Achieving privacy and accountability in traceable digital currency

Amira Barki[1] and Aline Gouget[1]

Thales DIS, Meudon, France

Abstract. Several Central Bank Digital Currency (CBDC) projects are considering the development of a digital currency that is managed on a *permissioned* blockchain, i.e. only authorized entities are involved in transactions verification. In this paper, we explore the best possible balance between *privacy* and *accountability* in such a traceable digital currency. Indeed, in case of suspicion of fraud or money laundering activity, it is important to enable the retrieval of the identity of a payer or a payee involved in a specific transaction. Based on a preliminary analysis of achievable anonymity properties in a transferable, divisible and traceable digital currency systems, we first present a digital currency framework along with the corresponding security and privacy model. Then, we propose a pairing-free traceable digital currency system that reconciles user's privacy protection and accountability. Our system is proven secure in the random oracle model.

Keywords: Digital currency · Privacy protection · Blockchain · Traceability · Divisibility · Accountability · non-interactive ZKPKs

1 Introduction

Central Bank Digital Currency projects[1] and Facebook Libra are actively considering the possibility to issue and manage digital currencies through the use of a *permisionned* blockchain, i.e. only authorized entities can verify the transactions in a decentralized manner whereas the digital currency is issued by one or several digital currency issuers. Such a digital currency fosters transparency by enabling the *traceability* of a specific digital currency unit. This property is useful for identifying suspicious activities through the use of artificial intelligence algorithms for instance. However, traceability does not provide on its own a way to retrieve the identity of a transaction payer or payee.

Let us assume that the digital currency system requires all the involved users to be clearly identified during a transaction. This would enable the identification of the payer and/or payee in the case of fraud or money laundering suspicion. However, this would also strongly jeopardize end-users' privacy.

[1] https://publications.banque-france.fr/sites/default/files/medias/documents/wp-732.pdf; https://consensys.net/blog/enterprise-blockchain/everything-you-need-to-know-about-central-bank-digital-currencies/

In this paper, we investigate to what extent users' privacy can be preserved while providing *accountability* in traceable digital currency systems. Thereby, in case of fraud or money laundering suspicious activity, it is still possible to identify the transaction payer and/or payee. To do so, we rely on a *Revocation Authority* that is outside of the blockchain consortium. Moreover, we consider that the following properties are essential in a practical digital currency system: (i) *transferability*, which allows a transaction payee to re-transfer received digital currency without the involvement of a digital currency issuer; (ii) *divisibility*, enabling the owner of a digital currency to split its amount in order to transfer only part of the digital currency to another user; and (iii) *traceability*, allowing to link transactions associated to the same issued digital currency.

Anonymity properties in a transferable, divisible and traceable digital currency system. Anonymity properties of transferable electronic cash (eCash) that are not based on blockchain technology have already been well studied. Canard and Gouget [5] formally define four[2] levels of anonymity for transferable eCash systems, namely *weak anonymity* (WA), *strong anonymity* (SA), *full anonymity* (FA) and *perfect anonymity* (PA) such that PA $\rightarrow$ FA $\rightarrow$ SA $\rightarrow$ WA. WA is fulfilled if an adversary $\mathcal{A}$ is unable to link a payment to a withdrawal. SA is fulfilled if WA is satisfied and $\mathcal{A}$ cannot figure out whether two payments were performed by the same user or not. FA is fulfilled if SA is satisfied and $\mathcal{A}$ is unable to recognize a digital currency that was used during previous payments between two honest users. PA is fulfilled if FA is satisfied and $\mathcal{A}$ is unable to figure out whether he has already owned the digital currency he is receiving, or not. It has been shown that a transferable eCash system cannot achieve *perfect anonymity* (PA) either against unbounded [7] or bounded [5] adversaries.

To the best of our knowledge, the anonymity properties of a digital currency system that is both transferable and traceable have not been formally studied. For such system, both FA and PA anonymity properties no longer make sense as, by design, anyone should be able to recognize a digital currency that was used during previous payments. Besides, any observer is always able to tell that the same anonymous user was involved in two subsequent payment transactions associated to the same digital currency, first as a payee then as a payer. Thus, the highest level of anonymity that a transferable and traceable digital currency system may achieve is a slightly weaker variant of *strong anonymity* (SA) since an adversary is able to link some transactions.

Also, for practical reasons, it is usually desirable that the digital currency be divisible in order to support payments of flexible amounts. If we consider a transferable and traceable digital currency system that is also divisible, then some additional payment transactions can be linked to the same user. Indeed, in such a system, the user will perform two concurrent transactions at each transfer: (1) a real transfer transaction worth the amount to be transferred to a given payee, and (2) an auto-transfer transaction of the remaining amount to himself. Therefore, any observer of such system is able to tell that two concurrent

[2] [5] and [3] define other anonymity notions unachievable by traceable digital currency.

transactions associated to the same previous transaction $T_{t,j}$ are performed by the same user, who is the payee of the previous transaction $T_{t,j}$. Moreover, the payee of a real transfer transaction is also able to tell that the two subsequent and concurrent transactions that follow the auto-transfer transaction were carried out by the same user who performed the real and auto transfers.

Thus, a transferable digital currency system that provides the traceability and divisibility properties cannot achieve *strong anonymity* since an adversary may leverage transactions' history to link some particular transactions to the same user. To our knowledge, the anonymity property that such system can achieve has not been formally studied in the state-of-the-art on digital currency.

Our Contribution. Security models of state-of-the-art divisible and transferable electronic cash systems did not consider the traceability feature which is defined as the ability to trace, in an anonymous way, an issued digital currency from its first transfer during a payment until its deposit at the digital currency issuer. Such traceability feature may be particularly interesting for the detection of suspicious transactions even though transactions are anonymous and unlinkable, *i.e.* one cannot decide whether two transactions were performed by the same user or not. In this work, we target the highest level of anonymity and unlinkability that a transferable, traceable and divisible digital currency system may achieve while providing accountability. That is, a slightly weaker variant of the strong anonymity property as few payments may be linked to the same user, which we refer to as *end-user's privacy*. First, we define a digital currency framework along with a security and privacy model. Then, we propose a digital currency system that achieves the expected properties of transferability, divisibility, traceability and accountability whilst solely relying on conventional cryptographic primitives, *e.g.* no pairing computations are required.

Organization. In Section 2, we introduce the cryptographic tools required to build our Digital Currency System ($\mathcal{DCS}$). Then, in Section 3, we present the framework for a $\mathcal{DCS}$. In Section 4, we describe the security and privacy model. Next, in Section 5, we detail our $\mathcal{DCS}$ construction. Due to lack of space, the description of the adaptation of ACL scheme [4] that we use to build our $\mathcal{DCS}$ system is provided in Appendix A, and the security analysis of our $\mathcal{DCS}$ system is provided in Appendix B.

2 Preliminaries

In this paper, we use the notation $r \in_R R$ to state that a value r is chosen uniformly at random from the set R, and $\{c_i\}_{i=1}^n$ denotes the set $\{c_1, c_2, \ldots, c_n\}$.

2.1 Computational hardness assumptions

Discrete Logarithm (DL) assumption. Let $\mathbb{G}$ be a cyclic group of prime order q and g be a generator of $\mathbb{G}$, the Discrete Logarithm assumption states that, given $y \in \mathbb{G}$, it is hard to find the integer $x \in \mathbb{Z}_q$ such that $y = g^x$.

Decisional Diffie Hellman (DDH) assumption. Let $\mathbb{G}$ be a cyclic group of prime order q, the Decisional Diffie-Hellman assumption states that, given a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{ab}$ or not.

In the sequel, $\mathbb{G}$ denotes a cyclic group of prime order q where the DDH problem is assumed to be hard, and g_E and g_I are random generators of $\mathbb{G}$.

2.2 Digital signature scheme

A digital signature scheme is a triplet of algorithms $(\texttt{SKeyGen}, \texttt{SSign}, \texttt{SVerif})$ where `SKeygen` takes as input the system parameters and outputs a signature key pair (sk, pk), `SSign` takes as input a secret signature key sk and a message m, and outputs a signature $\sigma = \texttt{SSign}[sk](m)$, and `SVerif` takes as input a public signature key pk, a message m and a signature σ and outputs 1 or 0. In the sequel, we assume the existence of a secure digital signature scheme `S`.

2.3 ElGamal encryption scheme

ElGamal cryptosystem [9] is a the triplet of algorithms $(\texttt{EGKeyGen}, \texttt{EGEnc}, \texttt{EGDec})$ where `EGKeyGen` takes as input the system parameters and outputs an encryption key pair (sk_E, pk_E) with $sk_E \in \mathbb{Z}_q^*$ and $pk_E = g_E^{sk_E}$, `EGEnc` takes as input a message $m \in \mathbb{G}$ and pk_E, and outputs the pair $(e_1 = g_E^r, e_2 = m \cdot pk_E^r)$ with $r \in_R \mathbb{Z}_q^*$, and `EGDec` takes as input (e_1, e_2) and sk_E, and outputs $m = e_2/e_1^{sk_E}$.

ElGamal cryptosystem is secure under the DDH assumption, and it has a threshold variant known as *threshold ElGamal* [11].

2.4 End-user's identity and related key pair

The *identity* of an end-user is defined as an identifier Id_i associated to an official document, *e.g.* a passport, enabling to verify user's identity. An *identity key pair* consists of the pair $(u_i, g_I^{u_i})$ where the secret $u_i \in \mathbb{Z}_q^*$ is selected at random.

2.5 Zero Knowledge Proof of Knowledge

A Zero Knowledge Proof of Knowledge (ZKPK) is an interactive protocol that allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that he knows some secrets verifying some statement(s) without revealing anything else. It is denoted by $\pi = PoK\{\alpha, \beta : \textit{ statements about } \alpha, \beta\}$ where Greek letters refer to $\mathcal{P}$'s knowledge.

A ZKPK must ensure three security properties, namely (1) **completeness**: a valid prover is accepted with overwhelming probability; (2) **soundness**: a cheating prover is rejected with overwhelming probability; and (3) **zero-knowledge**: no information about the secrets is revealed beyond the fact that the statements are true. Interactive ZKPKs can be transformed into non-interactive ZKPKs, denoted by NIZK, using the Fiat-Shamir technique [8].

