Experimental Design

March 18, 2024

1 Introduction

Both found offline e-cash schemes that provide transferability rely on combinations of extreme cryptographic protocols (malleable signatures, NIZK proofs and such) to provide transferability. However, if the DigiEuro were to be accepted by the public, those protocols might scare the masses. To prevent this, a solution that is less cryptographically extreme, and with that perhaps less efficient, might be preferred. Simplicity and understandability generate trust.

2 Design

2.1 Initialization

A trusted third party (TTP) responsible for handling identification (such as EBSI) publishes a bilinear pairing description and a Common Reference String (CRS) related to that description. The bilinear pairing description is defined as:

$$(G_1, G_2, G_T, p, e, g_1, g_2)$$

in which G_1 and G_2 are two different bilinear groups of order p. These groups have a mapping e to target group G_T. g_1 and g_2 are generators of respectively G_1 and G_2.

Every participant in the protocol also registers at the TTP and gets a private key x and public key g_1^x.
The CRS is defined as:

$$(g, u, g', u', h, v, h', v')$$

2.2 Withdrawal

When a user registers at the bank, the bank stores g^x to keep track of the user's balance. For this, the EBSI identification service can be used to prove the user's identity. Together with the bank, the user can receive a token and a (blind) signature from the bank.

2.3 Transaction

During a transaction, the user creates two GS proofs. Each proof has a specific target, commitment values c_1, c_2, d_1, d_2 and proof elements $\theta_1, \theta_2, \pi_1, \pi_2$. Only the first proof will be kept with the token. The second proof is used to verify that the user knows the secret key.
The receiver computes randomization factors h^s and v^s to the spender to prevent the spender from deciding on all the randomisation elements. These values are used in the commitments of d_1, d_2, containing no personal information of the spender.

1. The first proof is constructed with the bank's signature in the first transaction. The target for the first proof is computed as $e(g_1, g_2)^T$. The spender computes $y = \frac{x}{T}$, in which x is the spender's private key. Now the spender can create a proof for $e(g_1^x, g_2^y)$. Due to the property of bilinearity, this is the same as $e(g_1, g_2)^{xy}$, which equals $e(g_1, g_2)^T$. With the proof elements, the spender also sends g_2^y, the receiver can verify that the spender used its public key g^x in the proof. In the commitments, c_1, c_2, the public key of the spender is hidden. The target of this proof will be used in the next transaction, to compute the next target.

2. The second proof is meant to verify the spender's knowledge of the private key. the target of the proof is $x \cdot y$, note that the receiver does not know y and can thus not reverse compute x. With the proof the user sends the value of h'^y, to the receiver to verify that the values of x and y are used. The target of this proof can also be verified using $e(g_1, g_2)^{xy}$.

To prove ownership of the token at the current state, the receiver knows the value of s and can use that knowledge, to prove to the next receiver that the token belongs to him. The spender in the previous transaction cannot reuse the token, as the previous spender does not know s.

This same s will be used for the randomization in the next set of proofs. The randomizing value will be $-s$ for the next randomization as r. This way, the holder can prove that he knows the s used in the previous transaction, without revealing the exact value. The knowledge can then be verified by the equation $e(g_1^r, g_2^s) = e(g_1, g_2)$, as $e(g_1, g_2)^{rs} = e(g_1, g_2)^1$. This prevents users from removing some transactions from the chain to hide double-spending as they cannot generate proofs, without knowing the s used in the previous transaction. As the targets of the first proofs are created using the previous proof, they form a chain. Besides the current spending proof, the chain of proofs can also be validated. The same holds for checking the knowledge of s.

2.4 Deposit

A token can be deposited to the bank in the same way as a token is transferred between users in section 2.3. However in this case the bank is the receiver. As the user that wants to deposit the token has to share their public key, the bank knows to which account the balance should be added. The token, including the proofs, is then stored in a database.

2.5 Double Spending Detection

The bank can detect double spending if it receives a token with a signature already in the database. The bank can now loop over all the proofs accompanied by the tokens to find the commitments of the user that double spent. This user u_i can be found by finding the first proof where the two targets T_i are the same but have different commitment values for d_1 and d_2. Once the bank has found the two GS related to the double spending, the bank sends the full proofs to the TTP. The TTP checks if the two proofs are valid and can use the CRS to find the public key embedded in the commitments and the identity of the double-spender.