s_j^x can thus be involved in spendings of different values, which is the basis of the efficiency and the flexibility of our scheme.

Security Analysis. An interesting feature of our solution is that the bank does not need to know the index j to compute the serial numbers. This is due to the fact that $\mathtt{SN}_{j+1} = \mathtt{SN}_j^y$, for all $j \in [1, N-1]$ and so that the computation of a serial number is independent from j. Therefore, a spending does not reveal any additional information about the coin (such as the spent part) and so achieves the strongest notion of anonymity.

However, this has implications on the security analysis, since one must take into account the relations between the different serial numbers. Anonymity will then rely on a new assumption, called $N - \mathsf{MXDH}'$, which seems reasonable for Type-3 pairings, as we explain in Section 2.2.

Validity of a Transaction. Serial numbers are central to the detection of double-spending and so to ensure the traceability of the scheme. It is therefore necessary, during a spending of value V, to force the user to send a valid element $\phi_{V,j}$, by requesting a proof that the latter is well-formed. The user must then prove that (1) $\phi_{V,j}$ is an ElGamal encryption of some s_j^x under h_V (which is known since it corresponds to the spent amount), where (2) x has been certified, and (3) s_j is a valid parameter for a transaction of value V. The first two statements can easily be handled using the Groth-Sahai [GS08] methodology, but this is not the case for the third one. Indeed, as we explained, s_j (and so the index j) cannot be revealed unless breaking the anonymity of the scheme which would only achieve a weaker unlinkability property (as defined in [CPST15a]).

We could use the solution from [CPST15a] which consists in certifying each s_j under the public keys $\mathsf{pk}^1, \ldots, \mathsf{pk}^{N-j+1}$ and to prove that the s_j to be used is certified under the public key pk^V. However, such a solution is quite efficient for tree-based schemes where each s_j is associated with a unique node and so with a single amount, but not for our scheme where s_j can be involved in any transaction of value V such that $V \in [1, N-j+1]$. This would dramatically increase the bank's public key since it would contain about $N^2/2$ certificates.

While our public parameters will be of quadratic size, because of the sets $\mathcal{S}_V$, we hope the part necessary to the user to be at most linear in N. We will then use another solution which exploits the relation $e(s_j, \widetilde{g}_{V-1}) = e(s_{j+V-1}, \widetilde{g})$. To prove that $j \leq N - V + 1$, the user will thus simply prove that there is some s_k, for $k \in [1, N]$, such that $e(s_j, \widetilde{g}_{V-1}) = e(s_k, \widetilde{g})$. This can be

done efficiently if a certificate on each s_k is provided by the bank. One may note that this proof only ensures that $j \leq N - V + 1$ and not that $j \geq 1$. However, we will show, in the security analysis, that a user is unlikely to produce a proof for an element $s_j \notin \{s_1, \ldots, s_N\}$.

Security Tags. Detection of double-spending may not be sufficient to deter users from cheating. To prevent frauds it is also necessary to provide a way to identify dishonest users. Since we aim at achieving the anonymity property, such an identification cannot rely on some trusted entity with the power of tracing any user of the system. We will then use the standard technique of security tags which allows to recover the spender's identity from any pair of transactions detected as a double-spending. Similarly to the constructions of [CPST15a,CPST15b], we will add to the public parameters the elements t_j such that, $\forall j \in [1, N]$, $t_j = s_j^c$ for some $c \in \mathbb{Z}_p$ and define, for a transaction involving $\phi_{V,j}$, the security tag as $\psi_{V,j} = (g^{r'}, \mathsf{upk}^R \cdot t_j^x \cdot h_V^{r'})$ where upk is the user's public key and R is some public information related to the transaction. As we prove below, such a tag hides the identity of a spender as long as he does not double-spend its coin.

Remark 7. Divisible e-cash systems do not usually specify the way the coin should be spent. As explained above, our construction is the first one to allow sequential spendings, contrarily to tree-based construction where the coins may contain several holes (see Section 1.1). Therefore, for sake of simplicity, we assume in the following that the user sequentially reveals the serial numbers and so we associate each coin to an index j. The latter means that $\mathtt{SN}_1, \ldots, \mathtt{SN}_{j-1}$ have already been revealed and that the next spending of value V will reveal $\mathtt{SN}_j, \ldots, \mathtt{SN}_{j+V-1}$.

However, we stress that the user is free to spend the coin as he wants. The only constraint is that two spendings must not reveal the same serial numbers, otherwise the user will be accused of double-spending.

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 $\mathbb{G}_1$, $\widetilde{g}$ be a generator of $\mathbb{G}_2$, and H be collision-resistant hash function onto $\mathbb{Z}_p$. A trusted authority generates $(z, y) \xleftarrow{\$} \mathbb{Z}_p^2$ and, for $i = 1, \ldots, N$ (where N is the value of the coin), $a_i \xleftarrow{\$} \mathbb{Z}_p$. It then computes the public parameters as follows:

- $(s, t) \leftarrow (g^z, h^z)$;
- $(s_j, t_j) \leftarrow (s^{y^j}, t^{y^j})$, for $j = 1, \ldots, N$;
- $\widetilde{g}_k \leftarrow \widetilde{g}^{y^k}$, for $k = 0, \ldots, N-1$
- $h_i \leftarrow g^{a_i}$, for $i = 1, \ldots, N$;
- $\widetilde{h}_{i,k} \leftarrow \widetilde{g}^{-a_i \cdot y^k}$, for $i = 1, \ldots, N$ and $k = 0, \ldots, i-1$.

These parameters can also be cooperatively generated by a set of users and the bank, in a way similar to the one described in [CPST15a]. The point is that none of these entities should know the scalars $(a_i)_i$, y or z.

We divide the public parameters pp into two parts, $pp_{\mathcal{U}} \leftarrow \{g, h, u_1, u_2, w, H, \{h_i\}_{i=1}^N, \{(s_j, t_j)\}_{j=1}^N\}$ and $pp_{\mathcal{B}} \leftarrow \{\{\widetilde{g}_k\}_{k=0}^{N-1}, \{(\widetilde{h}_{i,k})_{k=0}^{i-1}\}_{i=1}^N\}$. The former contains the elements necessary to all the entities of the system whereas the latter contains the elements only useful to the bank during the `Deposit` protocol. We therefore assume that the users and the merchants only store $pp_{\mathcal{U}}$ and discard $pp_{\mathcal{B}}$. Note that the former is linear in N, while the latter is quadratic.

Our protocols make use of NIZK and NIWI proofs for multi-exponentiations and pairing-product equations which are covered by the Groth-Sahai proof system [GS08]. We then add to $pp_{\mathcal{U}}$ the description of a CRS for the perfect soundness setting and of a one-time signature scheme Σ_{ots} (*e.g.* the one from [BB08]).

5 Our Divisible E-Cash System

In this section, we provide an extended description of our new protocol and then discuss its efficiency. We describe a concrete instantiation in Appendix A.

5.1 The protocol

- $\texttt{Keygen}()$: 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.
- $\texttt{BKeygen}()$: The bank has two important roles to play. It must (1) deliver new coins to users during withdrawals and (2) control the transactions to detect double-spendings and identify the defrauders.
  The first point will require a signature scheme Σ_1 whose message space is $\mathbb{G}_1^2$ to certify the secret values associated with the withdrawn coins. We can therefore use the construction from [AGHO11] which is optimal in type-3 bilinear groups.
  The second point relies on the proof of validity of the elements $\phi_{V,j}$ sent during a transaction. As explained above, such a proof requires that the elements s_k are certified, for $k = 1, \ldots, N$. For the same reasons, their dual elements t_k must be certified too. It is therefore necessary to select a structure-preserving signature scheme Σ_0 whose message space is $\mathbb{G}_1^2$. We can then still choose the one from [AGHO11] but our security analysis shows that a scheme achieving a weaker security notion would be enough.
  Once the schemes Σ_0 and Σ_1 are selected, the bank generates $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow \Sigma_0.\texttt{Keygen}(pp)$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \Sigma_1.\texttt{Keygen}(pp)$. It then computes $\tau_j \leftarrow \Sigma_0.\texttt{Sign}(\mathsf{sk}_0, (s_j, t_j))$ for all $j \in 1, \ldots, N$ and sets $\mathsf{bsk} \leftarrow \mathsf{sk}_1$ and $\mathsf{bpk} \leftarrow \{\mathsf{pk}_0, \mathsf{pk}_1, \tau_1, \ldots, \tau_N\}$.
- $\texttt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: As explained in the previous section, each coin is associated with a random scalar x, which implicitly defines its serial numbers as $\mathtt{SN}_k = e(s_j^x, \widetilde{g}) = e(s, \widetilde{g})^{x \cdot y^k}$, for $k = 1, \ldots, N$. Delivering a new coin thus essentially consists in certifying this scalar x. However, for security reasons, it is necessary to bind the latter with the identity of its owner. Indeed, if this coin is double-spent, it must be possible to identify the user who has withdrawn it. This could be done by certifying the pair $(x, \mathsf{usk}) \in \mathbb{Z}_p^2$ (without revealing them), using for example the scheme from [CL04], but, in the standard model, the bank will rather certify the pair $(u_1^{\mathsf{usk}}, u_2^x) \in \mathbb{G}_1^2$. This is due to the fact that scalars cannot be efficiently extracted from Groth-Sahai proofs, contrarily to group elements in $\mathbb{G}_1$.
  In practice, the user computes u_1^{usk} and $u_2^{x_1}$ for some random $x_1 \xleftarrow{\$} \mathbb{Z}_p$ and sends them to the bank along with upk. He then proves knowledge of x_1 and usk in a zero-knowledge way (using, for example, the Schnorr's interactive protocol [Sch90]). If the bank accepts the proof, it generates a random $x_2 \xleftarrow{\$} \mathbb{Z}_p$, computes $u \xleftarrow{\$} u_2^{x_1} \cdot u_2^{x_2}$ and $\sigma \leftarrow \Sigma_1.\texttt{Sign}(\mathsf{sk}_1, (u_1^{\mathsf{usk}}, u))$ (unless u was used in a previous withdrawal) and returns σ and x_2 to the user. The latter then sets the coin's secret $x \leftarrow x_1 + x_2$ and coin state $C \leftarrow (x, \sigma, 1)$: the last element of C is the index of the next serial number to be used. Hence the remaining amount on the coin is $N + 1$ minus this index.
  Informally, the cooperative generation of the scalar x allows us to exclude (with overwhelming probability) false positives, *i.e.* a collision in the list L of serial numbers maintained by the bank which would not be due to an actual double-spending. We refer to Remark 8 for more details.
- $\texttt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, V), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, V))$: Let $C = (x, \sigma, j)$ be the coin the user wishes to spend. The latter selects two random scalars $(r_1, r_2) \xleftarrow{\$} \mathbb{Z}_p^2$ and computes $R \leftarrow H(info)$, $\phi_{V,j} \leftarrow (g^{r_1}, s_j^x \cdot h_V^{r_1})$ and $\psi_{V,j} \leftarrow (g^{r_2}, \mathsf{upk}^R \cdot t_j^x \cdot h_V^{r_2})$, where *info* is some information related to the transaction (such as the date, the amount, the merchant's public key,...).
  Now, he must prove that (1) his coin C is valid and (2) that the elements $\phi_{V,j}$ and $\psi_{V,j}$ are well-formed. The first point consists in proving knowledge of a valid signature σ on $(u_1^{\mathsf{usk}}, u_2^x)$, whereas the second point requires to prove knowledge of τ_{j+V-1} on (s_{j+V-1}, t_{j+V-1}). This can be efficiently done in the standard model by using the Groth-Sahai methodology [GS08]. Unfortunately, the resulting proofs can be re-randomized which enables a dishonest merchant to deposit several versions of the same transcript. To prevent such a randomization, the user generates a one-time signature key pair $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots})$ which will be used to sign the whole transcript. To ensure that only the spender can produce this signature, the public key pk_{ots} will be certified into $\mu \leftarrow w^{\frac{1}{\mathsf{usk} + H(\mathsf{pk}_{ots})}}$. One may note that these problems do not arise in

the ROM since the proofs would be simply converted into a (non-randomizable) signature of knowledge by using the Fiat-Shamir heuristic [FS87].
More formally, once the user has computed $\phi_{V,j}$, $\psi_{V,j}$ and μ, he computes Groth-Sahai commitments to $\mathsf{usk}, x, r_1, r_2, s_j, t_j, s_{j+V-1}, t_{j+V-1}, \tau_{j+V-1}, \sigma, \mu$, $U_1 = u_1^{\mathsf{usk}}$ and $U_2 = u_2^x$. He next provides:

1. a NIZK proof π that the committed values satisfy:

$$\begin{aligned} \phi_{V,j} = (g^{r_1}, s_j^x \cdot h_V^{r_1}) \quad &\wedge \quad \psi_{V,j} = (g^{r_2}, (g^R)^{\mathsf{usk}} \cdot t_j^x \cdot h_V^{r_2}) \\ \wedge \quad U_2 = u_2^x \quad \wedge \quad U_1 = u_1^{\mathsf{usk}} \quad &\wedge \quad \mu^{(\mathsf{usk}+H(\mathsf{pk}_{ots}))} = w \\ \wedge \quad e(s_j, \widetilde{g}_{V-1}) = e(s_{j+V-1}, \widetilde{g}) \quad &\wedge \quad e(t_j, \widetilde{g}_{V-1}) = e(t_{j+V-1}, \widetilde{g}) \end{aligned}$$

2. a NIWI proof π' that the committed values satisfy:

$$\begin{aligned} 1 = \Sigma_0.\mathtt{Verify}(\mathsf{pk}_0, (s_{j+V-1}, t_{j+V-1}), \tau_{j+V-1}) \\ \wedge \quad 1 = \Sigma_1.\mathtt{Verify}(\mathsf{pk}_1, (U_1, U_2), \sigma). \end{aligned}$$

Finally, he computes $\eta \leftarrow \Sigma_{ots}.\mathtt{Sign}(\mathsf{sk}_{ots}, H(R||\phi_{V,j}||\psi_{V,j}||\pi||\pi'))$ and sends it to $\mathcal{M}$ along with $\mathsf{pk}_{ots}, \phi_{V,j}, \psi_{V,j}, \pi$ and π'.
The merchant accepts if the proofs and the signatures are correct in which case he stores $(V, Z, \Pi) \leftarrow (V, (\phi_{V,j}, \psi_{V,j}), (\pi, \pi', \mathsf{pk}_{ots}, \eta))$ while the user updates its coin $C \leftarrow (x, \sigma, j+V)$.