In our digital currency system (c.f. Section 5), we use the following ZKPKs[3] :

[3] In addition to the ZKPKs of the adapted ACL scheme detailed in Appendix A.

- a NIZK proof of Discrete Logarithm (DL) knowledge $\pi_1 = PoK\{\alpha : x = g_I^\alpha\}$ where the identity public key x is revealed;
- a NIZK proof of equality of secrets, DL and representation knowledge combined with an OR proof $\pi_2 = PoK\{\alpha, \beta, \gamma, \delta : e_1 = g_E^\alpha \wedge e_2 = g_I^\beta \cdot pk_E^\alpha \wedge e_2 \cdot C^{-1} = pk_E^\gamma \wedge pk_E^\delta = C \cdot g_I^{-u_1} \vee \cdots \vee pk_E^\delta = C \cdot g_I^{-u_n}\}$ where an ElGamal ciphertext $(e_1, e_2) = (g_E^{r_1}, g_I^{u_i} \cdot pk_E^{r_1})$ and a value $C = g_I^{u_i} pk_E^{r_2}$ are revealed;
- a NIZK proof of equality of DLs $\pi_3 = PoK\{\alpha : e_2 \cdot g_I^{-u_i} = e_1^\alpha \wedge pk_E = g_E^\alpha\}$ where an ElGamal ciphertext $(e_1, e_2) = (g_E^{r_1}, g_I^{u_i} \cdot pk_E^{r_1})$ and $g_I^{u_i}$ are revealed.

2.6 (Partially) blind signatures

Blind signature schemes, first introduced by Chaum [6], are a variant of digital signature schemes that allow a receiver to get a signature without giving the signer any information about the actual message m or the resulting signature σ.

In digital currency use cases, the signer usually needs to add some common information to the blind signature such as an amount. To this end, Abe *et al.* [1] introduced an extension of blind signatures referred to as *partially blind signatures*, which enable the receiver and the signer to agree on a common information *info* to be added in the blind signature of a message m. More recently, Baldimtsi and Lysyanskaya [4] proposed an efficient and provably secure construction of blind signatures with attributes. It is inspired from Abe's blind signature scheme [2] in which blinding preserves some structural elements into which attributes can be embedded. If one of the attributes is known to both the receiver and the signer, then their scheme can be seen as a partially blind signature where the common information is the known attribute. Their scheme is quite efficient and practical as it only requires few exponentiations in a prime-order group in which the DDH problem is hard. Based on it, they built a single-use anonymous credential system called *Anonymous Credential Light (ACL)* [4] that is both provably secure and does not require the use of expensive bilinear pairings. The ACL scheme can also be used as eCash with attributes [10].

In Appendix A, we describe an adaptation of ACL attribute-based blind signature scheme that we use to issue digital currencies in our Digital Currency System $\mathcal{DCS}$. Indeed, to build our $\mathcal{DCS}$ system, we need a (partially) blind signature with two specific attributes: (1) an attribute that is known by both the signer and the user and which corresponds to the amount of digital currency withdrawn by the user; and (2) an attribute that is jointly computed by both the user and the signer but whose value is only known to the user, and which corresponds to the digital currency serial number `sn`. As for the message m, which is only known to the user, it corresponds to a user's public key $pk_{t,0}$. Both m and `sn` need to be kept secret from the signer to ensure the blindness property.

3 Digital currency framework

In this section, we first introduce the entities involved in the digital currency system. Then, we define the algorithms and protocols that the system relies on.

3.1 Entities

There are n digital currency issuers $\mathcal{CI}_1, \ldots, \mathcal{CI}_n$ forming the initial consortium. They are responsible for setting up and maintaining a permissioned distributed ledger used to share information about all transactions related to issued digital currencies. An external certification authority $\mathcal{P}_{CA}$ is selected by the consortium.

End-users with a banking account, called *banked users*, can withdraw digital currencies from $\mathcal{CI}_1, \ldots, \mathcal{CI}_n$, transfer/receive digital currencies to/from banked or unbanked users, and deposit owned digital currencies to $\mathcal{CI}_1, \ldots, \mathcal{CI}_n$. Unbanked users can only receive digital currencies from other users and re-transfer the received digital currencies. Banked and unbanked users are denoted by $\mathcal{U}_1, \ldots \mathcal{U}_m$ and we specify if it is a banked or unbanked user when needed.

In order to support transactions' anonymity revocation, the system involves two entities: an ID registration authority denoted by $\mathcal{IDR}$ and a revocation authority denoted by $\mathcal{RA}$. To prevent a malicious $\mathcal{IDR}$ or $\mathcal{RA}$ from illegitimately retrieving the identity of the users involved in a given transaction, both $\mathcal{IDR}$ and $\mathcal{RA}$ will not be part of the distributed ledger and $\mathcal{RA}$ only receives the transactions whose anonymity need to be lifted.

3.2 System framework

Let λ be a security parameter. A Digital Currency System $\mathcal{DCS}$ can be modeled using the following set of algorithms and protocols:

- $\texttt{ParamGen}(1^\lambda)$ is a probabilistic algorithms that outputs the system parameters sp.
- $\texttt{SKeyGen}(sp)$, $\texttt{ACLKeyGen}(sp)$ and $\texttt{EGKeyGen}(sp)$ are probabilistic algorithm that output a signature key pair.
- $\texttt{RequestCertificate}\big(*, \mathcal{P}_{CA}\big)$ is an interactive protocol between a requester holding a private/public key pair and the certification authority $\mathcal{P}_{CA}$ which either outputs a certificate on the public key or the output is $\perp$.
- $\texttt{IDKeyGen}(sp)$ is a probabilistic algorithm that outputs an identity key pair $(u_i, g_I^{u_i})$.
- $\texttt{IDRegister}(\mathcal{U}, \mathcal{IDR})$ is an interactive protocol between a user $\mathcal{U}$ and the ID registration authority $\mathcal{IDR}$ where $\mathcal{U}$ either receives an identity attestation $\texttt{Att}(Id_i, g_I^{u_i})$ or the output is $\perp$.
- $\texttt{DBUpdate}(\mathcal{U}, \mathcal{RA})$ is an interactive protocol between a user $\mathcal{U}$ and the revocation authority $\mathcal{RA}$ where both databases $\mathcal{DB}_{Priv}$ and $\mathcal{DB}_{Pub}$ are updated by adding respectively $\{Id_i, g_I^{u_i}, \texttt{Att}(Id_i, g_I^{u_i})\}$ and $\{g_I^{u_i}\}$, or the output is $\perp$.
- $\texttt{Withdrawal}(\mathcal{U}, \mathcal{CI}_i)$ is an interactive protocol between a banked user $\mathcal{U}_i$ and a digital currency issuer $\mathcal{CI}_i$ where $\mathcal{CI}_i$ computes a partially blind signature on (i) a user public key $pk_{t,0}$, (ii) a unique identifier $\texttt{sn}_t$, and (iii) $\texttt{amount}_t$ such that it does not know the values of both $\texttt{sn}_t$ and $pk_{t,0}$. The protocol either outputs a new digital currency DC_t or $\perp$.
- $\texttt{Insert}(\mathcal{U})$ is a protocol executed by a banked user $\mathcal{U}_i$ to insert a digital currency DC_t into the ledger. $\mathcal{U}_i$ computes an encryption of $g_I^{u_i}$ using $pk_{\mathcal{RA}}$

to get $E_{t,1}$ and builds a related NIZK proof $\pi_{2,t,1}$ that $E_{t,1}$ was correctly computed and that he knows u_i such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$. Both $E_{t,1}$ and $\pi_{2,t,1}$ are included in the transaction $T_{t,0}$. The protocol either outputs $T_{t,0}$ or $\perp$.

- $\texttt{Transfer}(\mathcal{U}_i, \mathcal{U}_r)$ is an interactive protocol between a user $\mathcal{U}_i$ currently owning the digital currency of a transaction $T_{t,j-1}$ with $j \geq 1$ and a receiver $\mathcal{U}_r$. Both $\mathcal{U}_i$ and $\mathcal{U}_r$ agree on the transfer amount $\texttt{amount}_{t,j}$ with $0 < \texttt{amount}_{t,j} \leq \texttt{amount}_{t,j-1}$. The transfer protocol requires two transactions:
  - A transaction $T_{t,j}$ from $\mathcal{U}_i$ to $\mathcal{U}_r$ of amount $\texttt{amount}_{t,j}$ that includes the encryption $E_{t,j+1}$ of $g_I^{u_r}$ using $pk_{\mathcal{RA}}$ along with a NIZK proof $\pi_{2,t,j+1}$.
  - A transaction $T^*_{t,j}$ from $\mathcal{U}_i$ to $\mathcal{U}_i$ of the remaining amount that includes the encryption $E^*_{t,j+1}$ of $g_I^{u_i}$ using $pk_{\mathcal{RA}}$ along with a NIZK proof $\pi^*_{2,t,j+1}$.

  Either both the real transfer $T_{t,j}$ and the auto-transfer $T^*_{t,j}$ transactions are inserted into the distributed ledger in a random order or the output is $\perp$.
- $\texttt{Deposit}(\mathcal{U}_i, \mathcal{CI}_i)$ is an interactive protocol between a banked user $\mathcal{U}_i$ owning a digital currency related to a validated transaction $T_{t,j-1}$ with $j \geq 1$ and a digital currency issuer $\mathcal{CI}_i$. It either outputs the deposit transaction $T_{t,j}$ and the banking account of $\mathcal{U}_i$ is updated with $\texttt{amount}_{t,j-1}$, or $\perp$.
- $\texttt{IDRetrieve}(\mathcal{RA})$ is a deterministic algorithm performed by the revocation authority $\mathcal{RA}$. Given $T_{t,j}$, $\mathcal{DB}_{Priv}$ and $\mathcal{RA}$'s private key as input, it outputs $\{Id_i, g_I^{u_i}, \texttt{Att}(Id_i, g_I^{u_i})\}$ along with a proof $\pi_{3,t,j+1}$ proving this claim, or $\perp$.

4 Security and Privacy model

In this section, we first formalize the global variables and oracles before formally defining the security properties through a set of experiments involving a challenger $\mathcal{C}$ and a probabilistic polynomial time (PPT) adversary $\mathcal{A}$.

