# Divisible E-cash Systems can be Truly Anonymous*

Sébastien Canard[1] and Aline Gouget[2]

[1] France Télécom R&D, 42 rue des Coutures, F-14066 Caen, France.
[2] Gemalto, 6, rue de la Verrerie, F-92190 Meudon, France.

**Abstract.** This paper presents an off-line divisible e-cash scheme where a user can withdraw a divisible coin of monetary value $2^L$ that he can parceled and spend anonymously and unlinkably. We present the construction of a security tag that allows to protect the anonymity of honest users and to revoke anonymity only in case of cheat for protocols based on a binary tree structure without using a trusted third party. This is the first divisible e-cash scheme that provides both full unlinkability and anonymity without requiring a trusted third party.

## 1 Introduction

Electronic cash systems allow users to withdraw electronic coins from a bank, and then to pay a merchant using electronic coins preferably without communicating with the bank or a trusted party during the payment. Finally, the merchant deposits the spent coins to the bank.

Electronic cash provides user anonymity against both the bank and the merchant during a purchase in order to emulate the perceived anonymity of regular cash transaction. It must be impossible to link two spending protocols and a spending protocol to a withdrawal protocol.

As it is easy to duplicate electronic data, an e-cash system must prevent a user from double-spending. Ideally, the anonymity of honest users must be protected and the identity of cheaters must be recovered without using a trusted third party. An electronic payment system must also prevent a merchant from depositing the same coin twice.

To be practical, an e-cash system must be based on efficient protocols. The most critical protocol is the spending phase between the user and the merchant that must be reasonably efficient. It should also be possible to withdraw or spend several coins more efficiently than repeating several times a single withdrawal or spending protocol.

* This work has been partially financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by the French Ministry of Research RNRT Project "CRYPTO++" .

### 1.1 Related Works

The compact E-cash scheme [4] allows to withdraw efficiently a wallet containing $2^L$ coins and provides all the security properties mentioned above. One solution to improve the efficiency of the spending phase is to manage a wallet that contains coins with several monetary values as it was done in [8]; the main drawback is that the user must choose during the withdrawal protocol how many coins he wants for each monetary value.

Divisible e-cash schemes allow a user to withdraw a coin of monetary value $2^L$ and then to spend this coin in several times by dividing the value of the coin. The aim is to allow a user to efficiently spend a coin of monetary value $2^\ell$, $0 \leq \ell \leq L$, (i.e. more efficiently than repeating $2^\ell$ times a spending protocol). Many off-line *divisible e-cash* systems have been proposed in the literature [22, 23, 13, 14, 21, 9, 20, 19] providing part of the security properties mentioned above. The first practical divisible e-cash system was proposed by Okamoto [21] and improved by Chan *et al.* in [9]. Both schemes provide anonymity of users but not unlinkability since it is still possible to link several spends from a single divisible coin.

The first *unlinkable divisible* e-cash system that fulfills the usual properties of anonymity and unlinkability was proposed in [20] and improved in [19]. The main drawback of these two systems is that they require a trusted third party to get the identity of the user in case of double-spend detection: this is consequently what we can call a *fair* divisible e-cash system. Moreover, the unlinkability provided by [20, 19] is not strong since the merchant and the bank know which part of the withdrawn divisible coin the user is spending which is an information leak on the user.

None of the divisible e-cash schemes of the state of the art provides simultaneously strong unlinkability and truly anonymity of users.

### 1.2 Our Contribution

We present a strong unlinkable and anonymous divisible off-line e-cash system without trusted third party. We first provide a generic construction and next apply it to the construction of Nakanishi and Sugiyama [20]. Our system is the first that provides the user anonymity such that it is impossible for anybody to make any link between spends and withdraws. Furthermore, our construction does not require a trusted third party to revoke the anonymity of a user that has spent twice the same coin. From a theoretical point of view, the identity of the user can only be revealed when such a case happens. This is the first divisible e-cash system providing this security property.

### 1.3 Organization of the Paper

This paper is organized as follows. Section 2 describes the security model and requirements for a divisible e-cash system. In Section 3, we present the general principle of the construction. Section 4 is the main one: it contains the new divisible e-cash called $\mathcal{DCS}$. Finally, in Section 5, we give the security proofs of our construction.

## 2 Security Model

We adopt the model of divisible e-cash system without trusted third party. The three usual players are the user $\mathcal{U}$, the bank $\mathcal{B}$ and the merchant $\mathcal{M}$. The security parameter is denoted by $k$.

### 2.1 Algorithms

- `ParamKeyGen`$(k)$**:** a probabilistic algorithm outputting the parameters of the system $Params$ ($Params$ contains the parameter $k$).
- `BKeyGen`$(Params)$**:** a probabilistic algorithm executed by $\mathcal{B}$ outputting the key pair $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$.
- `KeyGen`$(Params)$**:** a probabilistic algorithm executed by $\mathcal{U}$ (resp. $\mathcal{M}$) outputting $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$ (resp. $(sk_{\mathcal{M}}, pk_{\mathcal{M}})$).
- `Withdraw`$(\mathcal{B}(sk_{\mathcal{B}}, pk_{\mathcal{B}}, pk_{\mathcal{U}}, Params),\ \mathcal{U}(sk_{\mathcal{U}}, pk_{\mathcal{U}}, pk_{\mathcal{B}}, Params))$**:** an interactive protocol between $\mathcal{B}$ and $\mathcal{U}$. At the end, either $\mathcal{U}$ gets a divisible coin $\mathcal{C}$ of monetary value $2^L$ ($L$ belongs to $Params$) and outputs $OK$, or $\mathcal{U}$ outputs $\perp$. The output of $\mathcal{B}$ is either its view $\mathcal{V}_{\mathcal{B}}^{\mathtt{Withdraw}}$ of the protocol (including $pk_{\mathcal{U}}$), or $\perp$.
- `Spend`$(\mathcal{U}(2^\ell, pk_{\mathcal{M}}, \mathcal{C}, Params), \mathcal{M}(sk_{\mathcal{M}}, pk_{\mathcal{B}}, Params))$**:** an interactive protocol between $\mathcal{U}$ and $\mathcal{M}$. At the end, either $\mathcal{M}$ obtains a master serial number $S$ and a proof of validity $\Pi$ and outputs $(S, \Pi)$ or $\mathcal{M}$ outputs $\perp$. Either $\mathcal{U}$ updates $\mathcal{C}$ by saving the part of the divisible coin he spent (i.e. the value $S$) and outputs OK, or $\mathcal{U}$ outputs $\perp$.
- `Deposit` $(\mathcal{M}((S, \Pi), sk_{\mathcal{M}}, pk_{\mathcal{M}}, pk_{\mathcal{B}}, Params), \mathcal{B}(pk_{\mathcal{M}}, Params))$**:** an interactive protocol between $\mathcal{M}$ and $\mathcal{B}$. During the deposit, $\mathcal{B}$ receives $(S, \Pi)$ from $\mathcal{M}$, checks that it is fresh and that $\Pi$ is correct. If not, $\mathcal{B}$ outputs $\perp_1$. Else $\mathcal{B}$ computes $2^\ell$ serial numbers $\widetilde{S}_1, \ldots, \widetilde{S}_{2^\ell}$ from $(S, \Pi)$ and $Params$. If one of the serial number $(\widetilde{S}_i, S', \Pi')$ already belongs to $\mathcal{L}$, then the bank outputs $(\perp_2, S, \Pi, S', \Pi')$. Otherwise, $\mathcal{B}$ adds $(\widetilde{S}_i, S, \Pi)$, $1 \leq i \leq 2^\ell$, to its list $\mathcal{L}$ of spent coins, credits $\mathcal{M}$'s account, and returns $\mathcal{L}$. $\mathcal{M}$'s output is $OK$ or $\perp$.