– $\mathtt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (V, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: When a transcript is deposited by a merchant, the bank parses it as $(V, (\phi_{V,j}, \psi_{V,j}), (\pi, \pi', \mathsf{pk}_{ots}, \eta))$ and checks its validity (in the same way as the merchant did during the $\mathtt{Spend}$ protocol). $\mathcal{B}$ also verifies that it does not already exist in its database.
If everything is correct, $\mathcal{B}$ derives the serial numbers from $\phi_{V,j} = (\phi_{V,j}[1], \phi_{V,j}[2])$ by computing $\mathtt{SN}_k \leftarrow e(\phi_{V,j}[2], \widetilde{g}_k) \cdot e(\phi_{V,j}[1], \widetilde{h}_{V,k})$, for $k = 0, \ldots, V-1$. If none of these serial numbers is in L, the bank adds them to this list and stores the associated transcript. Else, there is at least one $\mathtt{SN}' \in L$ (associated with a transcript (V', Z', Π')) and one $k^* \in [0, V-1]$ such that $\mathtt{SN}' = \mathtt{SN}_{k^*}$. The bank then outputs the two transcripts (V, Z, Π) and (V', Z', Π') as a proof of a double-spending.

– $\mathtt{Identify}((V_1, Z_1, \Pi_1), (V_2, Z_2, \Pi_2), \mathsf{bpk})$: The first step before identifying a double-spender is to check the validity of both transcripts and that there is a collision between their serial numbers, *i.e.* there are $k_1 \in [0, V_1 - 1]$ and $k_2 \in [0, V_2 - 1]$ such that:

$$\begin{aligned} \mathtt{SN}_{k_1} &= e(\phi_{V_1,j_1}[2], \widetilde{g}_{k_1}) \cdot e(\phi_{V_1,j_1}[1], \widetilde{h}_{V_1,k_1}) \\ &= e(\phi_{V_2,j_2}[2], \widetilde{g}_{k_2}) \cdot e(\phi_{V_2,j_2}[1], \widetilde{h}_{V_2,k_2}) = \mathtt{SN}_{k_2} \end{aligned}$$

Let T_b be $e(\psi_{V_b,j_b}[2], \widetilde{g}_{k_b}) \cdot e(\psi_{V_b,j_b}[1], \widetilde{h}_{V_b,k_b})$, for $b \in \{1, 2\}$. The algorithm checks, for each registered public key upk_i, whether $T_1 \cdot T_2^{-1} = e(\mathsf{upk}_i, \widetilde{g}_{k_1}^{R_1} \cdot \widetilde{g}_{k_2}^{-R_2})$ until it gets a match. It then returns the corresponding key upk^* (or $\perp$ if the previous equality does not hold for any upk_i), allowing anyone to verify, without the linear cost in the number of users, that the identification is correct.

Remark 8. A collision in the list L means that two transcripts $(V_1, Z_1, \Pi_1) \neq (V_2, Z_2, \Pi_2)$ lead to a same serial number $\mathtt{SN}$. Let $Z_b = (\phi_{V_b,j_b}, \psi_{V_b,j_b})$, for $b \in \{1, 2\}$, the soundness of the NIZK proofs produced by the users during the spendings implies that:

$$\begin{aligned} e(\phi_{V_1,j_1}[2], \widetilde{g}_{k_1}) \cdot e(\phi_{V_1,j_1}[1], \widetilde{h}_{V_1,k_1}) &= e(\mathsf{s}_1, \widetilde{g}_{k_1})^{x_1} = \mathtt{SN} \\ &= e(\mathsf{s}_2, \widetilde{g}_{k_2})^{x_2} = e(\phi_{V_2,j_2}[2], \widetilde{g}_{k_2}) \cdot e(\phi_{V_2,j_2}[1], \widetilde{h}_{V_2,k_2}) \end{aligned}$$

for some $k_1 \in [0, V_1 - 1]$, $k_2 \in [0, V_2 - 1]$ and certified scalars x_1 and x_2, where the elements s_1 and s_2 verify, with $\ell_1, \ell_2 \in [1, N]$:

$$e(\mathsf{s}_1, \widetilde{g}_{V_1-1}) = e(s_{\ell_1}, \widetilde{g}) \text{ and } e(\mathsf{s}_2, \widetilde{g}_{V_2-1}) = e(s_{\ell_2}, \widetilde{g}).$$

Therefore, we have, for $b \in \{1,2\}$, $e(\mathsf{s_b}, \widetilde{g}) = e(s, \widetilde{g})^{y^{\ell_b - V_b + 1}}$, and so

$$\mathtt{SN} = e(s, \widetilde{g})^{x_1 \cdot y^{\ell_1 - V_1 + 1 + k_1}} = e(s, \widetilde{g})^{x_2 \cdot y^{\ell_2 - V_2 + 1 + k_2}}$$

A collision thus implies that $x_1 \cdot x_2^{-1} = y^{\ell_2 - \ell_1 + V_1 - V_2 + k_2 - k_1}$. Since x_1 and x_2 are randomly (and cooperatively) chosen, without knowledge of y, a collision for $x_1 \neq x_2$ will only occur with negligible probability. We can then assume that these scalars are equal and so that the collision in L is due to a double-spending.

Remark 9. The soundness of the proofs implies that the `Identify` algorithm will output, with overwhelming probability, an identity upk each time a collision is found in L. Indeed, let $(V_1, Z_1, \Pi_1), (V_2, Z_2, \Pi_2)$ be the two involved transcripts, and k_1, k_2 such that:

$$\begin{aligned}\mathtt{SN}_{k_1} &= e(\phi_{V_1,j_1}[2], \widetilde{g}_{k_1}) \cdot e(\phi_{V_1,j_1}[1], \widetilde{h}_{V_1,k_1}) \\ &= e(\phi_{V_2,j_2}[2], \widetilde{g}_{k_2}) \cdot e(\phi_{V_2,j_2}[1], \widetilde{h}_{V_2,k_2}) = \mathtt{SN}_{k_2}\end{aligned}$$

For $b \in \{1,2\}$, if Π_b is sound, then $(\phi_{V_b,j_b}[1], \phi_{V_b,j_b}[2]) = (g^{r_b}, s_{j_b}^{x_b} \cdot h_{V_b}^{r_b})$ for some $r_b \in \mathbb{Z}_p$ and so:

$$\mathtt{SN}_{k_1} = e(s_{j_1}^{x_1}, \widetilde{g}_{k_1}) = e(s_{j_2}^{x_2}, \widetilde{g}_{k_2}) = \mathtt{SN}_{k_2} \tag{1}$$

For the same reasons, $T_b = e(\psi_{V_b,j_b}[2], \widetilde{g}_{k_b}) \cdot e(\psi_{V_b,j_b}[1], \widetilde{h}_{V_b,k_b}) = e(\mathsf{upk}_b^{R_b} \cdot t_{j_b}^{x_b}, \widetilde{g}_{k_b})$, for $b \in \{1,2\}$.

As explained in the previous remark, the equality (1) is unlikely to hold for different scalars x_1 and x_2. We may then assume that $x_1 = x_2 = x$ and so that $\mathsf{upk}_1 = \mathsf{upk}_2 = \mathsf{upk}$ since the bank verifies, during a withdrawal, that the same scalar x (or equivalently the same public value $u = u_2^x$) is not used by two different users.

The relation (1) also implies that $e(t_{j_1}^x, \widetilde{g}_{k_1}) = e(t_{j_2}^x, \widetilde{g}_{k_2})$ and so that:

$$T_1 \cdot T_2^{-1} = e(\mathsf{upk}^{R_1}, \widetilde{g}_{k_1}) \cdot e(\mathsf{upk}^{R_2}, \widetilde{g}_{k_2})^{-1} = e(\mathsf{upk}, \widetilde{g}_{k_1}^{R_1} \cdot \widetilde{g}_{k_2}^{-R_2}).$$

The defrauder's identity upk will then be returned by the algorithm `Identify`, unless $\widetilde{g}_{k_1}^{R_1} \cdot \widetilde{g}_{k_2}^{-R_2} = 1_{\mathbb{G}_2}$. However, such an equality is very unlikely for distinct k_1 and k_2 (for the same reasons as the ones given in Remark 8) but also for $k_1 = k_2$ since it would imply that $R_1 = R_2$ and so a collision on the hash function H.

The security of our divisible E-Cash system is stated by the following theorems, whose proofs can be found in the next section.

Theorem 10. *In the standard model, our divisible E-Cash system is* ***traceable*** *under the $N-$*BDHI *assumption if Σ_0 is an EUF-SCMA signature scheme, Σ_1 is an EUF-CMA signature scheme, and H is a collision-resistant hash function.*

Theorem 11. *Let q be a bound on the number of $\mathcal{O}$`Spend` queries made by the adversary. In the standard model, our divisible E-Cash system achieves the* ***exculpability property*** *under the $q-$*SDH *assumption if Σ_{ots} is a SUF-OTS signature scheme, and H is a collision-resistant hash function.*

Theorem 12. *In the standard model, our divisible E-Cash system is* ***anonymous*** *under the* SXDH *and the $N-$*$\mathsf{MXDH'}$ *assumptions.*

Remark 13. A downside of our construction is that its anonymity relies on a quite complex assumption. This is due to the fact that most elements of the public parameters are related, which must be taken into account by the assumption. As we explain in Appendix C, we can rely on a more conventional assumption (while keeping the constant size property) by generating these parameters independently. Unfortunately, this has a strong impact on the efficiency of the protocol. Such a solution must then be considered as a tradeoff between efficiency and security assumption.

Schemes	**Martens [Mär15]**	**Canard *et al* [CPST15b]**	**Our work**
Parameters			
$pp_{\mathcal{U}} \cup \mathsf{bpk}$	$(N+2)\ \mathbb{G}_1 + N\ \mathbb{G}_2 + \mathsf{pk}$	$(4N+n+4)\ \mathbb{G}_1 + 2\ \mathsf{pk} + N\ \lvert\mathsf{Sign}\rvert$	$(3N+5)\ \mathbb{G}_1 + 2\ \mathsf{pk} + N\ \lvert\mathsf{Sign}\rvert$
$pp_{\mathcal{B}}$	-	$(4N-1)\ \mathbb{G}_2$	$(N^2+3N+2)/2\ \mathbb{G}_2$
`Withdraw` Protocol			
Computations	$\mathsf{ME}_{\mathbb{G}_1}(N) + \mathsf{Sign}$	$2\ \mathsf{E}_{\mathbb{G}_1} + \mathsf{Sign}$	$2\ \mathsf{E}_{\mathbb{G}_1} + \mathsf{Sign}$
Coin Size	$2N\ \mathbb{Z}_p + \mathbb{G}_1 + \lvert\mathsf{Sign}\rvert$	$2\ \mathbb{Z}_p + \lvert\mathsf{Sign}\rvert$	$2\ \mathbb{Z}_p + \lvert\mathsf{Sign}\rvert$
`Spend` Protocol			
Computations	$(1+2v)\ \mathsf{E}_{\mathbb{G}_1} + v\ \mathsf{ME}_{\mathbb{G}_1}(N-V) + v\ \mathsf{ME}_{\mathbb{G}_2}(V) + \mathsf{Sign} + \mathsf{NIZK}\{(2v+2)\ \mathsf{E}_{\mathbb{G}_1} + v\ \mathsf{P} + \mathsf{Sign}\}$	$(1+7v)\ \mathsf{E}_{\mathbb{G}_1} + \mathsf{Sign} + \mathsf{NIZK}\{(3+4v)\ \mathsf{E}_{\mathbb{G}_1} + 2v\ \mathsf{P} + (1+v)\ \mathsf{Sign}\}$	$8\ \mathsf{E}_{\mathbb{G}_1} + \mathsf{Sign} + \mathsf{NIZK}\{7\ \mathsf{E}_{\mathbb{G}_1} + 2\ \mathsf{P} + 2\ \mathsf{Sign}\}$
Communications	$2v\ \mathbb{G}_1 + \lvert\mathsf{Sign}\rvert + \lvert\mathsf{NIZK}\rvert$	$4v\ \mathbb{G}_1 + \lvert\mathsf{Sign}\rvert + \lvert\mathsf{NIZK}\rvert$	$4\ \mathbb{G}_1 + \lvert\mathsf{Sign}\rvert + \lvert\mathsf{NIZK}\rvert$
`Deposit` Protocol			
Computations	$2V\ \mathsf{E}_{\mathbb{G}_1}$	$2V\ \mathsf{P}$	$2V\ \mathsf{P}$
Communications	$V\ \mathtt{SN} + \lvert\mathtt{Spend}\rvert$	$V\ \mathtt{SN} + \lvert\mathtt{Spend}\rvert$	$V\ \mathtt{SN} + \lvert\mathtt{Spend}\rvert$

Fig. 3. Efficiency comparison between related works and our construction for coins of value N and `Spend` and `Deposit` of value V ($V \leq N$). The computation and communication complexities[2] are given from the user's point of view.

5.2 Efficiency

We compare in Figure 3, the efficiency of our construction with the state-of-the-art, and namely Martens [Mär15] (which improves the construction of [CG10]) and Canard *et al* [CPST15b]. One can note that our table differs from those provided in these papers. This is mostly due to the fact that they only describe the most favorable case, where the spent value V is a power of 2. However, in real life, such an event is quite unlikely. Most of the time, the users of such systems will then have to write $V = \sum b_i \cdot 2^i$, for $b_i \in \{0, 1\}$ and repeat the `Spend` protocol for each $b_i = 1$. Our description therefore considers the Hamming weight v of V (*i.e.* the number of b_i such that $b_i = 1$) but, for a proper comparison, also takes into account the possible optimisations of batch spendings (for example proving that the user's secret is certified can be done only once).

Another difference with [Mär15] comes from the fact that the author considered that "a multi-base exponentiation takes a similar time as a single-base exponentiation". Although some works (*e.g.* [BGR98]) have shown that an N-base exponentiation can be done more efficiently that N single-base exponentiations, considering that the cost of the former is equivalent to the one of a single exponentiation is a strong assumption, in particular when N can be greater than 1000 (if the coin's value is greater than 10$). Our table therefore distinguishes multi-base exponentiations from single ones.