4.1 Global variables

The set of honest users whose private keys and secret u_i are unknown to $\mathcal{A}$ is denoted by $\mathcal{HU}$, and the set of corrupted users is denoted by $\mathcal{CU}$.

$\mathcal{DL}$ denotes the set of transactions included in the distributed ledger. For every user $\mathcal{U}_i$, we maintain a specific set $\mathcal{TS}_{\mathcal{U}_i}$ of the transactions involving $\mathcal{U}_i$. We also maintain a set $\mathcal{TS}_{\mathcal{CU}}$ including all transactions involving a corrupted payee. $\mathcal{RT}$ denotes the set of transactions whose anonymity was lifted.

Two databases $\mathcal{DB}_{Priv}$ and $\mathcal{DB}_{Pub}$ that respectively contain $\{Id_i, g_I^{u_i}, \texttt{Att}(Id_i, g_I^{u_i})\}$ and $\{g_I^{u_i}\}$ entries are maintained. $\mathcal{DB}_{Priv}$ is only known by the revocation authority $\mathcal{RA}$ whilst $\mathcal{DB}_{Pub}$ is known by $\mathcal{RA}$ and digital currency issuers $\mathcal{CI}_i$.

4.2 Oracles

We model $\mathcal{A}$'s capabilities by giving it access to several oracles defined as follows, where $\perp$ denotes a random choice of a user or digital currency unit.

Entities creation and end-users corruption. The oracle $\mathcal{O}\texttt{SKeyGen}(\mathcal{P}_{CA})$ creates a certification authority by generating a signature key pair $(sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}})$ which is the root of trust. The oracle $\mathcal{O}\texttt{SKeyGen}(\mathcal{IDR})$ creates an ID registration authority by generating a signature key pair $(sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}})$ and requesting the associated certificate. The oracle $\mathcal{O}\texttt{SKeyGen}(\mathcal{RA})$ creates a revocation authority by generating an ElGamal key pair $(sk^{eg}_{\mathcal{RA}}, pk^{eg}_{\mathcal{RA}})$ and requesting the associated certificate. The oracle $\mathcal{O}\texttt{SKeyGen}(\mathcal{CI}_i)$ creates a new honest digital currency issuer $\mathcal{CI}_i$ by generating (i) an ACL [4] signature key pair $(sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i})$, (ii) a signature key pair $(sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i})$, and (iii) requesting their certification.

The oracle $\mathcal{O}\texttt{IDKeyGen}(\mathcal{U}_i)$ creates a new *honest* user $\mathcal{U}_i$, generates his identity key pair and executes the identity verification as well as the registration process at the revocation authority. The honest user $\mathcal{U}_i$ is added to the set $\mathcal{HU}$. The oracle $\mathcal{O}\texttt{UCorrupt}(\mathcal{U}_i)$ corrupts the user $\mathcal{U}_i$. Upon its execution, $\mathcal{A}$ gets the secret u_i and private keys of user $\mathcal{U}_i$. Concurrently, $\mathcal{U}_i$ is moved from $\mathcal{HU}$ to $\mathcal{CU}$.

Withdrawal protocol. The oracle $\mathcal{O}\texttt{Withdrawal}_{\mathcal{U}}(\mathcal{U}_i)$ executes the user's side of the `Withdrawal` protocol whereas $\mathcal{A}$ is acting as a digital currency issuer $\mathcal{CI}_i$.

The oracle $\mathcal{O}\texttt{Withdrawal}_{\mathcal{CI}}(\mathcal{CI}_i)$ executes $\mathcal{CI}$'s side of the `Withdrawal` protocol. It is used by $\mathcal{A}$ acting as a corrupted user $\mathcal{U}_i$.

The oracle $\mathcal{O}\texttt{Withdrawal}_{\mathcal{U}\&\mathcal{CI}}(\mathcal{U}_i, \mathcal{CI}_i)$ executes both sides of the `Withdrawal` protocol. It is used by $\mathcal{A}$ to simulate the protocol execution between an honest $\mathcal{U}_i$ and an honest $\mathcal{CI}_i$. The issued digital currency DC_t will not be known to $\mathcal{A}$.

Transfer protocol. The oracle $\mathcal{O}\texttt{Transfer}_S$ executes the sender's side of the `Transfer`[4] protocol with $\mathcal{A}$ acting as the receiver. $\mathcal{A}$ can only call it with either $(\mathcal{U}_i, \perp)$ or $(\perp, DC_t)$ as argument. Anyway, the sender randomly selects the amount to be transferred. The resulting transfer transaction $T_{t,j}$ is appended to the sets $\mathcal{TS}_{\mathcal{CU}}$ and $\mathcal{TS}_{\mathcal{U}_i}$.

The oracle $\mathcal{O}\texttt{Transfer}_R(\mathcal{U}_i)$ executes the receiver's side of the `Transfer` protocol. The resulting transaction $T_{t,j}$ is appended to the set $\mathcal{TS}_{\mathcal{U}_i}$.

The oracle $\mathcal{O}\texttt{Transfer}_{S\&R}$ executes both sides of the `Transfer` protocol. The adversary $\mathcal{A}$ can call it with either $(\mathcal{U}_i, \mathcal{U}_j, \perp)$, $(\mathcal{U}_i, \perp, \perp)$, $(\perp, \mathcal{U}_j, \perp)$, $(\perp, \mathcal{U}_j, DC_t)$ or $(\perp, \perp, DC_t)$ as argument. The resulting transaction $T_{t,j}$ is appended to the sets $\mathcal{TS}_{\mathcal{U}_i}$ and $\mathcal{TS}_{\mathcal{U}_j}$.

Deposit protocol. The oracle $\mathcal{O}\texttt{Deposit}_{\mathcal{U}}$ executes the user's side of the `Deposit` protocol. $\mathcal{A}$ can only call it with either $(\mathcal{U}_i, \perp)$ or $(\perp, DC_t)$ as argument. The resulting transaction $T_{t,j}$ is appended to the set $\mathcal{TS}_{\mathcal{U}_i}$.

The oracle $\mathcal{O}\texttt{Deposit}_{\mathcal{CI}}(\mathcal{CI}_i)$ executes $\mathcal{CI}$'s side of the `Deposit` protocol. It is used by $\mathcal{A}$ acting as a malicious user.

The oracle $\mathcal{O}\texttt{Deposit}_{\mathcal{U}\&\mathcal{CI}}$ executes both sides of the `Deposit` protocol. $\mathcal{A}$ can call it with either $(\mathcal{U}_i, \mathcal{CI}_i, \perp)$, $(\mathcal{U}_i, \perp, \perp)$, $(\perp, \mathcal{CI}_i, \perp)$, $(\perp, \mathcal{CI}_i, DC_t)$ or $(\perp, \perp, DC_t)$ as argument. The transaction $T_{t,j}$ is appended to the set $\mathcal{TS}_{\mathcal{U}_i}$.

[4] For the first transfer of a withdrawn digital currency, this oracle executes both the `Insert` and `Transfer` protocols.

User Identification. The oracle $\mathcal{O}\texttt{IDRetrieve}(T_{t,j})$ executes the `IDRetrieve` algorithm. It outputs the tuple $(Id_i, g_I^{u_i}, \texttt{Att}(Id_i, g_I^{u_i}))$ along with a proof $\pi_{3,t,j+1}$ proving this claim. If $T_{t,j}$ is a `Transfer` transaction, then Id_i is the identity of $T_{t,j}$'s payee. Otherwise, Id_i is the identity of the user who performed the `Insert` or `Deposit` transaction $T_{t,j}$. $T_{t,j}$ is added to the set of revoked transactions $\mathcal{RT}$.

4.3 Security properties

Hereinafter, we provide both formal and informal definitions of the security and privacy properties that a digital currency system $\mathcal{DCS}$ must satisfy.

Unforgeability. Informally, unforgeability requires that no entity, even a set of colluding users collaborating with the ID registration and revocation authorities, can transfer and/or deposit more digital currencies than what it has honestly got, *i.e.* transfer and/or deposit an amount that is greater than the total amount of what colluding users have withdrawn from currency issuers and received from other users minus already transferred and deposited digital currencies. More formally, the unforgeability experiment $\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^\lambda)$ is defined in Figure 1. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{unforg}}(1^\lambda)$ of the adversary is defined as $\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^\lambda) = 1]$.

A digital currency system $\mathcal{DCS}$ satisfies the *unforgeability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\texttt{Exp}_{\mathcal{A}}^{\text{unforg}}(1^\lambda)$

1. $\mathcal{HU} \leftarrow \varnothing$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{DL} \leftarrow \varnothing$, $\mathcal{DB}_{Priv} \leftarrow \varnothing$, $\mathcal{DB}_{Pub} \leftarrow \varnothing$, $\mathcal{T} \leftarrow \varnothing$
2. $(sp, (sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}}), (sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}}), (sk_{\mathcal{RA}}^{eg}, pk_{\mathcal{RA}}^{eg}), (sk_{\mathcal{CI}_i}^{acl}, pk_{\mathcal{CI}_i}^{acl}), (sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i}))$ $\leftarrow \texttt{Setup}(1^\lambda)$
3. $\{T_r\}_{r=1}^{\ell} \longleftarrow \mathcal{A}^{\mathcal{O}}(sk_{\mathcal{IDR}}, sk_{\mathcal{RA}}^{eg}, \mathcal{DB}_{Priv})$ where $\mathcal{O}$ = all oracles
4. $\forall r \in \{1, \ldots, \ell\}$, if T_r is valid, $T_r \notin \mathcal{DL}$ and $T_r \notin \mathcal{T}$, then $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_r\}$.
5. If $\sum_{r=1}^{\#\mathcal{T}} \texttt{Amount}_{T_r} > \sum_{i=1}^{\#\mathcal{CU}} \texttt{Amount}_{Withdrawn_i + Received_i - Transferred_i - Deposited_i}$, then return `1`.
6. Return `0`.