- `Identify`$((S_1, \Pi_1), (S_2, \Pi_2), Params)$**:** a deterministic algorithm executed by $\mathcal{B}$ that outputs a public key $pk_{\mathcal{U}}$ and a proof $\Pi_G$. If $\mathcal{M}$s who had submitted $\Pi_1$ and $\Pi_2$ are not malicious, then $\Pi_G$ is evidence that $pk_{\mathcal{U}}$ is the registered public key of a user that double-spent a coin.
- `VerifyGuilt`$(pk_{\mathcal{U}}, \Pi_G, Params)$**:** a deterministic algorithm executed by any actor that outputs 1 if the proof is correct and 0 otherwise. This verification permits anyone to be sure that the user with public key $pk_{\mathcal{U}}$ is guilty of double-spending a coin.

### 2.2 Notions of Security

In the following, it is assumed that the overlying experiment has run the algorithm `ParamKeyGen` on input $k$ to obtain the parameters $Params$.

- **Unforgeability.** Let $\mathcal{A}$ be a p.p.t. Turing Machine. At the start of the game, $\mathcal{A}$ is given the public key $pk_{\mathcal{B}}$ and $Params$. Suppose that $\mathcal{A}$ interacts $K$ times with an honest bank during withdrawal protocols, then the probability that the number of valid coins that has been spent is at least $2^L K + 1$ is negligible.
- **Unlinkability.** Let $\mathcal{A}$ be a p.p.t. Turing Machine. At the start of the game, $\mathcal{A}$ is given the key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ and $Params$. At the end, $\mathcal{A}$ chooses two honest users 0 and 1. A bit $b$ is secretly and randomly chosen. Then, a spending protocol is played by $\mathcal{A}$ with user $b$ (it is assumed that both honest users still have unspent coins). Finally, $\mathcal{A}$ outputs a bit $b'$. We require that for every $\mathcal{A}$ playing this game, the probability that $b = b'$ differs from 1/2 by a fraction that is at most negligible.
- **Identification of double-spenders.** Let $\mathcal{A}$ be a p.p.t. Turing Machine. At the start of the game, $\mathcal{A}$ is given the public key $pk_{\mathcal{B}}$ and $Params$. The probability that a `Deposit` protocol between an honest merchant and an honest bank outputs $(\perp_2, S, \Pi, S', \Pi')$ such that the output of `Identify` algorithm on inputs $(S, \Pi, S', \Pi')$ is not the public key $pk_{\mathcal{U}}$ of a corrupted user is negligible.
- **Exculpability.** Let $\mathcal{A}$ be a p.p.t. Turing Machine. At the start of the game, $\mathcal{A}$ is given the key pair $(pk_{\mathcal{B}}, sk_{\mathcal{B}})$ and $Params$. During the game, $\mathcal{A}$ interacts with honest users to supply them coins. At the end, $\mathcal{A}$ constructs two spent coins $(S_1, \Pi_1)$ and $(S_2, \Pi_2)$. The probability that the outputs of the `Identify` algorithm on inputs $(S_1, \Pi_1)$ and $(S_2, \Pi_2)$ is the public key $pk_{\mathcal{U}}$ of an honest user together with a valid proof $\Pi_G$ is negligible.

*Remark 1.* Notice that the exculpability property implies that the bank cannot create withdrawals for which the user has not participated. We don't need any extra security property, such as the proposal in [28].

## 3 General Description

In an anonymous e-cash system without a trusted third party, spending a single coin consists in generating a valid serial number $S$ to allow double-spending detection and a valid security tag $T$ masking the identity of the spender. The spender has to prove that $S$ and $T$ are well-formed without giving any information about his identity. In particular, the identity of the spender must be recovered only in case of double-spending by using the security tag $T$.

The main motivation of divisible e-cash is to provide a method to withdraw or spend several coins more efficiently than repeating several times a single withdrawal or spending protocol. We provide a general approach to construct divisible e-cash systems strongly unlinkable and truly anonymous (the user identity can be recovered only in case of fraud). This construction can be applied using several basic cryptographic tools.

### 3.1 Truly Anonymous E-cash Scheme based on Binary Trees

The general principle of our construction is derived from the classical binary tree approach [21, 9, 20] with slight modifications. Each divisible coin of monetary value $2^L$ is assigned to a binary tree of $L+2$ levels. The tree root (level 0) with monetary value $2^L$ is assigned to a serial number denoted by $N_{0,0}$. Any other node has a monetary value corresponding to half of the amount of its parent node, except for the leaves that have no monetary value: they are "dead" leaves. For every level $i$, $0 \leq i \leq L$, the $2^i$ nodes are assigned serial numbers denoted by $N_{i,j}$ with $1 \leq j \leq 2^i$, except for the "dead" leaves that are not related to any serial number. Any *divisible* e-cash system should verify the divisibility rule.

**Definition 1.** *When a node $N$ is used, none of descendant and ancestor nodes of $N$ can be used, and no node can be used more than once.*

This rule is satisfied if, and only if, over-spending is protected. The general principle of our proposal consists in using a single master serial number from which several serial numbers can be derived. Thus, each node of the tree, which includes the leaves, is also related to a particular value called a *tag key*. During the spending protocol, the identity of the spender is encrypted with a tag key in such a way that the decryption key can be derived only in case of a double-spending. Using the binary tree approach, each node of the tree is related to a tag key with the following properties.