An important feature for an electronic payment system is the efficiency of its `Spend` protocol. This is indeed the one subject to the strongest time constraints. For example, public transport services require that payments should be performed in less than 300ms [GSM12], to avoid congestion in front of turnstiles. From this perspective, our scheme is the most effective one and, above all, is the first one to achieve constant time (and size) spendings, no matter which value is spent. Moreover, our divisible E-Cash system offers the same efficiency as the withdrawals of [CPST15b], while keeping a reasonable size for the parameters $pp_{\mathcal{U}}$. Indeed, in our protocol, $pp_{\mathcal{U}}$ just requires 230 KBytes of storage space for $N = 1024$ (defining the coin's

[2] n denotes the smallest integer such that $N \leq 2^n$ and v the Hamming weight of V.
$\mathsf{E}_{\mathbb{G}}$ refers to an exponentiation in $\mathbb{G}$, $\mathsf{ME}_{\mathbb{G}}(m)$ to a multi-exponentiation with m different bases in $\mathbb{G}$, P to a pairing computation, and Sign to the cost of the signing protocol whose public key is pk.
$\mathsf{NIZK}\{\mathsf{E}_{\mathbb{G}}\}$ denotes the cost of a NIZK proof of a multi-exponentiation equation in $\mathbb{G}$, $\mathsf{NIZK}\{\mathsf{P}\}$ the one of a pairing-product equation, and $\mathsf{NIZK}\{\mathsf{Sign}\}$ the one of a valid signature.
Finally, `SN` refers to the size of a serial number and $\lvert\mathtt{Spend}\rvert$ to the size of the transcript of the `Spend` protocol.

value as 10.24$) if Barreto-Naehrig curves [BN06] are used to instantiate the bilinear groups. For the same settings, pp_U amounts to 263 KBytes for [CPST15b] and 98 KBytes for [Mär15].

From the bank's point of view, the downside of our scheme is the additional parameters pp_B that the bank must store, and they amount to 33 MBytes, but it should not be a problem for this entity. As for the other schemes, each deposit of a value V requires to store V serial numbers whose size can be adjusted by using an appropriate hash function (see Remark 14 below).

Remark 14. As explained in [CPST15a], the bank does not need to store the serial numbers but only their smaller hash values, as fingerprints. Therefore, the size of the V elements `SN` computed during a deposit of value V is the same for all the schemes. The `Deposit` size then mostly depends on the size of the `Spend` transcripts. By achieving smaller, constant-size spendings, we thus alleviate the storage burden of the bank and so improve the scalability of our divisible E-Cash system.

Remark 15. Public identification of defrauders has an impact on the complexity of the system. This roughly doubles the size of the parameters and requires several additional computations during a spending. Such a property also has consequences on the security analysis which must rely on a stronger assumption (namely the $N-\mathsf{MXDH}'$ one instead of its weaker variant) involving more challenge elements.

However, in some situations, it can be possible to consider an authority which would be trusted to revoke user's anonymity only in case of fraud. The resulting e-cash system, called *fair*, obviously weakens anonymity but may be a reasonable tradeoff between user's privacy and legal constraints.

Our scheme can be modified to add such an entity. One way would be to entrust it with the extraction key of the Groth-Sahai proof system. It could then extract the element $U_1 = u_1^{\mathsf{usk}}$ from any transaction and so identify the spender. The elements t_j would then become unnecessary and could be discarded from the public parameters. Moreover, the elements $\psi_{V,j}$, along with the associated proofs, would also become useless during the `Spend` protocol. The complexity of the scheme would then be significantly improved. The consequences of these changes on the security analysis are discussed in Remark 21 of the next section.

6 Security Analysis

6.1 Proof of Theorem 10: Traceability

Let us consider a successful adversary $\mathcal{A}$ which manages to spend more than he has withdrawn without being traced. This formally means that it is able to produce, after q_w withdrawals, u valid transcripts $\{(V_i, Z_i, \Pi_i)\}_{i=1}^{u}$ representing an amount of $\sum_{i=1}^{u} V_i > N \cdot q_w$, but such that $\texttt{Identify}((V_i, Z_i, \Pi_i), (V_j, Z_j, \Pi_j)) = \perp$, for all $i \neq j$. We can have the three following cases:

- Type-1 Forgeries: $\exists i$ such that Π_i contains commitments to a pair (s_{ℓ_i}, t_{ℓ_i}) which was not signed in a τ_ℓ by the bank, during the key generation phase;
- 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 $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$ query;
- Type-3 Forgeries: $\forall 1 \leq i \leq u$, $\exists \tau_{\ell_i}$ in bpk which is a valid signature on the pair (s_{ℓ_i}, t_{ℓ_i}) 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 $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$ query, but identification fails.

Intuitively, the first two cases imply an attack against the signatures schemes Σ_0 or Σ_1, respectively. This is formally stated by the two following lemmas:

Lemma 16. *Any Type-1 forger $\mathcal{A}$ with success probability ε can be converted into an adversary against the EUF-SCMA security of Σ_0 with the same success probability.*

Proof. The reduction $\mathcal{R}$ generates the public parameters (the group elements), and sends $\{(s_j, t_j)\}_{j=1}^{N}$ to the signing oracle of the EUF-SCMA security experiment which returns the signatures $\{\tau_j\}_{j=1}^{N}$

along with the challenge public key pk. It can run $\Sigma_1.\texttt{Keygen}$ to get the key pair $(\mathsf{sk}_1, \mathsf{pk}_1)$ and set bpk as $(\mathsf{pk}_0 = \mathsf{pk}, \mathsf{pk}_1, \tau_1, \ldots, \tau_N)$. One may note that $\mathcal{R}$ is able to answer any query from $\mathcal{A}$ since it knows $\mathsf{bsk} = \mathsf{sk}_1$.

At the end of the game, $\mathcal{R}$ extracts (it has generated the CRS of the Groth-Sahai proofs system and so knows the related extraction keys) from Π_i, for $i \in [1, u]$, a valid signature τ_{ℓ_i} on some pair (s_{ℓ_i}, t_{ℓ_i}) under the public key pk. Since $\mathcal{A}$ is a Type-1 forger with success probability ε, at least one of these pairs does not belong to the set $\{(s_j, t_j)\}_{j=1}^N$ and so is valid forgery which can be used to break the EUF-SCMA security of Σ_0, with probability ε. □

Lemma 17. *Any Type-2 forger $\mathcal{A}$ with success probability ε can be converted into an adversary against the EUF-CMA security of Σ_1 with the same success probability.*

Proof. The reduction $\mathcal{R}$ generates the public parameters (the group elements) and its public key as usual except that it sets pk_1 as pk, the challenge public key in the EUF-CMA security experiment. $\mathcal{R}$ 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 the signing oracle and forward 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 a commitment 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. The soundness of the proof implies that it also contains a commitment to an element σ such that $\Sigma_1.\texttt{Verify}((u_1^{\mathsf{usk}}, u_2^x), \sigma, \mathsf{pk}) = 1$. Such a forgery can then be used to break the EUF-CMA security of Σ_1. □

Now, it remains to evaluate the success probability of a Type-3 forger. The following lemma shows that it is negligible under $N - \mathsf{BDHI}$ assumption.

Lemma 18. *Any Type-3 forger $\mathcal{A}$ with success probability ε can be converted into an adversary against the $N - \mathsf{BDHI}$ assumption with the same success probability.*

Proof. Let $(\{g^{y^i}\}_{i=0}^N, \{\widetilde{g}^{y^i}\}_{i=0}^N) \in \mathbb{G}_1^{N+1} \times \mathbb{G}_2^{N+1}$ be a $N - \mathsf{BDHI}$ challenge. The reduction $\mathcal{R}$ generates random scalars $c, z' \leftarrow \mathbb{Z}_p$ and $a_i \leftarrow \mathbb{Z}_p$, for $i = 1 \ldots, N$, and sets the public parameters as follows:

- $(s_j, t_j) \leftarrow ((g^{y^{j-1}})^{z'}, (g^{y^{j-1}})^{c \cdot z'})$, for $j = 1, \ldots, N$;
- $\widetilde{g}_k \leftarrow \widetilde{g}^{y^k}$, for $k = 0, \ldots, N-1$
- $h_i \leftarrow g^{a_i}$, for $i = 1, \ldots, N$;
- $\widetilde{h}_{i,k} \leftarrow (\widetilde{g}^{y^k})^{-a_i}$, for $i = 1, \ldots, N$ and $k = 0, \ldots, i-1$.

By setting $(s, t) = (g^{z' \cdot y^{-1}}, g^{c \cdot z' \cdot y^{-1}})$ —recall that this pair is not published in pp—, one can easily check that the simulation is correct: $s_j = s^{y^j}$ and $t_j = t^{y^j}$. $\mathcal{R}$ then generates the CRS for the perfect soundness setting and stores the extraction keys. Finally, it computes the bank's key pair $(\mathsf{bsk}, \mathsf{bpk})$ as usual and so is able to answer every oracle queries.

At the end of the game, $\mathcal{R}$ extracts the elements $s^{(i)}$ committed in Π_i, for $i = 1, \ldots, u$. Each of these proofs also contains a commitment to signature τ_{ℓ_i} on the pair (s_{ℓ_i}, t_{ℓ_i}) such that: $e(s^{(i)}, \widetilde{g}_{V_i - 1}) = e(s_{\ell_i}, \widetilde{g})$. Since we here consider Type-3 forgeries, $\ell_i \in [1, N]$ (otherwise $\tau_{\ell_i} \notin \mathsf{bpk}$) and so $s_{\ell_i} = s^{y^{\ell_i}}$. Therefore, we have $s^{(i)} = s^{y^{\ell_i - V_i + 1}}$, where $\ell_i - V_i + 1 \leq N - V_i + 1$. We then distinguish the two following cases.

- Case 1: $\forall i \in [1, u]$, $\ell_i - V_i + 1 \geq 1$;
- Case 2: $\exists i \in [1, u]$ such that $\ell_i - V_i + 1 < 1$.

The first case means that $\mathcal{A}$ only used valid elements $s^{(i)}$ (*i.e.* $s^{(i)} = s_{j_i}$ such that $j_i \in [1, N - V_i + 1]$) to construct the proofs Π_i. So all the $(\sum_{i=1}^u V_i)$ serial numbers derived from the u transcripts returned by $\mathcal{A}$ belong to the set $\mathcal{S} = \{\cup_{k=1}^{q_w} \{e(s, \widetilde{g})^{x_k \cdot y^\ell}\}_{\ell=1}^N\}$, where $\{x_k\}_{k=1}^{q_w}$ is the list of the scalars certified by the bank during the $\mathcal{O}\texttt{Withdraw}_{\mathcal{U}}$ queries. An over-spending means that $\sum_{i=1}^u V_i > N \cdot q_w = |\mathcal{S}|$, there is at least one collision in the list of the serial numbers. However,

a collision without identification of a defrauder is unlikely, as we explained in Remark 9. Hence, case 1 can only occur with negligible probability.

Now, let us consider the second case: when such a case occurs, $\mathcal{R}$ is able to extract the element $s^{y^{\ell_i - V_i + 1}}$ such that $\ell_i - V_i + 1 \leq 0$, and compute $\mathsf{g} \leftarrow (s^{y^{\ell_i - V_i + 1}})^{1/z'} = g^{y^{\ell_i - V_i}}$ with $1 - N \leq \ell_i - V_i \leq -1$. Let k_i be the integer such that $\ell_i - V_i + k_i = -1$. The previous inequalities imply that $k_i \in [0, N-2]$ and so $\mathcal{R}$ can break the $N-\mathsf{BDHI}$ assumption by returning $e(g, \widetilde{g})^{y^{-1}} = e(\mathsf{g}, \widetilde{g}_{k_i})$. □

6.2 Proof of Theorem 11: Exculpability

The goal of the adversary $\mathcal{A}$ is to make the identify procedure to claim an honest user upk guilty of double-spending: it publishes two valid transcripts (V_1, Z_1, Π_1) and (V_2, Z_2, Π_2) such that $\mathsf{upk} = \mathtt{Identify}((V_1, Z_1, \Pi_1), (V_2, Z_2, \Pi_2))$, while this user did not perform the two transactions (maybe one). We can obviously assume that one of these transcripts has been forged by $\mathcal{A}$.

Let us consider a successful adversary. 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 the honest user to answer $\mathcal{O}\mathtt{Spend}$ queries.
- Type-2 forgeries: pk_{ots} was never used by this honest user.

Lemma 19. *Let q_s be a bound on the number of $\mathcal{O}\mathtt{Spend}$ queries. Any Type-1 forger $\mathcal{A}$ with success probability ε can be converted into an adversary against the SUF-OTS security of the one-time signature scheme Σ_{ots} with success probability greater than ε/q_s.*

Proof. The reduction $\mathcal{R}$ generates the public parameters along with the bank's key pair and selects an integer $i^* \in [1, q_s]$. Upon receiving the i^{th} $\mathcal{O}\mathtt{Spend}$ query, it acts normally if $i \neq i^*$, but uses the public key pk^*_{ots} and the signing oracle of the SUF-OTS security experiment if $i = i^*$.

Let pk_{ots} be the public key involved in the forged transcript. $\mathcal{R}$ aborts if $\mathsf{pk}_{ots} \neq \mathsf{pk}^*_{ots}$, which occurs with probability $1 - 1/q_s$. Else, the forged transcript contains a new one-time signature η under pk^*_{ots} which can be used against the security of Σ_{ots}. □

Lemma 20. *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 $\mathcal{A}$ with success probability ε can be converted into an adversary against the $q_s - \mathsf{SDH}$ assumption with success probability ε/q_a.*

Proof. Let $(g, g^{\alpha}, \ldots, g^{\alpha_{q_s}})$ be a $q_s - \mathsf{SDH}$ challenge, the reduction $\mathcal{R}$ 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 $(\mathsf{sk}^{(i)}_{ots}, \mathsf{pk}^{(i)}_{ots}) \leftarrow \Sigma_{ots}.\mathtt{Keygen}(1^k)$ and sets w as $g^{\prod_{i=1}^{q_s}(\alpha + H(\mathsf{pk}^{(i)}_{ots}))}$ (which is possible using the $q_s - \mathsf{SDH}$ challenge [BB08], 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 sets $\mathsf{upk}^* \leftarrow g^{\alpha}$ 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.
- $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, V)$ queries: $\mathcal{R}$ acts normally if $\mathsf{upk} \neq \mathsf{upk}^*$. Else, to answer the j^{th} query on upk^*, it computes $\mu \leftarrow g^{\prod_{i=1, i\neq j}^{q_s}(\alpha + H(\mathsf{pk}^{(i)}_{ots}))}$ which satisfies $\mu = w^{1/(\alpha + H(\mathsf{pk}^{(j)}_{ots}))}$, and uses $\mathsf{sk}^{(j)}_{ots}$ as in the $\mathtt{Spend}$ protocol.