Fig. 1. Unforgeability Security Experiment

Exculpability. Roughly speaking, exculpability requires that no entity, including the revocation authority and even with the help of malicious users and digital currency issuers, can generate a valid transaction $T_{t,j}$ that wrongly accuse an honest user either of performing a given `Insert` or `Deposit` transaction, or of being the payee or payer of a given `Transfer` transaction, while it is not the case. Hence, users cannot insert new digital currencies by pretending to be another honest user, or add in the distributed ledger a transaction that is falsely associated to an honest user that was not involved in that transaction. More formally, the exculpability experiment $\texttt{Exp}_{\mathcal{A}}^{\text{excul}}(1^\lambda)$ is defined in Figure 2. The advantage $\texttt{Adv}_{\mathcal{A}}^{\text{excul}}(1^\lambda)$ of the adversary is defined as $\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{excul}}(1^\lambda) = 1]$.

A digital currency system $\mathcal{DCS}$ satisfies the *exculpability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\texttt{Exp}^{\text{excul}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{HU} \leftarrow \varnothing$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{DL} \leftarrow \varnothing$, $\mathcal{DB}_{Pub} \leftarrow \varnothing$, $\mathcal{DB}_{Priv} \leftarrow \varnothing$, $\mathcal{TS}_{\mathcal{U}_i} \leftarrow \varnothing$
2. $(sp, (sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}}), (sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}}), (sk^{eg}_{\mathcal{RA}}, pk^{eg}_{\mathcal{RA}}), (sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i}), (sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i}))$ $\leftarrow \texttt{Setup}(1^\lambda)$
3. $T_{t,j}, \pi_{3,t,j+1} \longleftarrow \mathcal{A}^{\mathcal{O}}(sk^{eg}_{\mathcal{RA}}, sk^{acl}_{\mathcal{CI}_i}, sk_{\mathcal{CI}_i}, \mathcal{DB}_{Priv})$ where $\mathcal{O}$ = all oracles.
4. If $T_{t,j} \notin \mathcal{DL}$ or $\pi_{3,t,j+1}$ is not valid or $\texttt{IDRetrieve}(T_{t,j}, \mathcal{DB}_{Priv}, sk_{\mathcal{RA}}) = \perp$, then return 0.
5. If $\texttt{IDRetrieve}(T_{t,j}, \mathcal{DB}_{Priv}, sk^{eg}_{\mathcal{RA}}) = (Id_i, \pi_{3,t,j+1})$ where Id_i is associated to $\mathcal{U}_i \in \mathcal{HU}$ and $T_{t,j} \notin \mathcal{TS}_{\mathcal{U}_i}$, then return 1.
6. Return 0

Fig. 2. Exculpability Security Experiment

Accountability. Informally, the accountability property requires that no adversary, even a set of colluding users collaborating with the digital currency issuers, should be able to produce a valid transaction $T_{t,j}$ which cannot be traced by an honest revocation authority $\mathcal{RA}$ to the identity Id_i of the payer and/or the payee who was involved in it, *i.e.* to the value $g_I^{u_i}$ provided by one of the legitimate users during an execution of the `IDRegister` protocol. More formally, the accountability experiment $\texttt{Exp}^{\text{account}}_{\mathcal{A}}(1^\lambda)$ is defined in Figure 3. The adversary's advantage $\texttt{Adv}^{\text{account}}_{\mathcal{A}}(1^\lambda)$ is defined as $\texttt{Pr}[\texttt{Exp}^{\text{account}}_{\mathcal{A}}(1^\lambda) = 1]$.

A digital currency system $\mathcal{DCS}$ satisfies the *accountability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

$\texttt{Exp}^{\text{account}}_{\mathcal{A}}(1^\lambda)$

1. $\mathcal{HU} \leftarrow \varnothing$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{DL} \leftarrow \varnothing$, $\mathcal{DB}_{Priv} \leftarrow \varnothing$, $\mathcal{DB}_{Pub} \leftarrow \varnothing$
2. $(sp, (sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}}), (sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}}), (sk^{eg}_{\mathcal{RA}}, pk^{eg}_{\mathcal{RA}}), (sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i}), (sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i}))$ $\leftarrow \texttt{Setup}(1^\lambda)$
3. $T_{t,j} \longleftarrow \mathcal{A}^{\mathcal{O}}(sk^{acl}_{\mathcal{CI}_i}, sk_{\mathcal{CI}_i}, \mathcal{DB}_{Priv})$ where $\mathcal{O}$ = all oracles.
4. If $T_{t,j} \notin \mathcal{DL}$, then return 0.
5. If $\texttt{IDRetrieve}(T_{t,j}, \mathcal{DB}_{Priv}, sk^{eg}_{\mathcal{RA}}) = \perp$ or $\texttt{IDRetrieve}(T_{t,j}, \mathcal{DB}_{Priv}, sk^{eg}_{\mathcal{RA}}) = \widetilde{Id}_i$ where $\widetilde{Id}_i$ is a fake identity, then return 1.
6. Return 0

Fig. 3. Accountability Security Experiment

End-user's Privacy. Roughly speaking, end-user's privacy requires that, aside from the revocation authority and even with the help of malicious users, digital currency issuers and the ID registration authority, no entity can (1) link a given `Transfer`[5] or `Deposit` transaction to an execution of the `Withdrawal` protocol; or (2) figure out whether two `Transfer` transactions, or a `Transfer`

[5] We consider that the `Insert` transaction is combined with the first `Transfer` transaction associated with the new withdrawn digital currency.

$\texttt{Exp}_{\mathcal{A}}^{\text{UPrivacy}}(1^{\lambda})$

1. $\mathcal{HU} \leftarrow \varnothing$, $\mathcal{CU} \leftarrow \varnothing$, $\mathcal{DL} \leftarrow \varnothing$, $\mathcal{DB}_{Pub} \leftarrow \varnothing$, $\mathcal{DB}_{Priv} \leftarrow \varnothing$, $\mathcal{TS}_{\mathcal{CU}} \leftarrow \varnothing$, $\mathcal{RT} \leftarrow \varnothing$
2. $(sp, (sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}}), (sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}}), (sk^{eg}_{\mathcal{RA}}, pk^{eg}_{\mathcal{RA}}), (sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i}), (sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i}))$ $\leftarrow \texttt{Setup}(1^{\lambda})$
3. $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_R \leftarrow \mathcal{A}^{\mathcal{O}}(sk^{acl}_{\mathcal{CI}_i}, sk_{\mathcal{CI}_i}, sk_{\mathcal{IDR}}, \mathcal{DB}_{Pub})$ where $\mathcal{O}$ = all oracles except the following oracles: $\mathcal{O}\texttt{Transfer}_S(\mathcal{U}_i, \perp)$, $\mathcal{O}\texttt{Transfer}_R(\mathcal{U}_i)$, $\mathcal{O}\texttt{Transfer}_{S\&R}(\mathcal{U}_i, \mathcal{U}_j, \perp)$, $\mathcal{O}\texttt{Transfer}_{S\&R}(\perp, \mathcal{U}_j, \perp)$ and $\mathcal{O}\texttt{Transfer}_{S\&R}(\perp, \mathcal{U}_j, DC_t)$.
4. If $\mathcal{U}_0 \in \mathcal{CU}$ or $\mathcal{U}_1 \in \mathcal{CU}$, then output $\perp$.
5. If $\mathcal{U}_0$ and $\mathcal{U}_1$ do not own at least one digital currency of the same size and worth the same amount and such that, for that digital currency, the two concurrent latest transactions $T_{*,j-1}$ and $T^*_{*,j-1} \notin \mathcal{TS}_{\mathcal{CU}}$ and all the transactions in its history $\notin \mathcal{RT}$, then output $\perp$.
6. Select two digital currencies DC_t and $DC_{t'}$ respectively belonging to $\mathcal{U}_0$ and $\mathcal{U}_1$, and satisfying conditions of step 5. For $b \in \{0, 1\}$, run:
   $(T_b, T^*_b) \leftarrow \texttt{Transfer}(DC_t, \texttt{amount}_c, 0, \mathcal{U}_0, \mathcal{U}_R)$,
   $(T_{1-b}, T^*_{1-b}) \leftarrow \texttt{Transfer}(DC_{t'}, \texttt{amount}_c, 0, \mathcal{U}_1, \mathcal{U}_R)$
   which correspond to the transfer of the full amount $\texttt{amount}_c$ of digital currencies DC_t and $DC_{t'}$ from $\mathcal{U}_b$ to $\mathcal{U}_R$.
7. $b' \leftarrow A^{\mathcal{O}}(sk^{acl}_{\mathcal{CI}_i}, sk_{\mathcal{CI}_i}, sk_{\mathcal{IDR}}, \mathcal{DB}_{Pub})$ where $\mathcal{O}$ = all oracles except $\mathcal{O}\texttt{IDRetrieve}$ and with a restricted access to $\mathcal{O}\texttt{UCorrupt}$.
8. If $\mathcal{O}\texttt{UCorrupt}(\mathcal{U}_0)$ or $\mathcal{O}\texttt{UCorrupt}(\mathcal{U}_1)$, then output $\perp$.
9. If T_b, T^*_b, T_{1-b} or $T^*_{1-b} \in \mathcal{RT}$, then output $\perp$.
10. If $b' = b$, then return 1.
11. Return 0.

Fig. 4. End-user's Privacy Security Experiment

and a `Deposit` transactions $T_{t,j}$ and $T_{t',j'}$, including different $\mathcal{H}(T_{t,j-1})$ and $\mathcal{H}(T_{t',j'-1})$ values[6] and such that neither $\mathcal{H}(T_{t,j-1})$ is in the history of $T_{t',j'}$ nor $\mathcal{H}(T_{t',j'-1})$ is in the history of $T_{t,j}$, were performed by the same payer[7] (or not), not necessarily identified.

It is well known that the size of a transferable digital currency incrementally grows as the digital currency is transferred [7]. This characteristic may be leveraged by an adversary $\mathcal{A}$ to win the game and break end-user's privacy property, *e.g.* by selecting two users $\mathcal{U}_0$ and $\mathcal{U}_1$ that do not own digital currencies of the same size, *i.e.* which has been transferred the same number of times. Thus, in the defined game, we require that both users $\mathcal{U}_0$ and $\mathcal{U}_1$ own at least one digital currency of the same size that will be used to output the challenge transactions.