- The root tag key and the identity of the user are signed (in a blind manner) by the bank during the withdrawal protocol.
- From the tag key of a node $N$, it is possible for everyone to compute the tag keys related to the descendant nodes of $N$. It consequently exists a public deterministic function $\mathcal{F}$ that takes as input a tag key $K_{i,b_0}$ (where $i$ is the level of the targeted node in the tree and $b_0 \in \{0,1\}$ depends on the position of $K$ in the tree[3]), a bit $b$ (0 for left and 1 for right) and possibly some public parameters $Params$ and that outputs a new tag key $K_{i+1,b}$.

$$\mathcal{F} : (K_{i,b_0}, b, Params) \longrightarrow K_{i+1,b} = \mathcal{F}(K_{i,b_0}, b, Params).$$

- From the tag key of a node, it is impossible (without the knowledge of the root tag key) to compute a tag key which is not related to a descendant of the targeted node.
- The serial number of a particular node is the concatenation of the two children tag keys. Notation is given in Figure 1.

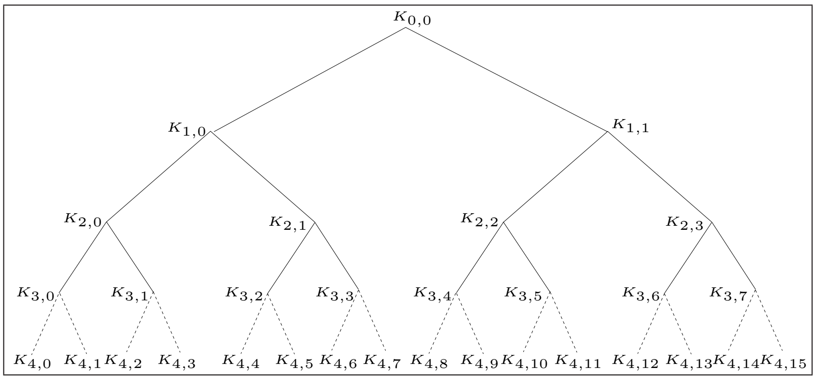


**Fig. 1.** General principle - Tree of keys

During the spending protocol, the user computes the tag key of the node he wants to spend. This tag key is used to compute the security tag, i.e. the encryption of the spender identity. This encryption should be

[3] $b_0 = 0$ if and only if the targeted node belongs to the left subtree of its ancestor.

verifiable and should include randomness. This randomness should be provided by the merchant to ensure the freshness of the spending, i.e., to prevent merchant from sending twice the same coin to the bank. The user also computes the tag keys related to the two direct descendants of the spent node. The concatenation of these two keys is the serial number of the spent coin. This serial number is transmitted during the spend protocol. Later, the bank will compute all the serial numbers of the leaves of the tree in order to detect a possible double-spending. If a double-spending is detected, then the bank has access to the encryption of the identity (from one spending) and the corresponding decryption key (from the other spending). Then, the bank can easily find the identity of the cheater.

*Example 1.* Assume $\mathcal{U}$ wants to spend four coins. Then, $\mathcal{U}$ selects four unitary coins, e.g. those associated to the node $K_{1,0}$. The user $\mathcal{U}$ sends to $\mathcal{M}$ the values $T = E_{K_{1,0}}(Id, R)$, $LK = K_{2,0}$, $RK = K_{2,1}$, and $S = LK \| RK$. The random value $R$ used in the encryption scheme is computed using values sent by the merchant. The user must also prove that the coins are signed by the bank and that it will be possible to identify a double-spender. Consequently, the spending protocol consists also in computing a zero-knowledge proof of knowledge $\Phi$ that corresponds to the predicates:

- $T$ is well-formed, i.e. $E_{K_{1,0}}(Id, R)$ has been computed using:
  - the tag key $K_{1,0}$ derived using $\mathcal{F}$ on inputs the root tag key $K_{0,0}$ signed by the bank,
  - the random $R$ that has been chosen by the merchant,
  - the identity $Id$ signed by the bank.
- $LK$ and $RK$ are well-formed, i.e., $K_{2,0}$ and $K_{2,1}$ are both derived from $K_{1,0}$ using $\mathcal{F}$.
- If $LK$ and $RK$ are well-formed, this implies that the serial number $S$ is also well-formed.

To construct a truly anonymous divisible e-cash system, it is then necessary to provide a function $\mathcal{F}$, a verifiable encryption scheme $E$ and a proof $\Phi$. We give an example in Section 4.

### 3.2 Useful Tools

**Proofs of Knowledge.** We use zero-knowledge proofs of knowledge constructed over a cyclic group $\mathcal{G}$ either of prime order $q$ or of unknown order: proof of equality of two known representations [10, 6], proofs of knowledge of a discrete logarithm [26, 17], of a representation, of a double discrete logarithm $PK(\alpha/z = g^\alpha \wedge y = g_1^{g_2^\alpha})$ [27, 20], proof of the "or" statement $PK(\alpha/T_1 = h_1^\alpha \vee T_2 = h_2^\alpha)$ [11, 25]. We also need a proof of knowledge of one out of two double discrete logarithm $PK(\alpha/T_1 = g^{h_1^\alpha} \vee y = g^{h_2^\alpha})$ which is a combination of the two above proofs. These proofs can also be used non interactively by using the Fiat-Shamir heuristic [16].

**Camenisch-Lysyanskaya Signature Schemes.** These signature schemes are proposed in [5] with in addition some specific protocols:

- an efficient protocol between a user $\mathcal{U}$ and a signer $\mathcal{S}$ that permits $\mathcal{U}$ to obtain from $\mathcal{S}$ a signature $\sigma$ of some commitment $C$ on values $(x_1, \ldots, x_l)$ unknown from $\mathcal{S}$. $\mathcal{S}$ computes $\texttt{CLSign}(C)$ and $\mathcal{U}$ gets $\sigma = \texttt{Sign}(x_1, \ldots, x_l)$ that can be verified by $\texttt{Verif}(\sigma, (x_1, \ldots, x_l)) = 1$.
- an efficient proof of knowledge of a signature on committed values, denoted by $PK(\alpha_1, \ldots, \alpha_l, \beta/\beta = \texttt{Sign}(\alpha_1, \ldots, \alpha_l))$.

These constructions are quite close to group signature schemes. This is the case of the two following examples, one based on the ACJT signature scheme [1], secure under the Flexible RSA assumption [15], and the other based on the BBS one [2], secure under the $q$-SDH assumption [2].