The adversary then outputs two valid transcripts (V_1, Z_1, Π_1) and (V_2, Z_2, Π_2) which accuse upk of double-spending. If $\mathsf{upk} \neq \mathsf{upk}^*$ then $\mathcal{R}$ aborts which will occur with probability $1 - 1/q_a$. Else, the soundness of the proof implies that the forged transcript was signed under pk_{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, $\mathsf{pk}_{ots} \notin \{\mathsf{pk}_{ots}^{(i)}\}_i$. Therefore, $H(\mathsf{pk}_{ots}) \notin \{H(\mathsf{pk}_{ots}^{(i)})\}_i$ with overwhelming probability, due to the collision-resistance of the hash function H. The element μ can then be used to break the $q_s-\mathsf{SDH}$ assumption in $\mathbb{G}_1$ (as in [BB08]). □

6.3 Proof of Theorem 12: Anonymity

In this proof, we assume that the coins are spent in a sequential way: the index j in $C = (x, \sigma, j)$ is increased by V after each spending of an amount V, and the new j is used in the next spending. A next coin is used when the previous coin is finished. But the proof would also apply if the user could adaptively choose the coin (x, σ), as well as (j, V) for every spending.

We can make the proof with a sequence of games, starting from the initial game for anonymity, with a random bit b (see Figure 2), where the simulator emulates the challenger but correctly generating all the secret values. The advantage is ε, and we want to show it is negligible.

In a next game, the simulator makes a guess on the amount $V^* \in [1, N]$ chosen by the adversary during the step 3 of the anonymity experiment (see Figure 2) and also makes a guess $j^* \in [1, N - V^* + 1]$ for the actual index of the coin of the user upk_b at the challenge time (but this challenge value could be chosen by the adversary, as said above). In addition, we denote q_w the bound on the number of $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ queries, and the simulator selects a random integer $\ell^* \in [1, q_w]$, for the expected index of the $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ query that generates the coin that will be used in the challenge. If during the simulation it appears they are not correct, one stops the simulation. This guess does not affect the success probability of the adversary, when the guess is correct, but just reduces the advantage from ε to $2\varepsilon/(q_w N^2)$.

Next, the simulator generates the CRS for the Groth-Sahai proofs in the perfect witness indistinguishability setting, so that it can later simulate the proofs. This is indistinguishable from the previous game under the SXDH assumption.

Now, the simulator will simulate the public parameters from an $N - \mathsf{MXDH}'$ challenge:

- $(g^{\gamma^k}, h^{\gamma^k})_{k=0}^{P} \in \mathbb{G}_1^{2P+2}$,
- $(g^{\alpha\cdot\delta\cdot\gamma^{-k}}, h^{\alpha\cdot\delta\cdot\gamma^{-k}})_{k=0}^{E} \in \mathbb{G}_1^{2E+2}$,
- $(g^{\chi\cdot\gamma^k}, h^{\chi\cdot\gamma^k})_{k=D+1}^{P} \in \mathbb{G}_1^{2C}$,
- and $((g^{\alpha\cdot\gamma^{-k}})_{k=0}^{C}, (g^{\chi\cdot\gamma^k/\alpha}, h^{\chi\cdot\gamma^k/\alpha})_{k=0}^{C},) \in \mathbb{G}_1^{3S}$,
- as well as $(\widetilde{g}^{\gamma^k}, \widetilde{g}^{\alpha\cdot\gamma^{-k}})_{k=0}^{C} \in \mathbb{G}_2^{2S}$,
- and a pair $(g^{z_1}, h^{z_2}) \in \mathbb{G}_1^2$ be an $N - \mathsf{MXDH}'$ challenge.

We recall that $C = N^3 - N^2$, $S = C + 1$, $E = N^2 - N$, $D = S + E = N^3 - N + 1$ and $P = D + C = 2N^3 - N^2 - N + 1$. Let d be the quotient of the division of N^2 by V^* (*i.e.* $N^2 = d \cdot V^* + r_d$ with $0 \le r_d < V^*$), then the simulator constructs the public parameters as follows.

- g and h are defined from g^{γ^k} and h^{γ^k} respectively, with $k = 0$;
- $u_1 \xleftarrow{\$} \mathbb{G}_1$ and $u_2 \leftarrow g^{w\cdot\gamma^P}$, for a random $w \in \mathbb{Z}_p$;
- $\widetilde{g}$ is defined from $\widetilde{g}^{\gamma^k}$, with $k = 0$;
- $(s_j, t_j) \leftarrow (g^{\gamma^{D+d(1-V^*+j-j^*)}}, h^{\gamma^{D+d(1-V^*+j-j^*)}})$, for $j = 1, \ldots, N$;
- $\widetilde{g}_k \leftarrow \widetilde{g}^{\gamma^{d\cdot k}}$, for $k = 0, \ldots, N-1$;
- $h_i \leftarrow g^{w_i\cdot\alpha\cdot\gamma^{d(-i+1)}}$, for $i \in [1, \ldots, N]$, with w_i a random scalar;
- $\widetilde{h}_{i,k} \leftarrow \widetilde{g}^{-w_i\cdot\alpha\cdot\gamma^{d(k-i+1)}}$, for $i \in [1, \ldots, N]$ and $k = 0, \ldots, i-1$.

We must check that

(1) the simulation of the parameters is correct: let us define $y = \gamma^d$, $(s, t) = (g^{\gamma^{D+d(1-V^*-j^*)}}, h^{\gamma^{D+d(1-V^*-j^*)}})$, and $a_i = \alpha \cdot w_i \cdot \gamma^{d(-i+1)}$ for $i \in [1, \ldots, N]$. We then have:
   - $(s_j, t_j) = ((g^{\gamma^{D+d(1-V^*-j^*)}})^{\gamma^{d\cdot j}}, (h^{\gamma^{D+d(1-V^*-j^*)}})^{\gamma^{d\cdot j}}) = (s^{y^j}, t^{y^j})$;
   - $\widetilde{g}_k = \widetilde{g}^{y^k}$, for $k = 0, \ldots, N-1$;

– $h_i = g^{a_i}$, for $i = 1, \ldots, N$;
– $\widetilde{h}_{i,k} = \widetilde{g}^{-a_i \cdot y^k}$, for $i = 1, \ldots, N$ and $k = 0, \ldots, i-1$.

The simulation is therefore correct;

(2) all of these elements can be provided from the $N - \mathsf{MXDH}'$ challenge: First, recall that $N^2 = d \cdot V^* + r$ with $0 \leq r < V^* \leq N$. Then $2 \leq V^* + j^* \leq N + 1$ and $N \leq d \leq N^2$.
Let us consider the pairs $(s_j, t_j) = (g^{\gamma^{D+d(1-V^*+j-j^*)}}, h^{\gamma^{D+d(1-V^*+j-j^*)}})$, for $j = 1, \ldots, N$: $1 + j - (V^* + j^*) \geq 2 - (N+1) \geq -N + 1$, therefore, $d(1 - V^* + j - j^*) \geq -d(N-1) \geq -N^2(N-1) \geq -C$. Moreover, $d(1 - V^* + j - j^*) \leq d(N-1) \leq N^2(N-1) \leq C$. Hence $D - C \leq D + d(1 - V^* + j - j^*) \leq D + C = P$. Since $D = S + E = C + 1 + E$, $D - C = E + 1 = N^2 - N + 1 \geq 0$. Hence, the pairs (s_j, t_j) can be defined from the tuple $(g^{\gamma^k}, h^{\gamma^k})_{k=0}^{P}$ of the $N - \mathsf{MXDH}'$ instance.
About the elements $\widetilde{g}_k = \widetilde{g}^{\gamma^{d \cdot k}}$, since we have $0 \leq d \cdot k \leq N^2(N-1) = C$, for $k = 0, \ldots, N-1$, they all are in the tuple $(\widetilde{g}^{\gamma^k})_{k=0}^{C}$.
Eventually, let us consider the elements $h_i = g^{w_i \cdot \alpha \cdot \gamma^{d(-i+1)}}$ and $\widetilde{h}_{i,k} = \widetilde{g}^{-w_i \cdot \alpha \cdot \gamma^{d(k-i+1)}}$, for $i \in [1, N]$ and $k \in [0, i-1]$. Since $-C \leq -d(N-1) \leq d(-i+1) \leq 0$ and $-C \leq d(k-i+1) \leq 0$, they all can be computed from the tuples $(g^{\alpha \cdot \gamma^{-k}})_{k=0}^{C}$ and $(\widetilde{g}^{\alpha \cdot \gamma^{-k}})_{k=0}^{C}$, just using the additional random scalar w_i.
The reduction $\mathcal{R}$ is thus able to generate the public parameters from the $N - \mathsf{MXDH}'$ instance.

The simulator now has to answer all the oracle queries, with all the secret keys.

– $\mathcal{O}\mathtt{Add}()$ queries: run the $\mathtt{Keygen}$ algorithm and return upk (or mpk);
– $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$ queries: for the ℓ^{th} $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ query, the simulator plays normally if $\ell \neq \ell^*$, but sending the pair $(u_1^{\mathsf{usk}}, (g^{\chi \cdot \gamma^P})^w = u_2^{\chi})$ otherwise (using the $N - \mathsf{MXDH}'$ instance). It can then simulate the proof of knowledge and receives a scalar x' along with a signature σ on $(u_1^{\mathsf{usk}}, u_2^{x^*})$, where $x^* = \chi + x'$. The coin is then implicitly defined as $C^* = (x^*, \sigma, 1)$ and we will now denote its owner by upk^*;
– $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ queries: the simulator plays normally (if the guesses are correct, upk^* cannot be asked to be corrupted);
– $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: the simulator stores the public key which is now considered as registered;
– $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, V)$ queries: if $\mathsf{upk} \neq \mathsf{upk}^*$ or the coin to be used for the spending has not been withdrawn during the $\ell^* - \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$-query, then the simulator knows all the secret keys, and so it can play normally. Else, it proceeds as follows. One can first remark that if the guesses are correct, $j \notin [j^* - V + 1, j^* + V^* - 1]$. Otherwise this spending and the challenge spending would lead to a double-spending.
  - If $j \geq j^* + V^*$, then $D + d(1 - V^* + j - j^*) \geq D + d \geq D + 1$, so $s_j^{x^*}$ and $t_j^{x^*}$ can be computed from the tuple $(g^{\chi \cdot \gamma^k}, h^{\chi \cdot \gamma^k})_{k=D+1}^{P}$. Indeed,

$$s_j^{x^*} = (g^{\gamma^{D+d(1-V^*+j-j^*)}})^{x^*} = g^{\chi \cdot \gamma^{D+d(1-V^*+j-j^*)}} \cdot (g^{\gamma^{D+d(1-V^*+j-j^*)}})^{x'}$$
$$t_j^{x^*} = (h^{\gamma^{D+d(1-V^*+j-j^*)}})^{x^*} = h^{\chi \cdot \gamma^{D+d(1-V^*+j-j^*)}} \cdot (h^{\gamma^{D+d(1-V^*+j-j^*)}})^{x'}.$$

    The simulator can then send ElGamal encryptions of $s_j^{x^*}$ and $t_j^{x^*} \cdot g^{R \cdot \mathsf{usk}^*}$ under h_V (which yields valid ϕ_{V^*,j^*} and ψ_{V^*,j^*}) along with simulated proofs.
  - If $j \leq j^* - V$, then we proceed as follows.
    Let $r \leftarrow -\chi \cdot \gamma^{D+d(-V^*+1+j-j^*)+d(V-1)}/\alpha$ and $(r'_1, r'_2) \xleftarrow{\$} \mathbb{Z}_p^2$. Then, $(g^{r/w_V + r'_1}, s_j^{x'} \cdot h_V^{r'_1})$ and $(h^{r/w_V} \cdot g^{r'_2}, t_j^{x'} \cdot g^{R \cdot \mathsf{usk}^*} \cdot h_V^{r'_2})$ are valid pairs $\phi_{V,j}$ and $\psi_{V,j}$ which can be computed from the tuple $(g^{\chi \cdot \gamma^k/\alpha}, h^{\chi \cdot \gamma^k/\alpha})_{k=0}^{C}$ of the $N - \mathsf{MXDH}'$ instance: Since $d \cdot V^* = N^2 - r_d > N^2 - N$,

$$D + d(-V^* + 1 + j - j^*) + d(V-1) = D + d(V - V^* + j - j^*)$$
$$\leq D - d \cdot V^* < D - N^2 + N < D - E = S = C + 1$$

This is thus less or equal to C, as the indices of the tuple.
It then remains to prove that $(g^{r/w_V+r'_1}, s_j^{x'} \cdot h_V^{r'_1})$ and $(h^{r/w_V} \cdot g^{r'_2}, t_j^{x'} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_V^{r'_2})$ are valid ElGamal encryptions of $s_j^{x^*}$ and $t_j^{x^*} \cdot g^{R\cdot \mathsf{usk}^*}$ under h_V. Let c be the secret scalar such that $h = g^c$, $r_1 = r/w_V + r'_1$ and $r_2 = c \cdot r/w_V + r'_2$, we then have: $g^{r_1} = g^{r/w_V+r'_1}$ and

$$\begin{aligned} s_j^{x^*} \cdot h_V^{r_1} &= s_j^{\chi} \cdot s_j^{x'} \cdot h_V^{r/w_V+r'_1} = s_j^{\chi} \cdot s_j^{x'} \cdot h_V^{r/w_V+r'_1} \\ &= g^{\chi\cdot\gamma^{D+d(1-V^*+j-j^*)}} \cdot (g^{w_V\cdot\alpha\cdot\gamma^{d(-V+1)}})^{r/w_V} \cdot s_j^{x'} \cdot h_V^{r'_1} \\ &= g^{\chi\cdot\gamma^{D+d(1-V^*+j-j^*)}} \cdot g^{-\chi\cdot\gamma^{D+d(1-V^*+j-j^*)}} \cdot s_j^{x'} \cdot h_V^{r'_1}) \\ &= s_j^{x'} \cdot h_V^{r'_1} = s_j^{x'} \cdot h_V^{r'_1} \end{aligned}$$

Similarly, $g^{r_2} = h^{r/w_W} \cdot g^{r'_2}$ and as just above

$$\begin{aligned} t_j^{x^*} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_V^{r_2} &= t_j^{x'} \cdot t_j^{\chi} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_V^{c\cdot r/w_V} \cdot h_V^{r'_2} \\ &= t_j^{x'} \cdot h^{\chi\cdot\gamma^{D+d(1-V^*+j-j^*)}} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h^{-\chi\cdot\gamma^{D+d(1-V^*+j-j^*)}} \cdot h_V^{r'_2} \\ &= t_j^{x'} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_V^{r'_2} \end{aligned}$$

The spending is thus correctly simulated since r'_1 and r'_2 are random scalars.