Due to the *traceability* feature, an adversary $\mathcal{A}$ is able to recognize a digital currency DC_t that it has already observed during previous transactions, and decide whether a given transaction is related to the same digital currency or not. Consequently, the challenge transactions T_b and T_{1-b} should correspond to transfers of the full amount of the digital currency to the receiver $\mathcal{U}_R$ that was selected by $\mathcal{A}$. Otherwise, an adversary $\mathcal{A}$ can for instance make oracle queries

[6] Since two transactions including the same $\mathcal{H}(T_{t,j-1})$ value are, by default, performed by the same user.

[7] As we want to trace issued digital currencies, any entity knows that the same user is involved in two subsequent transactions associated to the same digital currency, first as a payee then as a payer.

to $\mathcal{O}\texttt{Transfer}_S(\mathcal{U}_b, \perp)$ for $b \in \{0,1\}$ until it gets a transaction associated to the same DC_t as the challenge transaction, thus trivially winning the game.

Similarly, an adversary $\mathcal{A}$ can leverage transactions history that is available in $\mathcal{DL}$. Hence, $\mathcal{A}$ does not have access to some oracles as described in Figure 4.

Since an adversary $\mathcal{A}$ knows the amount associated to the challenge transactions, then both used digital currencies should be worth the same amount $\texttt{amount}_c$. Otherwise, $\mathcal{A}$ may exploit $\texttt{amount}_c$ to win the game, *e.g.* by leveraging the total amount of digital currencies owned by $\mathcal{U}_0$ and $\mathcal{U}_1$.

More formally, the end-user's privacy experiment $\texttt{Exp}_{\mathcal{A}}^{\text{UPrivacy}}(1^\lambda)$ is detailed in Figure 4. The adversary's advantage $\texttt{Adv}_{\mathcal{A}}^{\text{UPrivacy}}(1^\lambda)$ is defined as

$$\texttt{Adv}_{\mathcal{A}}^{\text{UPrivacy}}(1^\lambda) = |\texttt{Pr}[\texttt{Exp}_{\mathcal{A}}^{\text{UPrivacy}}(1^\lambda) = 1] - 1/2|$$

A digital currency system $\mathcal{DCS}$ satisfies *end-user's privacy* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

5 Our traceable, transferable and divisible digital currency system

5.1 Overview

Our digital currency system $\mathcal{DCS}$ relies on the following crypto building blocks: (i) a digital signature scheme ($\texttt{SKeyGen}$, $\texttt{SSign}$, $\texttt{SVerif}$) denoted by $\texttt{S}$, such as an ECDSA signature, (ii) the adaptation of Anonymous Credential Light (ACL) signature scheme ($\texttt{ACLKeyGen}$, $\texttt{ACLSign}$, $\texttt{ACLVerif}$) introduced in Appendix A, (iii) the ElGamal encryption scheme ($\texttt{EGKeyGen}$, $\texttt{EGEnc}$, $\texttt{EGDec}$) introduced in Section 2.3, and (iv) the three types of non-interactive zero-knowledge proof of knowledge π_1, π_2 and π_3 introduced in Section 2.5.

5.2 Our system description

Our $\mathcal{DCS}$ system involves five entities, namely a certification authority $\mathcal{P}_{CA}$, an ID registration authority $\mathcal{IDR}$, a revocation authority $\mathcal{RA}$, n digital currency issuers $\mathcal{CI}_1, \ldots, \mathcal{CI}_n$, and a large number of banked and unbanked end-users $\mathcal{U}_i$.

$\mathcal{CI}_1, \ldots, \mathcal{CI}_n$ setup and maintain a permissioned distributed ledger where all digital currencies transactions are stored. The revocation authority $\mathcal{RA}$ maintains a private database $\mathcal{DB}_{Priv}$ of triplets $\{Id_i, g_I^{u_i}, \texttt{att}(Id_i, g_I^{u_i})\}$ and a corresponding public database $\mathcal{DB}_{Pub}$ of $\{g_I^{u_i}\}$.

Setup. The system parameters are $sp = (sp_s, sp_{eg}, sp_{acl}, sp_{id})$ where sp_s denotes the parameters of the signature scheme $\texttt{S}$, $sp_{eg} = (\mathbb{G}, q, g_E)$ denotes the parameters of the El Gamal cryptosystem, $sp_{acl} = (\mathbb{G}, q, g, h, h_0, h_1)$ denotes the parameters of our variant of the ACL signature scheme and $sp_{id} = (\mathbb{G}, q, g_I)$.

During this phase, the certification authority $\mathcal{P}_{CA}$ uses the $\texttt{SKeyGen}$ algorithm to generate a signature key pair $(sk_{\mathcal{P}_{CA}}, pk_{\mathcal{P}_{CA}})$ that is used by $\mathcal{P}_{CA}$ to

issue certificates to the other system entities. Next, the ID registration authority $\mathcal{IDR}$ uses the $\texttt{SKeyGen}$ algorithm to generate a signature key pair $(sk_{\mathcal{IDR}}, pk_{\mathcal{IDR}})$. Using the $\texttt{EGKeyGen}$ algorithm, the revocation authority $\mathcal{RA}$ generates an El-Gamal encryption key pair $(sk^{eg}_{\mathcal{RA}}, pk^{eg}_{\mathcal{RA}})$. Then, both $\mathcal{IDR}$ and $\mathcal{RA}$ execute the $\texttt{RequestCertificate}$ protocol to respectively get the certificates $\texttt{cert}(pk_{\mathcal{IDR}}) = \texttt{SSign}[sk_{\mathcal{P}_{CA}}](pk_{\mathcal{IDR}}, ID\ registration)$ and $\texttt{cert}(pk_{\mathcal{RA}}) = \texttt{SSign}[sk_{\mathcal{P}_{CA}}](pk^{eg}_{\mathcal{RA}}, Revocation)$ that are issued by $\mathcal{P}_{CA}$.

Each $\mathcal{CI}_i$ generates an ACL signature key pair $(sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i})$ for digital currencies issuance and executes $\texttt{RequestCertificate}$ protocol to get the certificate $\texttt{cert}(pk^{acl}_{\mathcal{CI}_i}) = \texttt{SSign}[sk_{\mathcal{P}_{CA}}](pk^{acl}_{\mathcal{CI}_i}, z, issuer, DC\ issuance)$ from $\mathcal{P}_{CA}$. Each $\mathcal{CI}_i$ also generates a signature key pair $(sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i})$ to be used for the deposit of digital currencies and gets the associated certificate $\texttt{cert}(pk_{\mathcal{CI}_i}) = \texttt{SSign}[sk_{\mathcal{P}_{CA}}](pk_{\mathcal{CI}_i}, z, issuer, Deposit)$ from $\mathcal{P}_{CA}$.

Join procedure. When a banked or unbanked end-user $\mathcal{U}_i$ wants to join the digital currency system, he proceeds as follows.

1. **End-user identity verification by $\mathcal{IDR}$:**
   - $\mathcal{U}_i$ provides a proof of his identity Id_i to $\mathcal{IDR}$;
   - $\mathcal{U}_i$ generates a pair $(u_i, g_I^{u_i})$ using $\texttt{IDKeyGen}(sp)$;
   - $\mathcal{U}_i$ sends $g_I^{u_i}$ to $\mathcal{IDR}$ along with a proof π_1 proving the knowledge of u_i;
   - If all previous steps succeed, then $\mathcal{IDR}$ uses the signature scheme $\texttt{S}$ to issue an attestation $\texttt{att}(Id_i, g_I^{u_i}) = \texttt{SSign}[sk_{\mathcal{IDR}}](Id_i, g_I^{u_i})$ to $\mathcal{U}_i$.
2. **End-user registration with $\mathcal{RA}$:**
   - $\mathcal{U}_i$ sends $(Id_i, g_I^{u_i}, \texttt{att}(Id_i, g_I^{u_i}))$ to $\mathcal{RA}$ along with a proof π_1 proving the knowledge of u_i;
   - $\mathcal{RA}$ verifies $\texttt{att}(Id_i, g_I^{u_i})$ validity and that $\mathcal{U}_i$ actually knows u_i. If so, $\mathcal{RA}$ updates both $\mathcal{DB}_{Priv}$ by adding the triplet $\{Id_i, g_I^{u_i}, \texttt{att}(Id_i, g_I^{u_i})\}$ and $\mathcal{DB}_{Pub}$ by appending the singleton $\{g_I^{u_i}\}$.

Withdrawal protocol. The withdrawal protocol involves a banked user $\mathcal{U}_i$ and a digital currency issuer $\mathcal{CI}_i$ holding the key pair $(sk^{acl}_{\mathcal{CI}_i}, pk^{acl}_{\mathcal{CI}_i})$. It relies on the adaptation of Anonymous Credential Light (ACL) introduced in Appendix A.

First, mutual authentication is performed between $\mathcal{U}_i$ and $\mathcal{CI}_i$ with usual authentication means that are outside of the scope of this description. The digital currency issuer $\mathcal{CI}_i$ identifies the user $\mathcal{U}_i$ and checks the balance of his banking account before both $\mathcal{U}_i$ and $\mathcal{CI}_i$ agree on the amount $\texttt{amount}_\texttt{t}$ of the digital currency DC_t to be withdrawn by $\mathcal{U}_i$.

In a nutshell, the user $\mathcal{U}_i$ generates a random nonce $\texttt{sn}'_t$ that is not known by $\mathcal{CI}_i$ whilst $\mathcal{CI}_i$ generates a random nonce $\texttt{sn}^*_t$ so that the serial number of DC_t will be $\texttt{sn}_t = \texttt{sn}'_t + \texttt{sn}^*_t$. This value will only be known to $\mathcal{U}_i$. Moreover, the user $\mathcal{U}_i$ uses the $\texttt{SKeyGen}$ algorithm to generate a one time use key pair $(sk_{t,0}, pk_{t,0})$ and sets $pk_{t,0}$ as the message m to be blindly signed by $\mathcal{CI}_i$ using $\texttt{ACLSign}$. The protocol is played as described in Appendix A and it either outputs a digital currency $DC_t = (\texttt{sn}_t, \texttt{amount}_t, pk_{t,0}, \texttt{ACLSign}[sk^{acl}_{\mathcal{CI}_i}](\texttt{sn}_t, \texttt{amount}_t, pk_{t,0}))$ or 0.