## 4 Divisible E-cash System $\mathcal{DCS}$

We apply the general construction presented in Section 3.1 to the binary tree used in the system described in [20]. The function $\mathcal{F}$ is chosen to be the modular exponentiation. For each level $i$, there are three linked generators $g_{i,0}$ for "left", $g_{i,1}$ for "right" and $g_{i,2}$ to compute the security tag. For a node at level $i-1$ represented by the tag key denoted by $K_{i-1,b_0}$, the tag key of, e.g. the left children, is $K_{i,0} = g_{i,0}^{K_{i-1,b_0}}$. For the tag key $K_{i,b}$ and a random value $R$ computing using merchant data, the encryption of the user identity $pk_\mathcal{U}$ is defined to be $pk_\mathcal{U} g_{i+1,2}^{K_{i,b} \cdot R}$. In the following, we assume that $\mathcal{H}$ is a collision-resistant hash function.

### 4.1 Setup

We consider a group $\mathcal{G}$ of order $o_\mathcal{G}$. The elements $h_0$,$h_1$, $h_2$ are random generators of $\mathcal{G}$. $\mathcal{G}_1 = \langle g_1 \rangle$ is a subgroup of $\mathbb{Z}_{o_\mathcal{G}}^*$ and each group $\mathcal{G}_i = \langle g_i \rangle$ must be a subgroup of $\mathbb{Z}_{o_{i+1}}^*$ where $o_{i+1}$ is the order of $\mathcal{G}_{i+1}$. For example [20], it is possible to take $\mathcal{G}_i$ as a subgroup of $\mathbb{Z}_{o_i+1}^*$ for the prime $o_{i+1} = 2o_i + 1$ with all $i$. As a consequence, the group $\mathcal{G}_i$ is related to the level $i$ of the tree. The following generators are randomly chosen: $g$ in

$\mathcal{G}$, $g_{1,0}$, $g_{1,1}$, $g_{1,2}$ in $\mathcal{G}_1$, $g_{2,0}$, $g_{2,1}$, $g_{2,2}$ in $\mathcal{G}_2$, ..., $g_{L+1,0}$, $g_{L+1,1}$, $g_{L+1,2}$ in $\mathcal{G}_{L+1}$ whose discrete logarithms to the base $g_1, g_2, \ldots, g_{L+1}$ are unknown, respectively. All these data compose the public parameters *Params* of the system and can be computed by the bank. The bank $\mathcal{B}$ computes the key pair $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ of a Camenisch-Lysyanskaya signature scheme that will permit it to sign a divisible coin, using the `CLSign` algorithm.

A user $\mathcal{U}$ (resp. a merchant $\mathcal{M}$) can compute its key pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$ (resp. $(sk_{\mathcal{M}}, pk_{\mathcal{M}})$) by choosing randomly $u \in [0, o_{\mathcal{G}}[$ (resp. $m \in [0, o_{\mathcal{G}}[$) and computing $g^u$ (resp. $g^m$). The value $u$ (resp $m$) is the private key $sk_{\mathcal{U}}$ (resp. $sk_{\mathcal{M}}$) and $g^u$ (resp. $g^m$) is equal to the public key $pk_{\mathcal{U}}$ (resp. $pk_{\mathcal{M}}$).

### 4.2 Withdrawal Protocol

During a withdrawal protocol, $\mathcal{U}$ interacts with $\mathcal{B}$. $\mathcal{U}$'s inputs are $pk_{\mathcal{B}}$, $sk_{\mathcal{U}}$, $pk_{\mathcal{U}}$ and *Params*, and $\mathcal{B}$'s inputs are $pk_{\mathcal{U}}$, $sk_{\mathcal{B}}$, $pk_{\mathcal{B}}$ and *Params*.

| $\mathcal{U}$ | | $\mathcal{B}$ |
| --- | --- | --- |
| $s', r \in \mathbb{Z}^*_{o_{\mathcal{G}}}$<br>$C' = h_0^{s'} h_1^u h_2^r$<br>$U = PK(\alpha, \beta, \gamma / pk_{\mathcal{U}} = g^\alpha \wedge C' = h_0^\beta h_1^\alpha h_2^\gamma)$ | $\xrightarrow{C', U, pk_{\mathcal{U}}}$ | |
| | | Verify $U$<br>$r' \in \mathbb{Z}^*_{o_{\mathcal{G}}}$<br>$C = C' h_0^{r'}$<br>$\sigma = \mathtt{CLSign}(C)$ |
| | $\xleftarrow{r', \sigma}$ | |
| $s = s' + r' \pmod p$<br>$\mathtt{Verif}(\sigma, (s, u, r)) \stackrel{?}{=} 1$<br>$\mathcal{C} = (s, u, r, \sigma)$ | | $\mathcal{V}_{\mathcal{B}}^{\mathtt{Withdraw}} = (C, pk_{\mathcal{U}}, U, r', \sigma)$ |

**Fig. 2.** Withdrawal protocol

The withdrawal protocol permits $\mathcal{U}$ to obtain a new divisible coin by interacting with $\mathcal{B}$ as described in Figure 2. A divisible coin corresponds to a (blind) CL signature done by $\mathcal{B}$ on a secret $s$ and the secret key $u$ of $\mathcal{U}$. Both $\mathcal{U}$ and $\mathcal{B}$ participate to the randomness of the secret $s$. At the end of the `Withdraw` protocol, $\mathcal{U}$ gets a divisible coin $\mathcal{C} = (s, u, r, \sigma = \mathtt{Sign}(s, u, r))$.

### 4.3 Spending Protocol

When $\mathcal{U}$ wants to spend to $\mathcal{M}$ a sub-coin of value $2^\ell$ ($\ell = L - i$) from his divisible coin $\mathcal{C}$, he chooses an unspent node of the level $i$, e.g. the node $N_{i,j}$. A spending protocol of the node $N_{i,j}$ consists in the following.