During the challenge phase (*i.e.* the step 3 of the anonymity experiment), $\mathcal{A}$ outputs two public keys upk_0 and upk_1 along a value V. If the guesses were correct, $V = V^*$, $\mathsf{upk}^* = \mathsf{upk}_b$ and the coin involving x^* is spent, at index $j = j^*$. The simulator selects random r'_1 and r'_2, computes $R \leftarrow H(info)$, and returns, along with the simulated proofs, the pairs

$$\begin{aligned} \phi_{V^*,j^*} &= ((g^{z_1})^{-1/w_{V^*}} \cdot g^{r'_1}, s_j^{x'} \cdot g^{-\delta\cdot\alpha\cdot\gamma^{-d(V^*-1)}} \cdot h_{V^*}^{r'_1}) \\ \psi_{V^*,j^*} &= ((h^{z_2})^{-1/w_{V^*}} \cdot g^{r'_2}, t_j^{x'} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h^{-\delta\cdot\alpha\cdot\gamma^{-d(V^*-1)}} \cdot h_{V^*}^{r'_2}). \end{aligned}$$

One can note that $-d(V^*-1) \geq -N^2 + N = -E$ and so that the pair $(g^{\delta\cdot\alpha\cdot\gamma^{-d(V^*-1)}}, h^{\delta\cdot\alpha\cdot\gamma^{-d(V^*-1)}})$ belongs to the tuple $(g^{\alpha\cdot\delta\cdot\gamma^{-k}}, h^{\alpha\cdot\delta\cdot\gamma^{-k}})_{k=0}^{E}$.
Let $r_1 = -z_1/w_{V^*} + r'_1$ and $r_2 = -(c \cdot z_2)/w_{V^*} + r'_2$. If $z_1 = z_2 = \delta + \chi \cdot \gamma^D/\alpha$, then

$$\begin{aligned} (g^{r_1}, s_{j^*}^{x^*} \cdot h_{V^*}^{r_1}) &= (g^{r_1}, s_{j^*}^{\chi} \cdot s_{j^*}^{x'} \cdot h_{V^*}^{-z_1/w_{V^*}} \cdot h_{V^*}^{r'_1}) \\ &= (g^{r_1}, s_{j^*}^{\chi} \cdot s_{j^*}^{x'} \cdot g^{-\chi\cdot\gamma^{D+d(1-V^*)}} \cdot g^{\delta\cdot\alpha\gamma^{d(1-V^*)}} \cdot h_{V^*}^{r'_1}) \\ &= (g^{-z_1/w_{V^*}+r'_1}, s_{j^*}^{x'} \cdot g^{\delta\cdot\alpha\gamma^{d(1-V^*)}} \cdot h_{V^*}^{r'_1}) = \phi_{V^*,j^*} \end{aligned}$$

and

$$\begin{aligned} (g^{r_2}, t_{j^*}^{x^*} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_{V^*}^{r_2}) &= (g^{r_2}, t_{j^*}^{\chi} \cdot t_{j^*}^{x'} \cdot h_{V^*}^{-(c\cdot z_2)/w_{V^*}} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_{V^*}^{r'_2}) \\ &= (g^{r_2}, t_{j^*}^{\chi} \cdot t_{j^*}^{x'} \cdot h^{-\chi\cdot\gamma^{D+d(1-V^*)}} \cdot h^{\delta\cdot\alpha\gamma^{d(1-V^*)}} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_{V^*}^{r'_2}) \\ &= (h^{-z_2/w_{V^*}+r'_2}, t_{j^*}^{x'} \cdot h^{\delta\cdot\alpha\gamma^{d(1-V^*)}} \cdot g^{R\cdot \mathsf{usk}^*} \cdot h_{V^*}^{r'_2}) = \psi_{V^*,j^*} \end{aligned}$$

The challenge spending is thus correctly simulated too.

In the next game, we replace the $N - \mathsf{MXDH}'$ instance by a random instance, with random z_1 and z_2. From the simulation of ϕ_{V^*,j^*} and ψ_{V^*,j^*}, we see that they perfectly hide upk^*. Hence, the advantage of the adversary in this last game is exactly zero.

Remark 21. One can note that the h-based elements $h^{z_2}, \{h^{\gamma^k}\}_{k=0}^{P}, \{h^{\alpha\cdot\delta\cdot\gamma^{-k}}\}_{k=0}^{E}, \{h^{\chi\cdot\gamma^k}\}_{k=D+1}^{P}$ and $\{h^{\chi\cdot\gamma^k/\alpha}\}_{k=0}^{C}$ provided in the $N - \mathsf{MXDH}'$ challenge are only useful to simulate the security tags $\psi_{V,j}$ and ψ_{V^*,j^*}. In the case of fair divisible E-Cash system, they would no longer be necessary (see Remark 15) and so the security of the resulting scheme could simply rely on the weaker $N - \mathsf{MXDH}$ assumption.

7 Conclusion

We have proposed the first divisible e-cash system which achieves constant-time spendings, regardless of the spent value. Moreover, our solution keeps the best features of state-of-the-art, such as the efficiency of the withdrawals from [CPST15a] and the scalability of [CPST15b]. We argue that this is a major step towards the practical use of an e-cash system.

This also shows that the binary-tree structure, used by previous constructions, can be avoided. It may therefore open up new possibilities and incite new work in this area. We provide another construction in Appendix C whose security proof relies on a more classical assumption, still avoiding the tree structure, but with larger public parameters.

Acknowledgments

This work was supported in part by the French ANR Project ANR-12-INSE-0014 SIMPATIC 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

[AGHO11] 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.

[ASM08] 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.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004.

[BB08] 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.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 236–250. Springer, May / June 1998.

[BN06] 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 2006.

[CG07] 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.

[CG10] 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.

[Cha82] 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.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, August 2004.

[CP93] 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 1993.

[CPST15a] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible E-cash made practical. In *PKC 2015*, LNCS, pages 77–100. Springer, 2015.

[CPST15b] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Scalable divisible e-cash. In Tal Malkin, editor, *Conference on Applied Cryptography and Network Security (ACNS '15)*, Lecture Notes in Computer Science, New York, USA, 2015. Springer. To appear. Full version available on `http://eprint.iacr.org/2015/300`.

[Duc10] Léo Ducas. Anonymity from asymmetry: New constructions for anonymous HIBE. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 148–164. Springer, March 2010.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, August 1987.

[GMR88] 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.
[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
[GS08] 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.
[GSM12] GSMA. White paper: Mobile nfc in transport. `http://www.gsma.com/digitalcommerce/wp-content/uploads/2012/10/Transport_White_Paper_April13_amended.pdf`, 2012.
[Mär15] Patrick Märtens. Practical divisible e-cash. *IACR Cryptology ePrint Archive*, 2015:318, 2015.
[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, August 1992.
[Sch90] 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 1990.

A Implementation

We describe in this section an implementation of the divisible e-cash system of Section 5. The $N-\mathsf{MXDH}'$ assumption underlying the anonymity of our construction requires the use of type-3 pairings. We will therefore instantiate both Σ_0 and Σ_1 with the AGHO structure preserving signature scheme from [AGHO11] which has been proved optimal for this setting. For sake of clarity we recall below this scheme in the case where the message space is $\mathbb{G}_1^2$.

For all zero-knowledge proofs, we will describe the relations to be proven and underline the variables that are private to the prover.

Optimal Structure-Preserving Signatures for Messages in $\mathbb{G}_1^2$. We recall the construction from [AGHO11]:

- $\mathtt{Keygen}$: Set $\mathsf{sk} = (\alpha, \beta, \gamma_1, \gamma_2) \xleftarrow{\$} \mathbb{Z}_p^4$ and $\mathsf{pk} = (\widetilde{v}_1, \widetilde{v}_2, \widetilde{w}_1, \widetilde{w}_2) \leftarrow (\widetilde{g}^{\alpha}, \widetilde{g}^{\beta}, \widetilde{g}^{\gamma_1}, \widetilde{g}^{\gamma_2})$
- $\mathtt{Sign}(\mathsf{sk}, (m_1, m_2))$: To sign a pair of messages $(m_1, m_2) \in \mathbb{G}_1^2$, select a random $r \xleftarrow{\$} \mathbb{Z}_p$ and return $\sigma = (z_1, z_2, \widetilde{z}) \leftarrow (g^r, g^{\beta - r\alpha} \cdot m_1^{-\gamma_1} \cdot m_2^{-\gamma_2}, \widetilde{g}^{\frac{1}{r}})$
- $\mathtt{Verify}(\mathsf{pk}, (m_1, m_2), \sigma)$: Accept if $e(z_1, \widetilde{v}_1) \cdot e(z_2, \widetilde{g}) \cdot e(m_1, \widetilde{w}_1) \cdot e(m_2, \widetilde{w}_2) = e(g, \widetilde{v}_2)$ and $e(z_1, \widetilde{z}) = e(g, \widetilde{g})$.

Instantiation

- $\mathtt{Setup}$: The public parameters $pp_{\mathcal{U}} \leftarrow \{g, h, u_1, u_2, w, \{h_i\}_{i=1}^N, \{(s_j, t_j)\}_{j=1}^N, H\}$ and $pp_{\mathcal{B}} \leftarrow \{\{\widetilde{g}_k\}_{k=0}^{N-1}, \{\widetilde{h}_{i,0}, \ldots, \widetilde{h}_{i,i-1}\}_{i=1}^N\}$ are generated as described in Section 4.2. A common reference string $CRS \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$ for the Groth-Sahai proofs system [GS08] is then added to $pp_{\mathcal{U}}$. The signature scheme Σ_{ots} will be instantiated by the weakly secure signature scheme from [BB08].
- $\mathtt{Keygen}()$: Each user (resp. merchant) selects a random $\mathsf{usk} \leftarrow \mathbb{Z}_p$ (resp. msk) and gets $\mathsf{upk} \leftarrow g^{\mathsf{usk}}$ (resp. $\mathsf{mpk} \leftarrow g^{\mathsf{msk}}$).
- $\mathtt{BKeygen}()$: The bank runs twice the AGHO $\mathtt{Keygen}$ algorithm to generate $(\mathsf{sk}_0, \mathsf{pk}_0) \leftarrow ((\alpha^{(0)}, \beta^{(0)}, \gamma_1^{(0)}, \gamma_2^{(0)}), (\widetilde{v}_1^{(0)}, \widetilde{v}_2^{(0)}, \widetilde{w}_1^{(0)}, \widetilde{w}_2^{(0)}))$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow ((\alpha^{(1)}, \beta^{(1)}, \gamma_1^{(1)}, \gamma_2^{(1)}), (\widetilde{v}_1^{(1)}, \widetilde{v}_2^{(1)}, \widetilde{w}_1^{(1)}, \widetilde{w}_2^{(1)}))$. It then computes $\tau_j = (z_1^{(j)}, z_2^{(j)}, \widetilde{z}^{(j)}) \leftarrow \mathtt{Sign}(\mathsf{sk}_0, (s_j, t_j))$, for $j = 1, \ldots, N$, and sets $bsk \leftarrow \mathsf{sk}_1$ and $\mathsf{bpk} \leftarrow \{\mathsf{pk}_0, \mathsf{pk}_1, \tau_1, \ldots, \tau_N\}$.
- $\mathtt{Withdraw}(\mathcal{B}(\mathsf{bsk}, \mathsf{upk}), \mathcal{U}(\mathsf{usk}, \mathsf{bpk}))$: The user selects a random $x_1 \xleftarrow{\$} \mathbb{Z}_p$, computes $U_1 \leftarrow u_1^{\mathsf{usk}}$ and $U_2' \leftarrow u_2^{x_1}$, sends them to the bank, and proves, using the Schnorr's interactive protocol [Sch90], that

$$\mathsf{upk} = g^{\underline{\mathsf{usk}}} \ \wedge \ U_1 = u_1^{\underline{\mathsf{usk}}} \ \wedge \ U_2' = u_2^{\underline{x_1}}.$$

  If the proof is valid, the bank selects a random $x_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $U_2 \leftarrow U_2' \cdot u_2^{x_2}$. It then generates $\sigma = (z_1, z_2, \widetilde{z})$ and sends it, along with x_2 to the user who sets $C \leftarrow (x_1 + x_2, \sigma, 1)$.