Insertion protocol. The insertion protocol is performed by a banked user $\mathcal{U}_i$ who is willing to insert a withdrawn digital currency $DC_t = (\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}, \mathtt{ACLSign}\ [sk^{acl}_{\mathcal{CI}_i}](\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}))$ into the distributed ledger. The end-user $\mathcal{U}_i$ knows the secret key $sk_{t,0}$ associated to the public key $pk_{t,0}$ and a pair $(u_i, g_I^{u_i})$ such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$. To insert DC_t, $\mathcal{U}_i$ computes an encryption $E_{t,1} = (g_E^{r_1}, g_I^{u_i} pk_E^{r_1})$ of $g_I^{u_i}$ using $pk_E = pk^{eg}_{\mathcal{RA}}$, and $C_{t,1} = g^{u_i} pk_E^{r_2}$ along with a NIZK proof $\pi_{2,t,1}$ that $E_{t,1}$ was correctly computed and that he knowns u_i such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$. The challenge of $\pi_{2,t,1}$ is defined as $c = \mathcal{H}(DC_t)$. The insertion transaction that is transmitted to the permissioned ledger is $T_{t,0} = (insert, DC_t, E_{t,1}, \pi_{2,t,1}, \mathtt{S}_{t,0})$ where $\mathtt{S}_{t,0} = \mathtt{SSign}[sk_{t,0}](insert, DC_t, E_{t,1}, \pi_{2,t,1})$. Either the transaction $T_{t,0}$ is valid and inserted into the distributed ledger $\mathcal{DL}$ or the output is 0.

Transfer protocol. The transfer protocol involves a banked or an unbanked user $\mathcal{U}_i$ owning the digital currency related to a validated transaction $T_{t,j-1}$ which is either of the form $T_{t,0} = (insert, DC_t, E_{t,1}, \pi_{2,t,1}, \mathtt{S}_{t,0})$ with $DC_t = (\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}, \mathtt{ACLSign}[sk^{acl}_{\mathcal{CI}_i}](\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}))$ or $T_{t,j-1} = (transfer, \mathtt{amount}_{t,j-1}, \mathcal{H}(T_{t,j-2}), pk_{t,j}, E_{t,j}, \pi_{2,t,j}, \mathtt{S}_{t,j-1})$ for $j > 1$. The user $\mathcal{U}_i$ knows the private key $sk_{t,j}$ associated to $pk_{t,j}$ and a pair $(u_i, g_I^{u_i})$ such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$.

The transfer protocol also involves a banked or unbanked receiver of the digital currency transfer $\mathcal{U}_r$ who holds a pair $(u_r, g_I^{u_r})$ such that $g_I^{u_r} \in \mathcal{DB}_{Pub}$. Both $\mathcal{U}_r$ and $\mathcal{U}_i$ agree on the amount of the transfer $\mathtt{amount}_{t,j}$ with $0 < \mathtt{amount}_{t,j} \leq \mathtt{amount}_{t,j-1}$, and use the $\mathtt{SkeyGen}$ algorithm to respectively generate the signature key pairs $(sk_{t,j+1}, pk_{t,j+1})$ and $(sk^*_{t,j+1}, pk^*_{t,j+1})$.

The transfer protocol requires two transactions. For the first transaction, $\mathcal{U}_r$ computes the value $E_{t,j+1}$ which is the encryption of g^{u_r} using $pk^{eg}_{\mathcal{RA}}$ and the NIZK proof $\pi_{2,t,j+1}$ ensuring that $E_{t,j+1}$ was correctly computed and that he knows u_r such that $g^{u_r} \in \mathcal{DB}_{Pub}$. The challenge of $\pi_{2,t,j+1}$ is $c = \mathcal{H}(\mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1})$. Then, $\mathcal{U}_i$ computes the first transaction which corresponds to a digital currency transfer from $\mathcal{U}_i$ to $\mathcal{U}_r$:

$$T_{t,j} = (transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$$

where $\mathtt{S}_{t,j} = \mathtt{SSign}[sk_{t,j}](transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1})$.

The second transaction for this transfer is an auto-transfer of the remaining amount $\mathtt{amount}^*_{t,j} = \mathtt{amount}_{t,j-1} - \mathtt{amount}_{t,j}$ from $\mathcal{U}_i$ to $\mathcal{U}_i$:

$$T^*_{t,j} = (transfer, \mathtt{amount}^*_{t,j}, \mathcal{H}(T_{t,j-1}), pk^*_{t,j+1}, E^*_{t,j+1}, \pi^*_{2,t,j+1}, \mathtt{S}^*_{t,j})$$

where $\mathtt{S}^*_{t,j} = \mathtt{SSign}[sk_{t,j}](transfer, \mathtt{amount}^*_{t,j}, \mathcal{H}(T_{t,j-1}), pk^*_{t,j+1}, E^*_{t,j+1}, \pi^*_{2,t,j+1})$, $E^*_{t,j+1}$ is the encryption using $pk^{eg}_{\mathcal{RA}}$ of $g_I^{u_i}$ and $\pi^*_{2,t,j+1}$ is a NIZK proof ensuring that $E^*_{t,j+1}$ was correctly computed and that $\mathcal{U}_i$ knows u_i such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$. The challenge of $\pi^*_{2,t,j+1}$ is $c = \mathcal{H}(\mathtt{amount}^*_{t,j}, \mathcal{H}(T_{t,j-1}), pk^*_{t,j+1})$. All these values are computed by the user $\mathcal{U}_i$.

Both transactions $T_{t,j}$ and $T^*_{t,j}$ are jointly transmitted in a random order to the permissioned ledger. Either the two transactions $T_{t,j}$ and $T^*_{t,j}$ are valid and inserted into the distributed ledger or the output is 0.

Deposit protocol. The deposit protocol involves a banked user $\mathcal{U}_i$ owning the digital currency related to $T_{t,j-1}$, *i.e.* holding the private key $sk_{t,j}$ associated to the public key $pk_{t,j}$ included in $T_{t,j-1}$, and a digital currency issuer $\mathcal{CI}_i$ holding the key pair $(sk_{\mathcal{CI}_i}, pk_{\mathcal{CI}_i})$. It requires a single transaction computed by $\mathcal{U}_i$ and transmitted to the permissioned ledger:

$$T_{t,j} = (deposit, \mathtt{amount}_{t,j-1}, \mathcal{H}(T_{t,j-1}), pk_{\mathcal{CI}_i}, \mathtt{S}_{t,j})$$

where $\mathtt{S}_{t,j} = \mathtt{SSign}[sk_{t,j}](deposit, \mathtt{amount}_{t,j-1}, \mathcal{H}(T_{t,j-1}), pk_{\mathcal{CI}_i})$.

Either the transaction $T_{t,j}$ is inserted into the ledger and the banking account of $\mathcal{U}_i$ is updated with $\mathtt{amount}_{t,j-1}$ or the output is 0.

VerifyGuilt procedure. The VerifyGuilt procedure is performed by the revocation authority $\mathcal{RA}$ upon a request to identify the payee of a specific transaction $T_{t,j}$ of the form either $T_{t,0} = (insert, DC_t, E_{t,1}, \pi_{2,t,1}, \mathtt{S}_{t,0})$ or $T_{t,j} = (transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$ with $j \geq 1$.

Using the private key $sk^{eg}_{\mathcal{RA}}$, $\mathcal{RA}$ decrypts $E_{t,j+1}$ to get the value $g_I^{u_i}$ and builds a NIZK proof $\pi_{3,t,j}$ that the decryption was correctly performed. Then, it looks for the corresponding entry $\{Id_i, g_I^{u_i}, \mathtt{att}(Id_i, g_I^{u_i})\}$ in $\mathcal{DB}_{Priv}$. Finally, $\mathcal{RA}$ either outputs the identity Id_i along with the NIZK proof $\pi_{3,t,j+1}$ and the triplet $\{Id_i, g_I^{u_i}, \mathtt{att}(Id_i, g_I^{u_i})\}$ enabling to prove this claim, or the output is 0.

5.3 Security Analysis

Theorem 1. *Our digital currency system $\mathcal{DCS}$ satisfies the* unforgeability, exculpability *and* accountability *properties under the discrete logarithm* (DL) *assumption and the unforgeability of the used signature scheme* S*, and the* end-user's privacy *property is fulfilled under the Decisional Diffie Hellman* (DDH) *assumption in the random oracle model* [8].

6 Conclusion

In this paper, our contribution is threefold. First, we investigated the highest level of anonymity and unlinkability that a traceable, transferable and divisible digital currency system $\mathcal{DCS}$ may achieve. Then, we defined a framework for such a $\mathcal{DCS}$ system along with the associated security and privacy model. Finally, we proposed a traceable, transferable and divisible digital currency system that protects user's privacy while enabling the retrieval of user's identity in case of suspicious transactions, e.g. suspicion of fraud or money laundering activities. Our $\mathcal{DCS}$ system is mainly based on a slight adaptation of ACL signature and ElGamal encryption schemes as well as NIZK proofs, and it is proven secure in the random oracle model.

[8] Owing to the lack of space, the security proofs are provided in Appendix B.

References

1. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1163, pp. 244–251 (1996)
2. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1880, pp. 271–286. Springer (2000)
3. Baldimtsi, F., Chase, M., Fuchsbauer, G., Kohlweiss, M.: Anonymous transferable e-cash. In: Katz, J. (ed.) Public-Key Cryptography – PKC 2015. pp. 101–124. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
4. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Sadeghi, A., Gligor, V.D., Yung, M. (eds.) 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. pp. 1087–1098. ACM (2013)
5. Canard, S., Gouget, A.: Anonymity in transferable e-cash. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5037, pp. 207–223 (2008)
6. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982. pp. 199–203. Plenum Press, New York (1982)
7. Chaum, D., Pedersen, T.P.: Transferred cash grows in size. In: Rueppel, R.A. (ed.) Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings. Lecture Notes in Computer Science, vol. 658, pp. 390–407 (1992)
8. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 186–194 (1986)
9. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
10. KHinterwälder, G., Zenger, C.T., Baldimtsi, F., Lysyanskaya, A., Paar, C., Burleson, W.P.: Efficient E-Cash in Practice: NFC-Based Payments for Public Transportation Systems. In: Cristofaro, E.D., Wright, M.K. (eds.) Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7981, pp. 40–59 (2013)
11. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2587, pp. 389–406. Springer (2002)

Appendix A: Adaptation of Anonymous Credential Light

As mentioned in Section 2.6, for our $\mathcal{DCS}$ system, we need a (partially) blind signature with two specific attributes: (1) an attribute for the amount of withdrawn digital currency that is known by both the signer and the user; and (2) an attribute for the digital currency serial number $\mathtt{sn}$ that is jointly computed by the user and the signer but that is only known to the user. The message m corresponds to the user's public key $pk_{t,0}$, and is only known to the user. Both m and $\mathtt{sn}$ need to be kept secret from the signer to ensure the blindness property.