1. $\mathcal{M}$ sends to $\mathcal{U}$ a random value $rand$ and $\mathcal{U}$ computes $R = \mathcal{H}(pk_{\mathcal{M}} \| rand)$.
2. $\mathcal{U}$ randomly chooses $\tilde{g}, \tilde{h} \in \mathcal{G}$, $\tilde{g}_1 \in \mathcal{G}_1$, $\tilde{g}_2 \in \mathcal{G}_2$, $\ldots$, $\tilde{g}_{i+1} \in \mathcal{G}_{i+1}$.
3. $\mathcal{U}$ executes the algorithm presented in Figure 3 (in pseudo-code) for the node $N_{i,j}$, outputting the values[4] $(\widetilde{V}_0, \ldots, \widetilde{V}_i, V)$, using the path from the root tree to the node $N_{i,j}$. Next, $\mathcal{U}$ computes the security

**Input:** $i, j$
**Output:** $(\widetilde{V}_0, \ldots, \widetilde{V}_i, V)$

$\tilde{r} \leftarrow$ `Rand()`, $V \leftarrow g^s$, $\widetilde{V}_0 \leftarrow \tilde{g}^s \tilde{h}^{\tilde{r}}$, `CurrentNode` $\leftarrow root$
**If** $i = 0$, **then** return $(\widetilde{V}_0, V)$
$a \leftarrow 1, b \leftarrow 2^i$
**For** $k = 1$ to $i$
  $\widetilde{V}_k \leftarrow \tilde{g}_k^V$
  **If** $a \leq j \leq a + (b - a - 1)/2$, **then** \\ $N_{i,j}$ belongs to `leftSubTree(CurrentNode)`
    $V \leftarrow (g_{k,0})^V$, $b \leftarrow a + (b - a - 1)/2$ \\ `CurrentNode` $\leftarrow$ `leftSon(CurrentNode)`
  **Else** \\ $N_{i,j}$ belongs to `rightSubTree(CurrentNode)`
    $V \leftarrow (g_{k,1})^V$, $a = a + (b - a + 1)/2$ \\ `CurrentNode` $\leftarrow$ `rightSon(CurrentNode)`
return $(\widetilde{V}_0, \ldots, \widetilde{V}_i, V)$

**Fig. 3.** Spending protocol - Computation of $V$

tag: $LK = g_{i+1,0}^V$, $RK = g_{i+1,1}^V$, $T = pk_{\mathcal{U}} g_{i+1,2}^{V \cdot R}$ and $S = LK \| RK$.

*Example 2.* Assume $\mathcal{U}$ wants to spend four coins (the same as in Example 1. The user $\mathcal{U}$ sends to the merchant $\mathcal{M}$ the values $LK = g_{2,0}^{g_{1,0}^{g^s}}$, $RK = g_{2,1}^{g_{1,0}^{g^s}}$, $T = pk_{\mathcal{U}}(g_{2,2}^{R \cdot g_{1,0}^{g^s}})$ and $S = LK \| RK$ since $V = g_{1,0}^{g^s}$.

4. $\mathcal{U}$ proves to $\mathcal{M}$ the validity of $LK, RK, T$ (and thus the validity of $S$) using a non-interactive zero-knowledge proof of knowledge of a signature of $\mathcal{B}$ on the values $(s, u, r)$ and that the value $LK, RK, T$ are correctly computed. This proof of knowledge is constructed from

[4] The values $\widetilde{V}_0, \ldots, \widetilde{V}_i$ are computed to prove that the value $V$ is well computed. See proof $\Phi$ below and [20].

a zero-knowledge proof of knowledge using the Fiat-Shamir heuristic. This proof is as follows:

$$\begin{aligned}
\Phi = PK\Big(&\sigma, s, u, r, \tilde{r}, \alpha_1, \ldots, \alpha_{i+1}, \beta \ / \\
&\sigma = \mathtt{Sign}(s, u, r) \ \wedge \ \widetilde{V}_0 = \tilde{g}^s \tilde{h}^{\tilde{r}} \ \wedge \ \widetilde{V}_1 = \tilde{g}_1^{g^s} \ \wedge \ \widetilde{V}_1 = \tilde{g}_1^{\alpha_1} \ \wedge \\
&(\widetilde{V}_2 = \tilde{g}_2^{g_{1,0}^{\alpha_1}} \ \vee \ \widetilde{V}_2 = \tilde{g}_2^{g_{1,1}^{\alpha_1}}) \ \wedge \ \widetilde{V}_2 = \tilde{g}_2^{\alpha_2} \ \wedge \ \ldots \ \wedge \\
&(\widetilde{V}_{i+1} = \tilde{g}_{i+1}^{g_{i,0}^{\alpha_i}} \ \vee \ \widetilde{V}_{i+1} = \tilde{g}_{i+1}^{g_{i,1}^{\alpha_i}}) \ \wedge \ \widetilde{V}_{i+1} = \tilde{g}_{i+1}^{\alpha_{i+1}} \ \wedge \\
&LK = g_{i+1,0}^{\alpha_{i+1}} \wedge RK = g_{i+1,1}^{\alpha_{i+1}} \wedge T = pk_{\mathcal{U}} g_{i+1,2}^{R \cdot \alpha_{i+1}} \Big)
\end{aligned}$$

5. $\mathcal{U}$ sends the spent coins $(S, \Pi)$ to $\mathcal{M}$, with $\Pi = \{2^\ell, T, \Phi, R, \widetilde{V}_0, \ldots, \widetilde{V}_i\}$.

### 4.4 Deposit Protocol

When $\mathcal{M}$ wants to deposit a coin $(S, \Pi)$ to $\mathcal{B}$, $\mathcal{M}$ just sends the coin $(S, \Pi)$ to $\mathcal{B}$. The proof $\Pi$ should include the monetary value $2^\ell$ of the divisible coin, the security tag $T$, the proof of knowledge $\Phi$ and the random data $R$ provided by the merchant. $\mathcal{B}$ checks the validity of $\Phi$ and the consistency with $S$. If $(S, \Pi)$ is not a valid coin, $\mathcal{B}$ rejects the deposit. Else, $\mathcal{B}$ computes, from $S$, $2^\ell$ serial numbers $\widetilde{S}_{k_1}, \ldots \widetilde{S}_{k_{2^\ell}}$ corresponding to the $2^{\ell+1}$ dead leaves of the sub-tree. This is done by applying several modular exponentiation functions to $S$, using the right generators. $\mathcal{B}$ has to deal with $2^\ell$ unitary coins $(\widetilde{S}_{k_j}, S, \Pi)$, $1 \leq j \leq 2^\ell$.

For every unitary coin $(\widetilde{S}_{k_j}, S, \Pi)$, $\mathcal{B}$ checks if there is already an entry $(\widetilde{S}_{k_j}, S', \Pi')$ in the database. If there is no entry in the database for the serial number $\widetilde{S}_{k_j}$, then $\mathcal{B}$ accepts the deposit of the coin $(\widetilde{S}_{k_j}, S, \Pi)$, credits the $pk_{\mathcal{M}}$'s account and add $(\widetilde{S}_{k_j}, S, \Pi)$ to the database of spent coins. Else, there is an entry $(\widetilde{S}_{k_j}, S', \Pi')$ in the database. Then, $\mathcal{B}$ checks the freshness of merchant randomness $R$ in $\Pi$ compared to $\Pi'$. If it not fresh, $\mathcal{M}$ is a cheat and $\mathcal{B}$ refused the deposit. If $R$ is fresh, $\mathcal{B}$ accepts the deposit of the coin $(\widetilde{S}_{k_j}, S, \Pi)$, credits the $pk_{\mathcal{M}}$'s account and add $(\widetilde{S}_{k_j}, S, \Pi, S', \Pi',)$ to the list of double-spenders. For every entry of the database of double-spenders, $\mathcal{B}$ will executes the `Identify` algorithm.