– $\mathsf{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, V), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, V))$: To spend a value V, the user selects two random scalars $(r_1, r_2) \leftarrow \mathbb{Z}_p^2$, parses C as (x, σ, j) and computes $R \leftarrow H(info)$, $\phi_{V,j} \leftarrow (g^{r_1}, s_j^x \cdot h_V^{r_1})$ and $\psi_{V,j} \leftarrow (g^{r_2}, \mathsf{upk}^R \cdot t_j^x \cdot h_V^{r_2})$. He then generates $\mathsf{sk}_{ots} \xleftarrow{\$} \mathbb{Z}_p$, sets pk_{ots} as $\widetilde{g}^{\mathsf{sk}_{ots}}$ and computes $\mu \leftarrow w^{\frac{1}{\mathsf{usk}+H(\mathsf{pk}_{ots})}}$. He parses τ_{j+V-1} as $(z_1^{(j+V-1)}, z_2^{(j+V-1)}, \widetilde{z}^{(j+V-1)})$ and σ as $(z_1, z_2, \widetilde{z})$ and computes Groth-Sahai commitments to usk, x, r_1 and r_2 (2 elements of $\mathbb{G}_2$ each), to $U_1, U_2, s_j, t_j, s_{j+V-1}, t_{j+V-1}, \mu, z_1^{(j+V-1)}, z_2^{(j+V-1)}, z_1$ and z_2 (2 elements of $\mathbb{G}_1$ each) and to $\widetilde{z}^{(j+V-1)}$ and $\widetilde{z}$ (2 elements of $\mathbb{G}_2$ each). He next provides a NIZK proof π that

$$\begin{aligned}
&\phi_{V,j}[1] = g^{\underline{r_1}} \quad \wedge \quad \psi_{V,j}[1] = g^{\underline{r_2}} \\
\wedge \quad &\phi_{V,j}[2] = \underline{s_j}^{\underline{x}} \cdot h_V^{\underline{r_1}} \quad \wedge \quad \psi_{V,j}[2] = (g^R)^{\underline{\mathsf{usk}}} \cdot \underline{t_j}^{\underline{x}} \cdot h_V^{\underline{r_2}} \\
\wedge \quad &\underline{U_2} = u_2^{\underline{x}} \quad \wedge \quad \underline{U_1} = u_1^{\underline{\mathsf{usk}}} \quad \wedge \quad \underline{\mu}^{(\underline{\mathsf{usk}}+H(\mathsf{pk}_{ots}))} = w \\
\wedge \quad &e(\underline{d_j}, \widetilde{g}_{V-1}) = e(\underline{s_{j+V-1}}, \widetilde{g}) \quad \wedge \quad e(\underline{t_j}, \widetilde{g}_{V-1}) = e(\underline{t_{j+V-1}}, \widetilde{g})
\end{aligned}$$

The proof of the first two relations consists of 1 elements of $\mathbb{G}_1$ each. The next five equations add 10 elements of $\mathbb{G}_1$ and 20 elements of $\mathbb{G}_2$. The proof of the last two equations requires 2 elements of $\mathbb{G}_2$ each.

Finally, the user computes the NIWI proof π' that:

$$\begin{aligned}
&e(\underline{z_1}^{(j+V-1)}, \widetilde{v}_1^{(0)}) \cdot e(\underline{z_2}^{(j+V-1)}, \widetilde{g}) \cdot e(\underline{d_{(j+V-1)}}, \widetilde{w}_1^{(0)}) \cdot e(\underline{t_{(j+V-1)}}, \widetilde{w}_2^{(0)}) = e(g, \widetilde{v}_2^{(0)}) \\
&\wedge \quad e(\underline{z_1}^{(j+V-1)}, \underline{\widetilde{z}}^{(j+V-1)}) = e(g, \widetilde{g}) \\
&e(\underline{z_1}, \widetilde{v}_1^{(1)}) \cdot e(\underline{z_2}, \widetilde{g}) \cdot e(\underline{U_1}, \widetilde{w}_1^{(1)}) \cdot e(\underline{U_2}, \widetilde{w}_2^{(1)}) = e(g, \widetilde{v}_2^{(1)}) \\
&\wedge \quad e(\underline{z_1}, \underline{\widetilde{z}}) = e(g, \widetilde{g}).
\end{aligned}$$

The proof of the first and the third equation requires 2 elements of $\mathbb{G}_2$ each. The other relations add 4 elements of $\mathbb{G}_1$ and 4 elements of $\mathbb{G}_2$ each.

Finally, the user computes $\eta \leftarrow \Sigma_{ots}.\mathtt{Sign}(\mathsf{sk}_{ots}, H(R||\phi_j||\psi_j||\pi||\pi'))$ and sends it to $\mathcal{M}$ along with $\mathsf{pk}_{ots}, \phi_{V,j}, \psi_{V,j}, \pi$ and π'. The `Spend` transcript then contains 47 elements of $\mathbb{G}_1$ and 49 elements of $\mathbb{G}_2$.

We refer to Section 5 for a description of the `Deposit` and `Identify` algorithms since they do not depend on the instantiations of the signatures schemes.

B Proof of Theorem 5

Let us consider a $N - \mathsf{MXDH}$ challenge in a Type-3 setting. Since $g^z \in \mathbb{G}_1$, it can only be combined with elements of $\mathbb{G}_2$ during a pairing computation. In the generic bilinear group model, the latter can only be combinations of elements from the sets $(\widetilde{g}^{\gamma^k})_{k=0}^P$ and $(\widetilde{g}^{\alpha \cdot \gamma^{-k}})_{k=0}^C$. So, the assumption holds if $e(g^z, \prod_{k=0}^P \widetilde{g}^{a_{1,k}^* \cdot \gamma^k} \cdot \prod_{k=0}^C \widetilde{g}^{a_{2,k}^* \cdot \alpha \cdot \gamma^{-k}})$ is itself indistinguishable from a random element of $\mathbb{G}_T$, for any known scalars $a_{1,k}^*$ and $a_{2,k}^*$.

In the following, we associate group elements with polynomials whose formal variables are the unknown scalars involved in the $N - \mathsf{MXDH}$ challenge, namely α, γ and δ. We first prove that an adversary is unable to symbolically produce a *valid* element $e(g^{\delta + \chi \cdot \gamma^D/\alpha}, \prod_{k=0}^P \widetilde{g}^{a_{1,k}^* \cdot \gamma^k} \cdot \prod_{k=0}^C \widetilde{g}^{a_{2,k}^* \cdot \alpha \cdot \gamma^{-k}})$ (*i.e.* one which can be used to distinguish z) and then show that an accidental validity is quite unlikely.

In the generic bilinear group model, any element $v \in \mathbb{G}_1$ has been built through queries to the oracle of internal law in $\mathbb{G}_1$. We therefore know $a_{1,k}$, $a_{2,k}$, $a_{3,k}$, $a_{4,k}$ and $a_{5,k}$ such that:

$$v = \prod_{k=0}^{P} g^{a_{1,k} \cdot \gamma^k} \prod_{k=0}^{C} g^{a_{2,k} \cdot \alpha \cdot \gamma^{-k}} \prod_{k=0}^{E} g^{a_{3,k} \cdot \alpha \cdot \delta \cdot \gamma^{-k}} \prod_{k=D+1}^{P} g^{a_{4,k} \cdot \chi \cdot \gamma^k} \prod_{k=0}^{C} g^{a_{5,k} \cdot \chi \cdot \gamma^k / \alpha}$$

For the same reasons, any element T in $\mathbb{G}_T$ has been built through queries to the pairing oracle on $(v, \widetilde{v}) \in \mathbb{G}_1 \times \mathbb{G}_2$ and so can be written

$$T = e(v, \widetilde{v}) = e(g, \widetilde{g})^{P_1(\alpha,\gamma,\delta)\cdot P_2(\alpha,\gamma,\delta)}$$

where $P_1(\alpha, \gamma, \delta) = \sum_{k=0}^{P} a_{1,k} \cdot \gamma^k + \sum_{k=0}^{C} a_{2,k} \cdot \alpha \cdot \gamma^{-k} + \sum_{k=0}^{E} a_{3,k} \cdot \alpha \cdot \delta \cdot \gamma^{-k} + \sum_{k=D+1}^{P} a_{4,k} \cdot \chi \cdot \gamma^k + \sum_{k=0}^{C} a_{5,k} \cdot \chi \cdot \gamma^k/\alpha$ and $P_2(\alpha, \gamma, \delta) = \sum_{k=0}^{P} a'_{1,k} \cdot \gamma^k + \sum_{k=0}^{C} a'_{2,k} \cdot \alpha \cdot \gamma^{-k}$.

Let $\mathcal{S}$ be the set of polynomials P^* such that $e(g, \widetilde{g})^{P^*(\alpha,\gamma,\delta)}$:

$$P^*(\alpha, \gamma, \delta) = (\delta + \chi \cdot \gamma^D/\alpha) \cdot (\sum_{k=0}^{P} a^*_{1,k} \cdot \gamma^k + \sum_{k=0}^{C} a^*_{2,k} \cdot \alpha \cdot \gamma^{-k}).$$

Let us assume that $P_1 \cdot P_2 = P^* \in \mathcal{S}$ for some P_1 and P_2 described previously.

We first consider the monomials involving the variable δ: The ones contained in P_1 are $a_{3,k} \cdot \alpha \cdot \delta \cdot \gamma^{-k}$, for $k = 0, \ldots, E$. Since P_2 does not contain any term in $1/\alpha$, the monomials involving δ in $P_1 \cdot P_2$ will be $\delta \cdot \alpha \cdot P'$ for some $P' \in \mathbb{Z}_p[\alpha, \gamma]$. This implies that $a^*_{1,k} = 0$ for $k = 0, \ldots, P$. Indeed, if one were not zero then P^* would contain a term in $\delta \cdot \gamma^k$ which involves δ but not α, and so could not be equal to $P_1 \cdot P_2$. We may therefore consider that

$$P^* = (\delta + \chi \cdot \gamma^D/\alpha) \cdot \sum_{k=0}^{C} a^*_{2,k} \cdot \alpha \cdot \gamma^{-k} = \sum_{k=0}^{C} a^*_{2,k} \cdot \alpha \cdot \delta \cdot \gamma^{-k} + \sum_{k=0}^{C} a^*_{2,k} \cdot \chi \cdot \gamma^{D-k}$$

and so that there is at least one $k^* \in [0, C]$ such that $a_{2,k^*} \neq 0$. We then distinguish the two following cases.

- $k^* \in [0, E]$. P^* then contains a term in $\chi \cdot \gamma^{D-k^*}$ with $S \leq D - k^* \leq D$. So, let us consider the terms in $\chi \cdot \gamma^k$ contained in $P_1 \cdot P_2$. There are:
  (1) $a_{4,k_1} \cdot a'_{1,k_2} \cdot \chi \cdot \gamma^{k_1+k_2}$, for $k_1 \in [D+1, P]$ and $k_2 \in [0, P]$
  (2) $a_{5,k_1} \cdot a'_{2,k_2} \cdot \chi \cdot \gamma^{k_1-k_2}$, for $k_1 \in [0, C]$ and $k_2 \in [0, C]$
  However, the degree in γ of the monomials (1) is greater than $D + 1$ whereas the one of the monomials (2) is smaller than $C < S$. Therefore, it is impossible to construct groups elements $v \in \mathbb{G}_1$ and $\widetilde{v} \in \mathbb{G}_2$ whose associated polynomials P_1 and P_2 verify $P_1 \cdot P_2 \in \mathcal{S}$, and so to symbolically produce an element allowing to distinguish z when $k^* \in [0, E]$.
- $k^* \in [E + 1, C]$. P^* then contains a term in $\alpha \cdot \delta \cdot \gamma^{-k^*}$. However, the terms in $\alpha \cdot \delta \cdot \gamma^k$ involved in $P_1 \cdot P_2$ are $a_{3,k} \cdot a'_{1,k} \cdot \alpha \cdot \delta \cdot \gamma^k$, for $k \in [-E, P]$, and so differs from $\alpha \cdot \delta \cdot \gamma^{-k^*}$ (since $-k^* < -E$). Therefore, here again, one cannot symbolically construct an element allowing to distinguish z.

Now, let us evaluate the probability for an accidental validity, *i.e.* when two different polynomials involved in the answers returned by the oracle evaluate to the same values. One can note that the elements of $\mathbb{G}_1$ provided by the $N - \mathsf{MXDH}$ assumption are associated with polynomials of degree at most $P + 1$, whereas the ones in $\mathbb{G}_2$ are associated with polynomials of degree at most P. Therefore, the polynomials resulting from oracle queries are of degree at most $2P + 1 = 4N^3 - 2N^2 - 2N + 3$ due to the pairing computation.

Let q_G be the maximum number of oracle queries. Since the $N - \mathsf{MXDH}$ assumption contains $P + E + 5S + 1 = 7N^3 - 5N^2 - 2N + 8$ elements, there are at most $7N^3 - 5N^2 - 2N + 8 + q_G$ polynomials and so $(7N^3 - 5N^2 - 2N + 8 + q_G)^2/2$ pairs that could evaluate to the same value. By the Schwartz-Zippel lemma, such an event occurs with probability at most $(4N^3 - 2N^2 - 2N + 3) \cdot (7N^3 - 5N^2 - 2N + 8 + q_G)^2/2p \leq 2N^3 \cdot (7N^3 + q_G)^2/p$, which is negligible.

C Constant-Size Divisible E-Cash under Weaker Assumptions

C.1 High Level Description

The relations between the elements of the public parameters pp have a strong impact on the computational assumption underlying the anonymity of our scheme. The assumption must indeed take them into account while allowing the simulation in the reduction to insert a challenge for any amount V^* and any index j^*. The $N - \mathsf{MXDH}'$ assumption offers such a flexibility but at the cost of a rather high complexity.

One may wonder if it possible to rely on a simpler assumption while keeping the constant-time property. We prove below that this is the case, but the solution implies a significant increase of the size of the public parameters and a far more complex deposit protocol, which does not perfectly match with our initial goal.

Informally, the core idea of this new scheme is to randomly generate most of the public parameters. This offers much more flexibility to the reduction of the anonymity experiment which can therefore make use of a less complex assumption. More specifically, we define the public parameters as follows.

A trusted authority generates random $y_j \xleftarrow{\$} \mathbb{Z}_p$ and $r_{j,k} \xleftarrow{\$} \mathbb{Z}_p$, for $1 \leq j \leq N$ and $1 \leq k \leq N + 1 - j$. Then, it selects random $g, h, u_1, u_2, w \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$, and computes:

- $(s_{j,k}, t_{j,k}) \leftarrow (g^{r_{j,k}}, h^{r_{j,k}})$, for $1 \leq j \leq N$ and $1 \leq k \leq N + 1 - j$
- $\widetilde{g}_{(j,k)\mapsto i} \leftarrow \widetilde{g}^{y_i/r_{j,k}}$, for $1 \leq j \leq N$, $1 \leq k \leq N + 1 - j$ and $j \leq i \leq j + k - 1$

The user's part $pp_{\mathcal{U}}$ of the public parameters consists of the tuple $\{g, h, u_1, u_2, w, \widetilde{g}, \{(s_{j,k}, t_{j,k})\}_{j=1,k=1}^{j=N,k=N+1-j}\}$ while the bank's part $pp_{\mathcal{B}}$ consists of the tuple $\{\widetilde{g}_{(j,k)\mapsto i}\}_{j=1,k=1,i=j}^{j=N,k=N+1-j,i=j+k-1}$.