The Anonymous Credential Light (ACL) system, as described in [4], just considers attributes that are only known to the user and it does not directly define such particular attributes. We thus propose a couple of slight modifications of ACL system so as to support these two specific attributes. In our context, we only require these two attributes and we do not need any additional attribute only defined and know by the user.

Signer		**User**
	Public Input: $\mathbb{G}, q, g, h, h_0, h_1, y, z$	
	Common Input: $\mathtt{amount}$	
Private Input: x		**Private Input:** m
		Generate $R, \mathtt{sn}' \in_R \mathbb{Z}_q^*$
		Compute $C = h_0^R g^{\mathtt{sn}'}$
Verify π_1	$\xleftarrow{C,\pi_1}$	**Build** $\pi_1 = PoK\{\theta, \beta : C = h_0^\theta g^\beta\}$
Generate $\mathtt{sn}^* \in_R \mathbb{Z}_q^*$		
Compute $z_1 = C \cdot g^{\mathtt{sn}^*} h_1^{\mathtt{amount}}$ and $z_2 = z/z_1$		
Generate $u, r_1', r_2', c' \in_R \mathbb{Z}_q$		
Compute $a = g^u$		
$a_1' = g^{r_1'} z_1^{c'}$ and $a_2' = h^{r_2'} z_2^{c'}$	$\xrightarrow{sn^*, a, a_1', a_2'}$	**Check that** $sn^* \neq 0$
		Compute $z_1 = C \cdot g^{\mathtt{sn}^*} h_1^{\mathtt{amount}}$
		Select $\gamma \in_R \mathbb{Z}_q^*$
		Compute $\zeta = z^\gamma$, $\zeta_1 = z_1^\gamma$ and $\zeta_2 = \zeta/\zeta_1$
		Select $\tau \in_R \mathbb{Z}_q^*$ and **compute** $\eta = z^\tau$
		Check that $a, a_1', a_2' \in \mathbb{G}$
		Generate $t_1, t_2, t_3, t_4, t_5 \in_R \mathbb{Z}_q^*$
		Compute $\alpha = a g^{t_1} y^{t_2}$
		$\alpha_1' = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$ and $\alpha_2' = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$
		$\varepsilon = \mathcal{H}(\zeta, \zeta_1, \alpha, \alpha_1', \alpha_2', \eta, m)$
	$\xleftarrow{e}$	$e = \varepsilon - t_2 - t_4 \mod q$
Compute $c = e - c' \mod q$		
$r = u - cx \mod q$	$\xrightarrow{c, r, c', r_1', r_2'}$	
		Compute $\rho = r + t_1 \mod q$
		$w = c + t_2 \mod q$, $\rho_1' = \gamma r_1' + t_3 \mod q$
		$\rho_2' = \gamma r_2' + t_5 \mod q$, $\omega' = c' + t_4 \mod q$
		$\mu = \tau - \omega' \gamma \mod q$
		Check if $\omega + \omega' \stackrel{?}{=}$
		$\mathcal{H}(\zeta, \zeta_1, g^\rho y^\omega, g^{\rho_1'} \zeta_1^{\omega'}, h^{\rho_2'} \zeta_2^{\omega'}, z^\mu \zeta^{\omega'}, m) \mod q$
		The signature is $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho_1', \rho_2', \omega', \mu)$.

Fig. 5. Adapted ACL - Credential issuance on message m and attributes $\mathtt{sn}$ and $\mathtt{amount}$

The ACL scheme consists of a the triplet of algorithms (`ACLKeyGen`, `ACLSign`, `ACLVerif`) where `ACLKeyGen` takes as input the system parameters and outputs a signature key pair (sk^{acl}, pk^{acl}), `ACLSign` takes as input two attributes `amount` and `sn` as well as a message m, and outputs the signature σ, whereas `ACLVerif` takes as input σ, `amount`, `sn` and m, and outputs either `1` or `0`. Hereinafter, we do not provide the description of the original ACL system [4] but rather only detail our adaptation, depicted in Figure 5, while pointing out the difference with respect to the original scheme. The system public parameters consist of the tuple $(\mathbb{G}, q, g, h, h_0, h_1, \mathcal{H})$ where $\mathbb{G}$ is a cyclic group of prime order q where the DDH problem is assumed to be hard, g, h, h_0, h_1 are generators of $\mathbb{G}$, and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a hash function. The signer secret key is $x \in_R \mathbb{Z}_q$. The corresponding certified public key is $(\mathbb{G}, q, g, h, y, z)$ where $y = g^x \mod q$ is the *real* public key and $z \in_R \mathbb{G}$ is referred to as *tag public key.*

The obtained signature on message m and both attributes $\mathtt{sn} = \mathtt{sn}' + \mathtt{sn}^*$ which is only known to the user and `amount` which is known to both the user and the signer is the tuple $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu)$. The user needs to store σ along with the values m, `sn`, `amount`, R and γ. To show the obtained signature in an unlinkable way while revealing the message m and both attributes `sn` and `amount`, the user follows the protocol described in Figure 6.

User		**Verifier**
	Public Input: $\mathbb{G}, q, g, h, h_0, h_1, y, z$	
Private Input: $(m, \mathtt{sn}, \mathtt{amount})$, $\sigma = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \mu)$, γ and R		
Select $r_0, r_1, r_2 \in_R \mathbb{Z}_q^*$ **Compute** $\tilde{\zeta_1} = h_0^{r_0} h_1^{r_1} g^{r_2}$ $c = \mathcal{H}(\zeta_1, \tilde{\zeta_1}, date \ \& \ time)$ $s_0 = r_0 + c\gamma \cdot R$ $s_1 = r_1 + c\gamma \cdot \mathtt{amount}$ $s_2 = r_2 + c\gamma \cdot \mathtt{sn}$	$\xrightarrow[s_2, \mathtt{amount}, \mathtt{sn}]{m, \sigma, \tilde{\zeta_1}, s_0, s_1}$	**Check if** $\zeta \neq 1$ **Compute** $c = \mathcal{H}(\zeta_1, \tilde{\zeta_1}, date \ \& \ time)$ **Check if** $\tilde{\zeta_1}\zeta_1^c = h_0^{s_0} h_1^{s_1} g^{s_2}$ **and** $\omega + \omega' \stackrel{?}{=}$ $\mathcal{H}(\zeta, \zeta_1, g^\rho y^\omega, g^{\rho'_1} \zeta_1^{\omega'}, h^{\rho'_2} \zeta_2^{\omega'}, z^\mu \zeta^{\omega'}, m) \mod q$

Fig. 6. Adapted ACL - Credential presentation

ACL system security is proven in the random oracle model. In particular, *blindness* is ensured under the DDH assumption, and *unforgeability* is guaranteed under the DL assumption for sequential composition. Our slight adaptation of ACL does not impact the security proofs[9]. Indeed, it has no impact on the *unforgeability* requirement. As long as the same `amount` attribute is encoded in several signatures, then (partial)[10] blindness would be ensured.

[9] We refer the reader to [4] for the detailed security proofs of the ACL system.
[10] Since the signer knows the value of the attribute `amount`.

Appendix B: Security Proofs of Theorem 1

Proposition 1. *Our digital currency system $\mathcal{DCS}$ is unforgeable under the discrete logarithm assumption and the unforgeability of the* S *signature scheme.*

Proof (sketch). Let us show that if an adversary $\mathcal{A}$ is able to break the unforgeability property of our construction, then it is possible to break the unforgeability of either the ACL signature scheme [4] (which is unforgeable under the DL assumption) or the signature scheme S.

Assume that the adversary $\mathcal{A}$ wins the game. This means that the amount of digital currency held by $\mathcal{A}$ is strictly greater than the sum of the total amount of digital currencies honestly withdrawn by $\mathcal{A}$ and honestly received in transfers minus the total amount of digital currency honestly transferred to other users and deposited to digital currency issuers. This is only possible if the adversary $\mathcal{A}$ has successfully generated either:
(i) a valid digital currency without the knowledge of the key $sk^{acl}_{\mathcal{CI}_i}$:

$$DC_t = (\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}, \mathtt{ACLSign}[sk^{acl}_{\mathcal{CI}_i}](\mathtt{sn}_t, \mathtt{amount}_t, pk_{t,0}))$$

which implies that $\mathcal{A}$ has broken the unforgeability property of the ACL signature scheme, and thus the discrete logarithm assumption;

or (ii) a valid digital currency transfer without the knowledge of the key $sk_{t,j}$:

$$T_{t,j} = (transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$$

where $\mathtt{S}_{t,j} = \mathtt{SSign}[sk_{t,j}](transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1})$. This implies that $\mathcal{A}$ has broken the unforgeability property of signature scheme S;

or (iii) a valid digital currency deposit without the knowledge of the key $sk_{t,j}$:

$$T_{t,j} = (deposit, \mathtt{amount}_{t,j-1}, \mathcal{H}(T_{t,j-1}), pk_{\mathcal{CI}_i}, \mathtt{S}_{t,j})$$

where $\mathtt{S}_{t,j} = \mathtt{SSign}[sk_{t,j}](deposit, \mathtt{amount}_{t,j-1}, \mathcal{H}(T_{t,j-1}), pk_{\mathcal{CI}_i})$. This also implies that $\mathcal{A}$ has broken the unforgeability property of signature scheme S.

Note that the double-spending or double-deposit of a digital currency is not possible since all transfer and deposit transactions are recorded in the distributed ledger. Prior to the acceptance of a transfer or deposit transaction, a corresponding check is done in the distributed ledger.