### 4.5 Identify

Assume that a double detection has been done. Then $\mathcal{B}$ knows two accepted spending $(2^{I_1}, S_1 = LK_1 \| RK_1, T_1, R_1, \Phi_1)$ with $I_1 = L - i_1$ and $(2^{I_2}, S_2 = LK_2 \| RK_2, T_2, R_2, \Phi_2)$ with $I_2 = L - i_2$ such that e.g. $S_1$ is an ancestor of $S_2$ or $S_1 = S_2$. If $S_1 = S_2$ then the bank can directly get the public key $pk_{\mathcal{U}}$ by computing $\left(T_1^{R_2}/T_2^{R_1}\right)^{1/(R_2-R_1)} = pk_{\mathcal{U}}$. If $S_1$ is an ancestor of $S_2$, then the bank computes the masking value $g_{I_2+1,2}^{V_2}$ (s.t. $T_2 = pk_{\mathcal{U}} g_{I_2+1,2}^{R_2 \cdot V_2}$) from the knowledge of $LK_1$ and $RK_1$ and the path[5] from $N_{i_1}^{j_1}$ up to $N_{i_2}^{j_2}$ as described in Figure 4. Then, $\mathcal{B}$ computes the public key

| |
|---|
| **Input:** $i_1, j_1, i_2, j_2$ |
| **Output:** $V_2$ |

```
CurrentNode ← N_{i1}^{j1}
If N_{i2}^{j2} belongs to leftSubTree(CurrentNode), then
  V2 ← LK1; CurrentNode ← leftSon(CurrentNode);
Else
  V2 ← RK1; CurrentNode ← rightSon(CurrentNode);
For k = i1 + 2 to i2 do
   If N_{i2}^{j2} belongs to leftSubTree(CurrentNode) , then
     V2 ← (g_{k,0})^{V2}; CurrentNode ← leftSon(CurrentNode)
   Else
     V2 ← (g_{k,1})^{V2}; CurrentNode ← rightSon(CurrentNode)
   k = k + 1
 return V2
```

**Fig. 4.** Identify protocol - Computation of $V_2$

$pk_{\mathcal{U}}$ as follows: $(T_2)^{\frac{1}{R_2}} / g_{I_2+1,2}^{V_2} = pk_{\mathcal{U}}$ .

### 4.6 Verify Guilt

The algorithm `VerifyGuilt` can be executed by any actor from the parameters of the system *Params* and a proof $\Pi_G$. One can parse the proof $\Pi_G$ as $\left((2^{\ell_1}, S_1, R_1, T_1, \Pi_1), (2^{\ell_2}, S_2, R_2, T_2, \Pi_2)\right)$ and next run `Identify` on these values. If the algorithm `Identify` returns a public key $pk_{\mathcal{U}}$, then one can check if $\Pi_1$ is consistent with $(2^{\ell_1}, S_1, R_1, T_1)$ and if $\Pi_2$ is consistent with $(2^{\ell_2}, S_2, R_2, T_2)$. If both are consistent then accept, else reject.

## 5 Security Arguments

In this section, we provide the Theorem that stipulates that the $\mathcal{DCS}$ scheme is a secure divisible e-cash system.

[5] The values $N_{i_1}^{j_1}$ and $N_{i_2}^{j_2}$ are not know by $\mathcal{B}$ but $\mathcal{B}$ knows the path from $N_{i_1}^{j_1}$ up to $N_{i_2}^{j_2}$ since it knows the path used to compute the colliding serial numbers.

**Theorem 1.** *In the random oracle model, the $\mathcal{DCS}$ scheme is secure:*

- *If the CL signature scheme is unforgeable, then $\mathcal{DCS}$ is unforgeable.*
- *Under the DDH assumption, $\mathcal{DCS}$ is unlinkable.*
- *If the CL signature scheme is unforgeable, then $\mathcal{DCS}$ permits the identification of double-spenders.*
- *Under the DL assumption (and the Flexible RSA assumption if $\mathcal{DCS}$ relies on the ACJT scheme), $\mathcal{DCS}$ has the exculpability property.*

*Proof.* We have to show that $\mathcal{DCS}$ verifies all security properties.

**Unforgeability.** We want to show that if an adversary $\mathcal{A}$ is able to break the unforgeability of our construction, then it is possible to break the unforgeability of the CL signature scheme under adaptive chosen message attack.

We can interact with $\mathcal{A}$ during the withdrawal protocol by playing the role of an honest bank with access to the signature oracle. After each successful spending executed by $\mathcal{A}$, we extract, using standard technique, the values $(u, s, r, \sigma)$ satisfying the relation embedded into the valid proof of knowledge $\Pi$. Since there are more spent coins than $\mathcal{A}$ can legitimely own, and since there is no detection of double-spending (by assumption), then it is necessary that, among all extracted values $(u_j, s_j, r_j, \sigma_j)$, one signature $\sigma$ on a message $m = (s, u, r)$ is unknown and does not come from the signature oracle. Thus, this one more signature is a signature (forgery) in the CL's scheme on the message $m = (u, s, r)$.

As the CL signature scheme is proven secure against adaptive chosen message attacks under the Flexible RSA assumption (if the scheme relies on the ACJT scheme) or the $q$-SDH (if the scheme relies on the BBS scheme), it follows that $\mathcal{A}$ cannot succeed with non negligible probability. Because our proof requires rewinding to extract $s'$ and $r$ from an adversary $\mathcal{A}$, our proof is valid only against sequential attacks. Indeed, in a concurrent setting where the attacker is allowed to interact with the bank in an arbitrarily interleaving manner, our machine may be forced to rewind an exponential number of times. This drawback can be overcome by using for instance well-know techniques [12] which would require from the user to encrypt $s'$ and $r$ in a verifiable manner [7].

**Unlinkability.** We want to show that if an adversary $\mathcal{A}$ is able to break the unlinkability of our construction, then it is possible to break an instance of the Diffie-Hellman problem. In fact, we use a variant of the

Diffie-Hellman problem, called Matching Multi Diffie-Hellman (MMDH) problem, and we prove in Appendix A that if someone is able to solve the MMDH problem, then it is possible to solve a given instance of the DDH problem.