One can note that, for any $1 \leq j \leq N$, $1 \leq k \leq N - j + 1$ and $j \leq i \leq j + k - 1$, we have:

$$e(s_{j,k}, \widetilde{g}_{(j,k)\mapsto i}) = e(g^{r_{j,k}}, \widetilde{g}^{y_i/r_{j,k}}) = e(g, \widetilde{g})^{y_i}$$

and

$$e(t_{j,k}, \widetilde{g}_{(j,k)\mapsto i}) = e(h^{r_{j,k}}, \widetilde{g}^{y_i/r_{j,k}}) = e(h, \widetilde{g})^{y_i}.$$

We keep the same withdrawal procedure as in Section 5 and so associate each coin with a secret and certified scalar x. The latter implicitly defines $\mathtt{SN}_1, \ldots, \mathtt{SN}_N$ as $e(g, \widetilde{g})^{y_1 \cdot x}, \ldots, e(g, \widetilde{g})^{y_N \cdot x}$.

To spend a value V with a coin whose index is j, the user upk will send $\phi_{V,j} = s_{j,V}^x$ along with $\psi_{V,j} = t_{j,V}^x \cdot \mathsf{upk}^R$, where R is defined as in the previous construction. Correctness follows from the fact that $V \leq N + 1 - j$ (otherwise, the user would overspend its coin) and so $(s_{j,V}, t_{j,V}) \in pp_{\mathcal{U}}$. It is worthy to note that encryption is unnecessary here since the independence of the elements in pp makes the attack described in Section 4.1 impracticable.

During the `Deposit` protocol, the bank is able to recover the serial numbers $\mathtt{SN}_j, \ldots, \mathtt{SN}_{j+V-1}$ from $\phi_{V,j}$ by computing, for $i = j, \ldots, j + V - 1$:

$$e(\phi_{V,j}, \widetilde{g}_{(j,V)\mapsto i}) = e(s_{j,V}, \widetilde{g}_{(j,V)\mapsto i})^x = e(g, \widetilde{g})^{y_i \cdot x} = \mathtt{SN}_i.$$

However, this computation involves $\widetilde{g}_{(j,V)\mapsto i}$ and so requires the knowledge of the index j used during the spending. Unfortunately, revealing this information breaks the anonymity of the construction which will only achieve a weaker unlinkability notion, as explained in [CPST15a].

To achieve anonymity, it is therefore necessary to hide j and to find another way for computing the serial numbers. The solution proposed in [CPST15a] is to notice that the bank does not necessarily need to know the index j. Indeed, since it knows V (*i.e.* the spent value), it can repeat the previous procedure for any possible $\widetilde{g}_{(j',V)\mapsto i}$. More specifically, it can compute, for any $1 \leq j' \leq N + 1 - V$ and $j' \leq i \leq j' + V - 1$, $e(\phi_{V,j'}, \widetilde{g}_{(j',V)\mapsto i})$. Since $j \in [1, N + 1 - V]$, the bank is ensured to recover $\mathtt{SN}_j, \ldots, \mathtt{SN}_{j+V-1}$. Nevertheless, the obvious drawback of this solution is that the bank must perform many useless computations (namely, the ones such that $j' \neq j$) and then store theirs outputs, which are invalid serial numbers.

C.2 The protocol

We here provide an extended description of the protocol sketched above. The `Keygen` and `Withdraw` algorithms are the same as the ones presented in Section 5 and so are not recalled here.

- `BKeygen`(): As in the previous construction, the bank will have to certify, during a withdrawal, the secret values associated to the coin. It then generates a key pair $(\mathsf{sk}_1, \mathsf{pk}_1)$ for a structure preserving signature scheme Σ_1 whose message space is $\mathbb{G}_1^2$.
  The bank must also certify the pairs $(s_{j,k}, t_{j,k})$ so that the user can prove the validity of the elements $\phi_{V,j}$ and $\psi_{V,j}$ during a spending. More specifically, the user must prove that he used a valid pair $(s_{j,V}, t_{j,V})$ for the amount V. The bank then generates N key pairs $(\mathsf{sk}_0^{(k)}, \mathsf{pk}_0^{(k)})$ (one for each possible value $k \in [1, N]$) for a signature scheme Σ_0 whose message space is $\mathbb{G}_1^2$ and uses them to compute $\tau_{j,k} \leftarrow \Sigma_0.\mathtt{Sign}(\mathsf{sk}_0^{(k)}, (s_{j,k}, t_{j,k}))$, for $j = 1, \ldots, N$ and $k = 1, \ldots, N+1-j$. Therefore, $(s_{j,V}, t_{j,V})$ will be valid pair for a spending of value V if and only if it has been certified under $\mathsf{pk}_0^{(V)}$.
  The bank secret key bsk is then set as sk_1, while bpk is set as $\{\{\mathsf{pk}_0^{(k)}\}_{k=1}^{N}, \mathsf{pk}_1, \{\tau_{j,k}\}_{j=1,k=1}^{j=N,N+1-j}\}$.
- $\mathtt{Spend}(\mathcal{U}(\mathsf{usk}, C, \mathsf{bpk}, \mathsf{mpk}, V), \mathcal{M}(\mathsf{msk}, \mathsf{bpk}, V))$: Let $C = (x, \sigma, j)$ be the coin the user wishes to spend. The latter computes $R \leftarrow H(\mathit{info})$, $\phi_{V,j} \leftarrow s_{j,V}^x$ and $\psi_{V,j} \leftarrow t_{j,V}^x \cdot \mathsf{upk}^R$, where info is some information related to the transaction (such as the date, the amount, the merchant's public key,...).
  Next, he generates a key pair $(\mathsf{sk}_{ots}, \mathsf{pk}_{ots})$ for a one-time signature scheme Σ_{ots} and computes $\mu \leftarrow w^{\frac{1}{\mathsf{usk}+H(\mathsf{pk}_{ots})}}$.
  He must now prove the validity of $\phi_{V,j}$ and $\psi_{V,j}$ and so computes Groth-Sahai commitments to $\mathsf{usk}, x, s_{j,V}, t_{j,V}, \tau_{j,V}, \sigma, \mu$, $U_1 = u_1^{\mathsf{usk}}$ and $U_2 = u_2^x$. He then provides:
  1. a NIZK proof π that the committed values satisfy:

$$\phi_{V,j} = s_{j,V}^x \quad \wedge \quad \psi_{V,j} = t_{j,V}^x \cdot \mathsf{upk}^R \quad \wedge \quad w = \mu^{(\mathsf{usk}+H(\mathsf{pk}_{ots}))}$$
$$\wedge \quad U_1 = u_1^{\mathsf{usk}} \qquad \wedge \quad U_2 = u_2^x$$

  2. a NIWI proof π' that the committed values satisfy:

$$1 = \Sigma_0.\mathtt{Verify}(\mathsf{pk}_0^{(V)}, (s_{j,V}, t_{j,V}), \tau_{j,V})$$
$$\wedge \quad 1 = \Sigma_1.\mathtt{Verify}(\mathsf{pk}_1, (U_1, U_2), \sigma).$$

  Finally, he computes $\eta \leftarrow \Sigma_{ots}.\mathtt{Sign}(\mathsf{sk}_{ots}, H(R||\phi_{V,j}||\psi_{V,j}||\pi||\pi'))$ and sends it to $\mathcal{M}$ along with $\mathsf{pk}_{ots}, \phi_{V,j}, \psi_{V,j}, \pi$ and π'.
  The merchant accepts if the proofs and the signatures are correct in which case he stores $(V, Z, \Pi) \leftarrow (V, (\phi_{V,j}, \psi_{V,j}), (\pi, \pi', \mathsf{pk}_{ots}, \eta))$ while the user updates its coin $C \leftarrow (x, \sigma, j+V)$.
- $\mathtt{Deposit}(\mathcal{M}(\mathsf{msk}, \mathsf{bpk}, (V, Z, \Pi)), \mathcal{B}(\mathsf{bsk}, L, \mathsf{mpk}))$: When a transcript is deposited by a merchant, the bank parses it as $(V, (\phi_{V,j}, \psi_{V,j}), (\pi, \pi', \mathsf{pk}_{ots}, \eta))$ and checks its validity (in the same way as the merchant did during the `Spend` protocol). $\mathcal{B}$ also verifies that it does not already exist in its database.
  If everything is correct, $\mathcal{B}$ computes, for $1 \leq j' \leq N+1-V$ and $j' \leq i \leq j'+V-1$, the elements $z_{j',i} \leftarrow e(\phi_{V,j'}, \widetilde{g}_{(j',V)\mapsto i})$. If none of these values is in L, the bank adds them to this list and stores the associated transcript. Else, there is at least one $z' \in L$ (associated with a transcript (V', Z', Π')) and one pair $(j^*, i^*) \in [0, N+1-V] \times [j', j'+V-1]$ such that $z' = z_{j^*,i^*}$. The bank then outputs the two transcripts (V, Z, Π) and (V', Z', Π') as a proof of a double-spending.
- $\mathtt{Identify}((V_1, Z_1, \Pi_1), (V_2, Z_2, \Pi_2), \mathsf{bpk})$: The first step before identifying a double-spender is to check the validity of both transcripts and that there is a collision between their serial

numbers, *i.e.* there are $(j_1^*, i_1^*) \in [0, N+1-V_1] \times [j_1^*, j_1^*+V_1-1]$ and $(j_2^*, i_2^*) \in [0, N+1-V_2] \times [j_2^*, j_2^*+V_2-1]$ such that:

$$\begin{aligned} z_{(j_1^*,i_1^*)} &= e(\phi_{V_1,j_1}, \widetilde{g}_{(j_1^*,V_1)\mapsto i_1^*}) \\ &= e(\phi_{V_2,j_2}, \widetilde{g}_{(j_2^*,V_2)\mapsto i_2^*}) = z_{(j_2^*,i_2^*)} \end{aligned}$$

Let T_b be $e(\psi_{V_b,j_b}, \widetilde{g}_{(j_b^*,V_b)\mapsto i_b^*})$, for $b \in \{1,2\}$. The algorithm checks, for each registered public key upk_i, whether $T_1 \cdot T_2^{-1} = e(\mathsf{upk}_i, (\widetilde{g}_{(j_1^*,V_1)\mapsto i_1^*})^{R_1} \cdot (\widetilde{g}_{(j_2^*,V_2)\mapsto i_2^*})^{-R_2})$ until it gets a match. It then returns the corresponding key upk^* (or $\perp$ if the previous equality does not hold for any upk_i), allowing anyone to verify, without the linear cost in the number of users, that the identification is correct.

Remark 22. One can note that, among the $V(N+1-V)$ elements $z_{j,i}$ computed (and stored) by the bank during a deposit of value V, only V of them will be valid serial numbers. One must then take care that the other (invalid) ones will not lead to false positives, *i.e.* a collision in L which would not be due to a double-spending.

So let us consider a collision as described in the `Deposit` protocol. Let x_1 (resp. x_2) be such that $\phi_{V_1,j_1} = s_{j_1,V_1}^{x_1}$ (resp. $\phi_{V_2,j_2} = s_{j_2,V_2}^{x_2}$). We then have:

$$\begin{aligned} e(\phi_{V_1,j_1}, \widetilde{g}_{(j_1^*,V_1)\mapsto i_1^*}) &= e(s_{j_1,V_1}^{x_1}, \widetilde{g}_{(j_1^*,V_1)\mapsto i_1^*}) \\ &= e(g, \widetilde{g})^{x_1 \cdot r_{j_1,V_1} \cdot y_{i_1^*}/r_{j_1^*,V_1}} \\ &= e(g, \widetilde{g})^{x_2 \cdot r_{j_2,V_2} \cdot y_{i_2^*}/r_{j_2^*,V_2}} \\ &= e(s_{j_2,V_2}^{x_2}, \widetilde{g}_{(j_2^*,V_2)\mapsto i_2^*}) = e(\phi_{V_2,j_2}, \widetilde{g}_{(j_2^*,V_2)\mapsto i_2^*}) \end{aligned}$$

A collision thus implies that $x_1 \cdot r_{j_1,V_1} \cdot y_{i_1^*}/r_{j_1^*,V_1} = x_2 \cdot r_{j_2,V_2} \cdot y_{i_2^*}/r_{j_2^*,V_2}$. Since x_1 and x_2 have been cooperatively generated without knowledge of the secret, random scalars y_j and $r_{j,k}$, such an equality is unlikely to hold for distinct x_1 and x_2. Since the scalars y_j and $r_{j,k}$ have been generated independently, we can also conclude, with overwhelming probability, that $y_{i_1^*} = y_{i_2^*}$ (and so $i_1^* = i_2^*$). We then get the following equality: $r_{j_1,V_1}/r_{j_1^*,V_1} = r_{j_2,V_2}/r_{j_2^*,V_2}$. We distinguish two cases:

- Case 1: $j_1 = j_1^*$. Therefore, $r_{j_1,V_1}/r_{j_1^*,V_1} = 1$ and so $j_2 = j_2^*$. The value $z_{(j_1^*,i_1^*)} = z_{(j_2^*,i_2^*)}$ is then equal to $\mathtt{SN}_{i_1^*}$ ($= \mathtt{SN}_{i_2^*}$, since $i_1^* = i_2^*$). The user has thus used twice the same serial numbers and so has double-spent its coin.
- Case 2: $j_1 \neq j_1^*$. We then have $r_{j_1,V_1} = r_{j_2,V_2}$ and $r_{j_1^*,V_1} = r_{j_2^*,V_2}$ with overwhelming probability. Therefore, the same element $s_{j_1} = s_{j_2}$ has been used for two different transactions by the same user. The latter has thus double-spent his coin.