Thus, our $\mathcal{DCS}$ construction satisfies the unforgeability property under the DL assumption and the unforgeability property of the used signature scheme S.

Proposition 2. *Our digital currency system $\mathcal{DCS}$ fulfills the exculpability property under the discrete logarithm assumption and the unforgeability of the used signature scheme* S.

Proof (sketch). Let us show that if an adversary $\mathcal{A}$ is able to break the exculpability property of our construction, then it is possible to break either:

- the soundness property of a NIZK proof π_2 or π_3, that relies on the discrete logarithm assumption, or
- the discrete logarithm assumption and compute the discrete logarithm u_i of $g_I^{u_i}$, or
- the unforgeability property of the signature scheme $\mathtt{S}$ and forge an attestation $\mathtt{att}(Id_i, g_I^{u_i})$.

Assume that the adversary $\mathcal{A}$ wins the game. This means that $\mathcal{A}$ has output a valid transaction $T_{t,j}$ that is of the form either $T_{t,0} = (insert, DC_t, E_{t,1}, \pi_{2,t,1}, \mathtt{S}_{t,0})$ or $T_{t,j} = (transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$ with $j \geq 1$, along with a proof π_3 that wrongly accuse an honest user of being involved in it. $\mathcal{A}$ can easily compute part of the transaction. However, it cannot replay or misuse a previously computed NIZK proof π_2 due to the proof's challenge which is computed for a specific digital currency DC_t for a $T_{t,0}$ transaction, or for a given amount, transaction $T_{t,j-1}$ and public key $pk_{t,j+1}$ for a $T_{t,j}$ with $j \geq 1$. Similarly, a proof π_3 can be neither replayed nor misused. Consequently, three cases are possible to win the game, either:

- *Case 1*: $\mathcal{A}$ has produced either (i) a valid NIZK proof $\pi_{2,t,k}$ related to a value $E_{t,k+1}$ such that $g_I^{u_i} \in \mathcal{DB}_{Pub}$ without knowing the value of u_i, which implies that the adversary has broken the soundness property of $\pi_{2,t,k}$, or (ii) a valid NIZK proof $\pi_{3,t,k}$ using an erroneous value u_i associated to an honest user, which implies that the adversary has broken the soundness property of $\pi_{3,t,k}$. Since π_2 and π_3 meet the soundness property under the discrete logarithm (DL) assumption, this case implies that $\mathcal{A}$ has broken the DL assumption.
- *Case 2*: $\mathcal{A}$ has computed the discrete logarithm of an honest user's $g_I^{u_i}$ value and used the retrieved u_i value to build a valid NIZK proof $\pi_{2,t,k}$. This case implies that $\mathcal{A}$ has broken the discrete logarithm (DL) assumption.
- *Case 3*: $\mathcal{A}$ has selected a random $u_i \in \mathbb{Z}_q^*$ and has produced a valid attestation $\mathtt{att}(Id_i, g_I^{u_i})$ corresponding to the output of the signature scheme $\mathtt{S}$ on the target identity Id_i, without knowledge of the secret key $sk_{\mathcal{IDR}}$. Then, $\mathcal{A}$ has registered with $\mathcal{RA}$ using the forged attestation and used the associated $(u_i, g_I^{u_i})$ pair to compute the ElGamal encryption and NIZK proof π_2. This case implies that $\mathcal{A}$ has broken the unforgeability property of the signature scheme $\mathtt{S}$.

Thus, our $\mathcal{DCS}$ construction satisfies the exculpability property under the DL assumption and the unforgeability property of the used signature scheme $\mathtt{S}$.

Proposition 3. *Our digital currency system $\mathcal{DCS}$ fulfills the accountability property under the discrete logarithm assumption and the unforgeability of the used signature scheme* $\mathtt{S}$.

Proof (sketch). Let us show that if an adversary $\mathcal{A}$ is able to break the accountability property of our construction, then it is possible to break either:

- the soundness property of a NIZK proof π_2, that relies on the discrete logarithm assumption, or

- the unforgeability property of the signature scheme $\mathtt{S}$ and forge an attestation $\mathtt{att}(\widetilde{Id}_i, g_I^{u_i})$.

Assume that the adversary $\mathcal{A}$ wins the game. This means that $\mathcal{A}$ has output a valid transaction $T_{t,j}$ that is of the form either $T_{t,0} = (insert, DC_t, E_{t,1}, \pi_{2,t,1}, \mathtt{S}_{t,0})$ or $T_{t,j} = (transfer, \mathtt{amount}_{t,j}, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$ with $j \geq 1$, such that it cannot be traced by $\mathcal{RA}$ to the identity Id_i of the payer and/or payee. $\mathcal{A}$ can easily compute part of the transaction. However, it cannot replay or misuse a previously computed NIZK proof π_2 due to the way the proof's challenge is computed. Consequently, two cases are possible to win the game, either:

- *Case 1*: $\mathcal{A}$ has produced a valid NIZK proof $\pi_{2,t,k}$ related to a value $E_{t,k+1}$ such that $g_I^{u_i} \notin \mathcal{DB}_{Priv}$ (*i.e.* it has generated a false OR proof that is successfully verified). This implies that the adversary has broken the soundness property of $\pi_{2,t,k}$. Since π_2 is sound under the discrete logarithm (DL) assumption, then this case means that $\mathcal{A}$ has broken the DL assumption.
- *Case 2*: $\mathcal{A}$ has selected a random $u_i \in_R \mathbb{Z}_q^*$ and has produced a valid attestation $\mathtt{att}(\widetilde{Id}_i, g_I^{u_i})$ corresponding to the output of the signature scheme $\mathtt{S}$ on a fake identity $\widetilde{Id}_i$, without knowledge of the secret key $sk_{\mathcal{IDR}}$. Then, $\mathcal{A}$ has registered with $\mathcal{RA}$ using the forged attestation and used the associated $(u_i, g_I^{u_i})$ pair to compute the ElGamal encryption and NIZK proof π_2. This case implies that $\mathcal{A}$ has broken the unforgeability property of the signature scheme $\mathtt{S}$.

Thus, our $\mathcal{DCS}$ construction satisfies the accountability property under the DL assumption and the unforgeability property of the used signature scheme $\mathtt{S}$.

Proposition 4. *Our digital currency system $\mathcal{DCS}$ fulfills the end-user's privacy property under the Decisional Diffie-Hellman (DDH) assumption.*

Proof (sketch). Let us show that if an adversary $\mathcal{A}$ is able to break the end-user's privacy property of our construction, then it is possible to break the Decisional Diffie-Hellman (DDH) assumption.

Assume that the adversary $\mathcal{A}$ wins the game. This means that $\mathcal{A}$ has selected two honest users $\mathcal{U}_0$ and $\mathcal{U}_1$ and a third user $\mathcal{U}_R$ to run the challenge transactions (T_b, T_b^*) and (T_{1-b}, T_{1-b}^*) for $b \in \{0, 1\}$ which correspond to the transfers of the full amount from $\mathcal{U}_b$ to $\mathcal{U}_R$ and that $\mathcal{A}$ succeeded in linking at least one of the transactions to the correct $\mathcal{U}_b$. The challenge transactions are of the form:

- $T_b = (transfer, \mathtt{amount}_c, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}, E_{t,j+1}, \pi_{2,t,j+1}, \mathtt{S}_{t,j})$ which is a transfer from $\mathcal{U}_0$ to $\mathcal{U}_R$ where $E_{t,j+1}$ and $\pi_{2,t,j+1}$ are computed by $\mathcal{U}_R$;
- $T_b^* = (transfer, 0, \mathcal{H}(T_{t,j-1}), pk_{t,j+1}^*, E_{t,j+1}^*, \pi_{2,t,j+1}^*, \mathtt{S}_{t,j}^*)$ which is an auto-transfer from $\mathcal{U}_0$ to $\mathcal{U}_0$ where $E_{t,j+1}^*$ and $\pi_{2,t,j+1}^*$ are computed by $\mathcal{U}_0$;
- $T_{1-b} = (transfer, \mathtt{amount}_c, \mathcal{H}(T_{t',j-1}), pk_{t',j+1}, E_{t',j+1}, \pi_{2,t',j+1}, \mathtt{S}_{t',j})$ which is a transfer from $\mathcal{U}_1$ to $\mathcal{U}_R$ where $E_{t',j+1}$ and $\pi_{2,t',j+1}$ are computed by $\mathcal{U}_1$;
- $T_{1-b}^* = (transfer, 0, \mathcal{H}(T_{t',j-1}), pk_{t',j+1}^*, E_{t',j+1}^*, \pi_{2,t',j+1}^*, \mathtt{S}_{t',j}^*)$ which is an auto-transfer from $\mathcal{U}_1$ to $\mathcal{U}_1$ where $E_{t',j+1}^*$ and $\pi_{2,t',j+1}^*$ are computed by $\mathcal{U}_1$.

Two cases are possible to win the game, either:

- *Case 1*: $\mathcal{A}$ has linked a `Transfer`[11] or `Deposit` transaction to a given execution of the `Withdrawal` protocol through either the digital currency unique identifier $\mathtt{sn}_t$ or the public key $pk_{t,0}$. This implies that the adversary has broken the blindness property of the ACL construction which is proven under the DDH assumption in the random oracle model.
- *Case 2*: $\mathcal{A}$ has linked a `Transfer` transaction to another `Transfer` or `Deposit` transaction. Since the signatures included in T_b and T_{1-b} are computed using one-time use key pairs, then they cannot be leveraged to win the game. Thus, the only values that may enable $\mathcal{A}$ to win the game are the ElGamal encryptions $E^*_{t,j+1}$ and $E^*_{t',j+1}$, or the NIZK proofs $\pi_{2,t',j+1}$ and $\pi^*_{2,t',j+1}$. Thus, this case implies that the adversary has broken either (i) the security of the ElGamal cryptosystem which is proven under the DDH assumption, or (ii) the zero-knowledge property of the proofs π_2 which is ensured under the DDH assumption.

Thus, our $\mathcal{DCS}$ construction satisfies the end-user's privacy property under the DDH assumption.

[11] We recall that, for the sake of clarity, we consider that the `Insert` transaction is combined with the first `Transfer` transaction in our security and privacy model.