We can interact with $\mathcal{A}$ during the withdraw protocol by playing the role of an honest user except for the two first interactions where we use the MMDH instance. During spending protocols, we can interact with $\mathcal{A}$ by playing the role on an honest user, except when the divisible coin corresponds to one of the two divisible coins associated with the MMDH instance to be solved.

We can win the game when $\mathcal{A}$ chooses the two first users (corresponding to the MMDH instance) and thus use the MMDH instance during the execution of the final spend. If $\mathcal{A}$ does not choose users $i_0$ and $i_1$ for the challenge we need to play again the game.

We denote by $q_U$ the average number of users created by $\mathcal{A}$. Our success probability is $\epsilon' = 1 - (1 - (1/2 + \epsilon/2))^{q_U} \equiv 1/2 + q_U\epsilon/2$ within polynomial $\mathcal{T}' = q_U\mathcal{T} + \tau$, where $\tau$ is polynomial.

*Remark 2.* In the simulation, we use the instance of the MMDH problem to interact with $\mathcal{A}$. We also need to choose a value for the bit $b$. If our choice of $b$ is correct, then there is no problem and we will be able to conclude with the advantage $\epsilon$ of $\mathcal{A}$. If this choice is uncorrect, $\mathcal{A}$ has a probability exactly equal to 1/2 as ours. Repeating the game many times, our success probability of solving the MMDH instance is greater than 1/2.

**Identification of Double-spenders.** We want to show that if an adversary $\mathcal{A}$ is able to break the identification of double-spenders property, then it is possible to break the unforgeability of the CL signature scheme.

We have access to a signature oracle taking as input a commitment and outputting a signature on committed values. We interact with $\mathcal{A}$ during withdrawal protocols by playing the role of an honest bank. We also interact with $\mathcal{A}$ during spending protocols playing the role of the merchant. Note that there is no honest users in the game. After each successful spending executed by $\mathcal{A}$, we extract the values $(u, s, r, \sigma)$ satisfying the relation embedded into the valid proof of knowledge $\Pi$. When there is a double-spending, i.e. $(\perp_1, S_1, \Pi_1), (S_2, \Pi_2)$, that means that there exist a valid serial number $\tilde{S}$ which can be computed from both $S_1$ and $S_2$. Furthermore, the proof $\Pi_1$ is consistent with $S_1$ and the proof $\Pi_2$ is consistent with $S_2$ and $R_1 \neq R_2$ where $R_1$ is the random chosen by the merchant in $\Pi_1$ and $R_2$ is the random chosen by the merchant in

$\Pi_2$. Both $\Pi_1$ and $\Pi_2$ contains a proof of knowledge of a signature of the bank on the master serial number seed $s$ used to generate $S_1$, $S_2$ and $\tilde{S}$. Thus, these two signatures $\sigma_1$ and $\sigma_2$ are such that at least one of the two is different from the signatures obtained during the execution of the `Withdrawal` protocols submitted to the signature oracle. This signature ($\sigma_1$ or $\sigma_2$) is thus a forgery on CL signature scheme. As the CL signature scheme is proven secure against adaptive chosen message attacks, it follows that $\mathcal{A}$ cannot succeed with non-negligible probability.

**Exculpability.** The adversary $\mathcal{A}$ wins the game if he can falsely accuse an honest user of a double-spending. This means that the adversary can interact with honest users to obtain spending from them and he wins if he can produce one spend $(S', T', \Pi')$ related to a valid one $(S', T', \Pi')$ and such that the output of $\texttt{Identify}((S, T, \Pi), (S', T', \Pi'))$ is a public key $pk_{\mathcal{U}}$ of a honest user (with non negligible probability).

The security proof of the exculpability involves forking lemma-like technique for an attacker that exploits both valid spending played by honest users and valid withdrawals played by honest users when the extractability of the RO proofs-of-knowledge relies on the DL assumption in order to falsely accuse an honest user. If the Camenisch-Lysysanskaya scheme of the withdrawal protocol uses a group of unknown order, then the exculpability relies on both the DL assumption for an attacker that exploits valid spendings played by honest users in order to falsely accuse an honest user, and on the factorization assumption to ensure the non-malleability and the soundness of the proof of knowledge $\Phi$ (see [3]).

## 6 Conclusion

In this paper, we present the first off-line divisible e-cash scheme that provides strong unlinkability and truly anonymity. We introduced the idea of using a security tag in a divisible e-cash scheme. The anonymity of users is achieved without impacting the performance of the spending protocol and without using a trusted third party. The spending protocol exploits the binary structure underlying the divisible coin in order to get an efficient spending protocol. However, even if the new scheme permits the spending of multiple coins at a time, it uses double-exponentiation proofs for the spending phase which is still a little expensive. Thus, for a small number of coins at a time, the spending is still expensive. Another possible improvement for the scheme could be to find a method to detect double spending without computing $2^L$ serial numbers for a divisible coin of monetary value $2^L$.

## Acknowledgements

We are grateful to Pascal Paillier and Jacques Traoré for their suggestions of improvement, and to Serge Fehr and anonymous referees for their valuable comments. We also wish to mention that a similar work has been independently done by Jan Camenisch, Markulf Kohlweiss, Anna Lysyanskaya and Maria Meyerovich.

## References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-resistant Group Signature Scheme. Advances in Cryptology - Crypto'00, volume 1880 of LNCS, pages 255-270, 2000.
2. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures using Strong Diffie Hellman. Advances in Cryptology - Crypto'04, volume 3152 of LNCS, pages 41-55, 2004.
3. F. Boudot and J. Traoré. Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. ICISC'99, volume 1726 of LNCS, pages 87-102, 1999.
4. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-cash. Advances in Cryptology - Eurocrypt'05, volume 3494 of LNCS, pages 302-321, 2005.
5. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. Advances in Cryptology - Crypto'04, volume 3152 of LNCS, pages 56-72, 2004.
6. J. Camenisch and M. Michels. Proving in Zero-knowledge that a Number is the Product of Two Safe Primes. Advances in Cryptology - Eurocrypt'99, volume 1592 of LNCS, pages 107-122, 1999.
7. J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In D. Boneh, editor, Advances in Cryptology - Crypto '03, volume 2729 of LNCS, pages 126-144. Springer, 2003.
8. S. Canard, A. Gouget, and E. Hufschmitt. A Handy Multi-coupon System. Applied Cryptography and Network Security - ACNS 2006, volume 3989 of LNCS, pages 66-81, 2006.
9. A.H. Chan, Y. Frankel, and Y. Tsiounis. Easy Come - Easy Go Divisible Cash. Advances in Cryptology - Eurocrypt'98, volume 1403 of LNCS, pages 561-575, 1998.
10. D. Chaum and T. Pedersen. Transferred Cash Grows in Size. Advances in Cryptology - Eurocrypt'92, volume 658 of LNCS, pages 390-407, 1993.
11. R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. Advances in Cryptology - Crypto'94, volume 839 of LNCS, pages 174-187, 1994.
12. I. Damgard. Efficient Concurrent Zero-knowledge in the Auxiliary String Model. Advances in Cryptology - Eurocrypt '00, volume 1807 of LNCS, pages 418-430, 2000.