Efficiency. Compared to the divisible e-cash system described in Section 5, this construction offers the same complexity for the `Withdraw` protocol and slightly more efficient spendings. However, it suffers from two major drawbacks. First, the size of the public parameters is far more important. Indeed, $pp_{\mathcal{U}}$ (resp. $pp_{\mathcal{B}}$) now contains $O(N^2)$ elements in $\mathbb{G}_1$ (resp. $O(N^3)$ elements in $\mathbb{G}_2$). Second, each deposit of value V requires to perform $V(N+1-V)$ pairings and to store their outputs. This computational and storage cost can quickly become prohibitive, especially for a payment system which may have to support millions of daily transactions. We therefore argue that the construction of Section 5 should be preferred for practical purposes.

C.3 Security Results

The security of this construction is stated by the Theorems 26, 27 and 28 which make use of the EXDH assumption introduced in the full version of [CPST15a]. We also recall its weaker version (namely the weak-EXDH assumption) and show that both variants are actually equivalent.

Definition 23 (weak-EXDH assumption). *Given* $(g, g^x, g^a, g^{a\cdot y}) \in \mathbb{G}_1^4$ *and* $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$, *along with* $g^z \in \mathbb{G}_1$, *it is hard to distinguish whether* $z = x \cdot y \cdot a$ *or* z *is random.*

Definition 24 (EXDH assumption). *Given* $(g, h, g^x, h^x, g^a, h^a, g^{a\cdot y}, h^{a\cdot y}) \in \mathbb{G}_1^8$ *and* $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^y) \in \mathbb{G}_2^3$, *along with* $(g^{z_1}, h^{z_2}) \in \mathbb{G}_1^2$, *it is hard to distinguish whether* $z_1 = z_2 = x \cdot y \cdot a$ *or* (z_1, z_2) *is random.*

One can note that the weak-EXDH assumption has been introduced under a different name in [Duc10], namely the $\mathcal{P}$-BDH assumption.

Lemma 25. *The* EXDH *and weak-*EXDH *assumptions are equivalent.*

Proof. First, one can note that the EXDH assumption obviously implies the weak-EXDH one. Moreover, we note that the latter implies the DDH assumption in $\mathbb{G}_1$. Indeed, $(g, g^x, g^{y\cdot a}, g^z)$ is a valid DDH challenge in $\mathbb{G}_1$.

Now, let us consider an adversary $\mathcal{A}$ succeeding against the EXDH assumption with a non-negligible advantage $\mathtt{Adv}(\mathcal{A})$:

$$\mathtt{Adv}(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z = x \cdot y] - \Pr[\mathcal{A}(\mathcal{S}, g^{z_1}, h^{z_2}) | z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p]$$

where $\mathcal{S} = \{g, h, g^x, h^x, g^a, h^a, g^{y\cdot a}, h^{y\cdot a}\}$. We define the following two advantages:

$$\mathtt{Adv}_1(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z = x \cdot y] - \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z \xleftarrow{\$} \mathbb{Z}_p],$$
$$\mathtt{Adv}_2(\mathcal{A}) = \Pr[\mathcal{A}(\mathcal{S}, g^z, h^z) | z \xleftarrow{\$} \mathbb{Z}_p] - \Pr[\mathcal{A}(\mathcal{S}, g^{z_1}, h^{z_2}) | z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p].$$

Since $\mathtt{Adv}(\mathcal{A}) \leq \mathtt{Adv}_1(\mathcal{A}) + \mathtt{Adv}_2(\mathcal{A})$, at least one of the latter is non-negligible.

If $\mathtt{Adv}_1(\mathcal{A})$ is non-negligible, then $\mathcal{A}$ can be used against the weak-EXDH assumption: given a challenge $(g, g^x, g^a, g^{a\cdot y}, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^y, g^z)$ one selects a random $c \xleftarrow{\$} \mathbb{Z}_p$ and runs $\mathcal{A}$ on $(g, g^c, g^x, (g^x)^c, g^a, (g^a)^c, g^{a\cdot y}, (g^{a\cdot y})^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^y, g^z, (g^z)^c)$. The assumption made on $\mathtt{Adv}_1(\mathcal{A})$ implies that $\mathcal{A}$ will return, with non-negligible probability, a valid guess which can be used for the weak-EXDH instance.

If $\mathtt{Adv}_2(\mathcal{A})$ is non-negligible, then $\mathcal{A}$ can be used to solve the DDH problem in $\mathbb{G}_1$: given a challenge (g, g^x, g^y, g^z), one selects random $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ and runs $\mathcal{A}$ on $(g, g^x, g^b, (g^x)^b, g^a, (g^x)^a, g^{a\cdot c}, (g^x)^{a\cdot c}, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^c, g^y, g^z)$. By setting $h = g^x$, one can see that the latter tuple is a valid EXDH instance and that the guess returned by $\mathcal{A}$ can be used to solve the DDH problem. Therefore, $\mathcal{A}$ can be used against the DDH assumption and so against the stronger weak-EXDH assumption.

The weak-EXDH assumption thus implies the EXDH one. Both assumptions are then equivalent. □

Theorem 26. *In the standard model, the protocol of Section C.2 is* ***anonymous*** *under the* SXDH *and* EXDH *assumptions.*

Theorem 27. *In the standard model, the protocol of Section C.2 is* ***traceable*** *if* Σ_0 *is an EUF-SCMA signature scheme,* Σ_1 *is an EUF-CMA signature scheme, and* H *is a collision-resistant hash function.*

Theorem 28. *Let* q *be a bound on the number of* $\mathcal{O}\mathtt{Spend}$ *queries made by the adversary. In the standard model, the protocol of Section C.2 achieves the* ***exculpability property*** *under the* $q-$SDH *assumption if* Σ_{ots} *is a SUF-OTS signature scheme, and* H *is a collision-resistant hash function.*

Compared to the scheme of Section 5, we no longer need the $N-$BDHI assumption and the anonymity property now relies on the EXDH assumption (and so on the weak-EXDH one), instead of the MXDH′ one. Although we cannot prove that the former is weaker than the latter, we argue that we can be more confident in the weak-EXDH assumption since it is much easier to study and it has already been used in another context [Duc10].

The proof of exculpability is similar to the one of Section 6.2. Regarding traceability, Type-3 forgeries of Section 6.1 are now impossible due to the soundness of the proof system: the elements $\phi_{V,j}$ and $\psi_{V,j}$ must have been constructed using a certified pair $(s_j, t_j) \in \mathbb{G}_1^2$. Therefore, the $N-$BDHI assumption is no longer necessary. The proof of anonymity is provided below.

C.4 Proof of Theorem 26

We prove that this new construction satisfies the anonymity requirement by using a sequence of games, starting from the anonymity game defined in Figure 2 for a bit b. We want to show that the advantage ϵ is negligible.

In a next game, the simulator makes a guess $\ell^* \in [1, q_w]$ (where q_w is a bound on the number of $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ queries) on the coin used by upk_b during the step 6 of the anonymity game, *i.e.* it assumes that it is the one withdrawn during the ℓ^* query. It also makes a guess on the value $V^* \in [1, N]$ chosen by $\mathcal{A}$ during the step 3 of this experiment and on the index $j^* \in [1, N - V^* + 1]$ of the coin at the challenge time. These guesses do not affect the success probability of the adversary if they are correct but reduce the advantage from ϵ to $\epsilon/(q_w \cdot N^2)$. Next, the simulator generates the CRS for the Groth-Sahai proofs system in the perfect witness indistinguishability setting. This game is indistinguishable from the previous one under the SXDH assumption

Now, the simulator will generate the public parameters from a EXDH challenge $(g, h, g^x, h^x, g^a, h^a, g^{a \cdot y}, h^{a \cdot y}, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^y, g^{z_1}, h^{z_2})$. It first selects random $y_j \xleftarrow{\$} \mathbb{Z}_p$ and $r_{j,k}$ for $1 \leq j \leq N$ and $1 \leq k \leq N + 1 - j$, along with random d_1 and d_2. It then computes:

- $(u_1, u_2) \leftarrow (g^{d_1}, g^{d_2})$
- for $1 \leq j \leq N$ and $1 \leq k \leq N + 1 - j$:
  - if $j \geq j^* + V^*$, then $(s_{j,k}, t_{j,k}) \leftarrow (g^{r_{j,k}}, h^{r_{j,k}})$
  - if $j^* \leq j < j^* + V^*$, then
    - $(s_{j,k}, t_{j,k}) \leftarrow ((g^a)^{r_{j,k}}, (h^a)^{r_{j,k}})$ if $j + k \geq j^* + V^*$
    - $(s_{j,k}, t_{j,k}) \leftarrow ((g^{a \cdot y})^{r_{j,k}}, (h^{a \cdot y})^{r_{j,k}})$ otherwise
  - if $j < j^*$, then
    - $(s_{j,k}, t_{j,k}) \leftarrow ((g^a)^{r_{j,k}}, (h^a)^{r_{j,k}})$ if $j + k > j^*$
    - $(s_{j,k}, t_{j,k}) \leftarrow (g^{r_{j,k}}, h^{r_{j,k}})$ otherwise
- for $1 \leq j \leq N$, $1 \leq k \leq N + 1 - j$ and $j \leq i \leq j + k - 1$:
  - if $j \geq j^* + V^*$, then $\widetilde{g}_{(j,k) \mapsto i} \leftarrow (\widetilde{g}^a)^{y_i / r_{j,k}}$
  - if $j^* \leq j < j^* + V^*$, then
    - $\widetilde{g}_{(j,k) \mapsto i} \leftarrow (\widetilde{g}^y)^{y_i / r_{j,k}}$ if $j + k \geq j^* + V^*$ and $i < j^* + V^*$
    - $\widetilde{g}_{(j,k) \mapsto i} \leftarrow \widetilde{g}^{y_i / r_{j,k}}$ otherwise
  - if $j < j^*$, then
    - $\widetilde{g}_{(j,k) \mapsto i} \leftarrow (\widetilde{g}^y)^{y_i / r_{j,k}}$ if $j + k > j^*$ and $i \geq j^*$
    - $\widetilde{g}_{(j,k) \mapsto i} \leftarrow \widetilde{g}^{y_i / r_{j,k}}$ if $j + k > j^*$ and $i < j^*$
    - $\widetilde{g}_{(j,k) \mapsto i} \leftarrow (\widetilde{g}^a)^{y_i / r_{j,k}}$ otherwise

The simulation is therefore correct since, for any $1 \leq j \leq N$, $1 \leq k \leq N + 1 - j$ and $j \leq i \leq j + k - 1$:

- $e(s_{j,k}, \widetilde{g}_{(j,k) \mapsto i}) = e(g, \widetilde{g})^{a \cdot y_i}$ if $i < j^*$ or $i \geq j^* + V^*$
- $e(s_{j,k}, \widetilde{g}_{(j,k) \mapsto i}) = e(g, \widetilde{g})^{a \cdot y \cdot y_i}$ otherwise

The simulator now proceeds as follows to answer oracle queries:

- $\mathcal{O}\mathtt{Add}()$ queries: run the Keygen algorithm and return upk (or mpk);
- $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}(\mathsf{bsk}, \mathsf{upk})$ queries: for the ℓ^{th} $\mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$ query, the simulator plays normally if $\ell \neq \ell^*$, but sending the pair $(u_1^{\mathsf{usk}}, (g^x)^{d_2} = u_2^x)$ otherwise (using the EXDH instance). It can then simulate the proof of knowledge and receives a scalar x' along with a signature σ on $(u_1^{\mathsf{usk}}, u_2^{x^*})$, where $x^* = x + x'$. The coin is then implicitly defined as $C^* = (x^*, \sigma, 1)$ and we will now denote its owner by upk^*;
- $\mathcal{O}\mathtt{Corrupt}(\mathsf{upk}/\mathsf{mpk})$ queries: the simulator plays normally (if the guesses are correct, upk^* cannot be asked to be corrupted);
- $\mathcal{O}\mathtt{AddCorrupt}(\mathsf{upk}/\mathsf{mpk})$: the simulator stores the public key which is now considered as registered;

– $\mathcal{O}\mathtt{Spend}(\mathsf{upk}, V)$ queries: if $\mathsf{upk} \neq \mathsf{upk}^*$ or the coin to be used for the spending has not been withdrawn during the $\ell^* - \mathcal{O}\mathtt{Withdraw}_{\mathcal{U}}$-query, then the simulator knows all the secret keys, and so it can plays normally. Otherwise, it proceeds as follows. Let j be the index of the coin to be spend, $j \notin [j^* - V + 1, j^* + V^* - 1]$ if the guesses on j^* and V^* are correct. Otherwise this spending and the challenge spending would lead to a double-spending. Therefore, $(s_{j,V}, t_{j,V}) = (g^{r_{j,V}}, h^{r_{j,V}})$ for a known scalar $r_{j,V}$. The simulator is then able to return a valid pair $(\phi_{V,j}, \psi_{V,j}) = ((g^x \cdot g^{x'})^{r_{j,V}}, (h^x \cdot h^{x'})^{r_{j,V}} \cdot \mathsf{upk}^R)$ along with simulated proofs: the simulation is correct.

During the challenge phase, $\mathcal{A}$ outputs two public keys upk_0 and upk_1 and a value V. If the guesses were correct, $V = V^*$, $\mathsf{upk}^* = \mathsf{upk}_b$ and the coin involving x^* is spent at index $j = j^*$. The simulator then returns $(\phi_{V^*,j^*}, \psi_{V^*,j^*}) = ((g^{z_1} \cdot (g^{a \cdot y})^{x'})^{r_{j^*,V^*}}, (h^{z_2} \cdot (h^{a \cdot y})^{x'})^{r_{j^*,V^*}} \cdot (\mathsf{upk}^*)^R)$ along with simulated proofs. If $z = x \cdot y \cdot a$, then the pair $(\phi_{V^*,j^*}, \psi_{V^*,j^*})$ is valid. Indeed:

$$\begin{aligned}
\phi_{V^*,j^*} &= (g^{a \cdot y})^{(x+x') \cdot r_{j,k}} = s_{j^*,V^*}^{x^*} \\
\psi_{V^*,j^*} &= (h^{a \cdot y})^{(x+x') \cdot r_{j,k}} \cdot (\mathsf{upk}^*)^R = t_{j^*,V^*}^{x^*} \cdot (\mathsf{upk}^*)^R
\end{aligned}$$

Finally, we replace in the last game the EXDH challenge by a random one with random z_1 and z_2. The elements ϕ_{V^*,j^*} and ψ_{V^*,j^*} then perfectly hide upk^*, so the advantage of the adversary in this game is zero.