13. S. D'Amingo, and G. Di Crescenzo. Methodology for Digital Money based on General Cryptographic Tools. Advances in Cryptology - Eurocrypt'94, volume 950 of LNCS, pages 156-170, 1994.
14. T. Eng, and T. Okamoto. Single-term Divisible Coins. Advances in Cryptology - Eurocrypt'94, volume 950 of LNCS, pages 306-319, 1994.
15. E. Fujisaki and T. Okamoto. Statistical Zero-knowledge Protocols to Prove Modular Polynomial Relations. Advances in Cryptology - Crypto'97, volume 1294 of LNCS, pages 16-30, 1997.
16. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Advances in Cryptology - Crypto'86, volume 263 of LNCS, pages 186-194, 1986.
17. M. Girault, G. Poupard and J. Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. Advances in Cryptology - Journal of Cryptology, Volume 19, Number 4. Pages 463-487, Springer-Verlag, 2006.
18. H. Handschuh, Y. Tsiounis, and M. Yung. Decision Oracles are Equivalent to Matching Oracles. Public Key Cryptography PKC '99, volume 1560 of LNCS, pages 276-289. Springer, 1999.
19. T. Nakanishi, M. Shiota, and Y. Sugiyama. An Unlinkable Divisible Electronic Cash with User's Less Computations using Active Trustees. ISITA 2002, 2002.
20. T. Nakanishi and Y. Sugiyama. Unlinkable Divisible Electronic Cash. ISW'00, pages 121-134, 2000.
21. T. Okamoto. An Efficient Divisible Electronic Cash Scheme. Advances in Cryptology - Crypto'95, volume 963 of LNCS, pages 438-451, 1995.
22. T. Okamoto, K. Ohta. Universal Electronic Cash. Advances in Cryptology - Crypto'91, volume 576 of LNCS, pages 324-337, 1992.
23. J.C. Pailles. New Protocols for Electronic Money. Advances in Cryptology - Asiacrypt'92, volume 718 of LNCS, pages 263-274, 1993.
24. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology, Volume 13 - Number 3. Pages 361-396, Springer-Verlag, 2000.
25. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On Monotone Formula Closure of SZK. FOCS 1994, pages 454-465, 1994.
26. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. Advances in Cryptology - Crypto'89, volume 435 of LNCS, pages 239-252, 1990.
27. M. Stadler. Publicly Verifiable Secret Sharing. Advances in Cryptology - Crypto'96, volume 1070 of LNCS, pages 190-199, 1996.
28. M. Trolin. A stronger definition for anonymous electronic cash. Cryptology ePrint Archive: Report 2006/241. 2006.


## A Matching Multi Diffie-Hellman problem

The problem underlying the property of unlinkability for $\mathcal{DCS}$ is the *Matching Multi Diffie-Hellman problem (MMDH)*. We show that MMDH can be used to solve the *Decisional Diffie-Hellman problem (DDH)*.

**Decisional Diffie-Hellman (DDH) problem:** given a random generator $g \in \mathcal{G}$ where $\mathcal{G}$ has prime order and the values $h^x, h^y, h^z$, the problem consists in deciding if $xy = z$ or not.

**Matching Multi Diffie-Hellman (MMDH) problem:** let $\mathcal{H}$, $\mathcal{H}_1$ and $\mathcal{H}_2$ be groups of prime order such that $\mathcal{H}_1$ is a subgroup of $\mathbb{Z}_o^*$ where $o$ is the order of $\mathcal{H}_2$. Given three random generators $h \in \mathcal{H}$, $h_1 \in \mathcal{H}_1$ and $h_2 \in \mathcal{H}_2$ and the values $h^{\alpha_0}, h^{\alpha_1}, h_2^{h_1^{\alpha_b}}$ and $h_2^{h_1^{\alpha_{\bar{b}}}}$ where $b \in \{0, 1\}$, the problem consists in deciding if $b = 0$ or 1.

**Decisional Multi Diffie-Hellman (DMDH) problem:** let $\mathcal{H}$, $\mathcal{H}_1$ and $\mathcal{H}_2$ be groups of prime order such that $\mathcal{H}_1$ is a subgroup of $\mathbb{Z}_o^*$ where $o$ is the order of $\mathcal{H}_2$. Given three random generators $h \in \mathcal{H}$, $h_1 \in \mathcal{H}_1$ and $h_2 \in \mathcal{H}_2$ and the values $h^\alpha, h_2^{h_1^\beta}$, the problem consists in deciding if $\alpha = \beta$ or not.

**Derived Decisional Diffie-Hellman (DDDH) problem:** given random generators $g_1, g_2 \in \mathcal{G}$ where $\mathcal{G}$ has prime order and the values $g_1^a, g_2^b$, the problem consists in deciding if $a = b$ or not.

The problem MMDH is at least as difficult as DMDH. In fact, the MMDH is the matching problem related to the decisional one DMDH. Therefore, Handschuh, Tsiounis and Yung show [18] that decision oracles are equivalent to matching oracles, which can be applied to our context.

The problem DMDH is at least as difficult as DDDH. Indeed, given an instance $(g_1, g_2, g_1^a, g_2^b)$ of the DDDH problem, we can transform it into an instance $(h = g_1, h_1, h_2 = g_2, h^\alpha = g_1^a, h_1^{h_2^\beta} = h_1^{g_2^b})$ where $h_1$ is taken at random, of the DMDH problem. Thus, $a = b$ if and only if $\alpha = \beta$.

The problem DDDH is at least as difficult as DDH. Indeed, given an instance $(g, g^x, g^y, g^z)$ of the DDH problem, we can transform is into an instance $(g_1 = g, g_2 = g^x, g_1 = g^x, g_2 = g^z)$ of the DDDH problem. Thus, we have $z = xy$ if and only if $a = b$.

We deduce that MMDH is at least as difficult as DDH.