Offline Digital Euro: a Minimum Viable CBDC using Groth-Sahai proofs
— Master's Thesis —

Leon Kempen
Delft University of Technology
Delft, The Netherlands
L.M.Kempen@student.tudelft.nl

Johan Pouwelse
Delft University of Technology
Delft, The Netherlands
J.A.Pouwelse@tudelft.nl

Abstract— **Current digital payment solutions are extremely fragile and offer less privacy than traditional cash. Their critical dependency on an online service used to perform and validate transactions makes them void if this service is unreachable. Moreover, no transaction can be executed during server malfunctions or power outages. Due to climate change, the likelihood of extreme weather increases. As extreme weather is a major cause of power outages, the frequency of power outages is expected to increase. The lack of privacy is an inherent result of their account-based design or the use of a public ledger. The critical dependency and lack of privacy can be resolved with a Central Bank Digital Currency that can be used offline. This thesis proposes a design and a first implementation for an offline-first digital euro. The protocol offers complete privacy during transactions using zero-knowledge proofs. Furthermore, transactions can be executed offline without third parties and retroactive double-spending detection is facilitated. To protect the users' privacy, but also guard against money laundering, we have added the following privacy-guarding mechanism. The bank and trusted third parties for law enforcement must collaborate to decrypt transactions, revealing the digital pseudonym used in the transaction. Importantly, the transaction can be decrypted without decrypting prior transactions attached to the digital euro. The protocol has a working initial implementation showcasing its usability and demonstrating functionality.**

I. Introduction

The current infrastructure for digital payment systems is fragile because of their design to support instant payments or to obtain global consensus. However this infrastructure has two main concerns: it is required to be online and allows third parties, other than the payer and payee, to observe balance transfers. A lesser-known digital payment system, electronic cash (e-cash), mitigates these problems. Moreover, e-cash also provides instant payments without requiring a global consensus. Central banks around the world are attempting to use e-cash as a foundation for their Central Bank Digital Currencies (CBDCs). This thesis proposes a protocol for the digital euro, the CBDC of the European Central Bank.

For the past decade, the share of digital payments has increased and the number of cash payments has declined [1]. With that, the dependency on a connection to an online infrastructure during a transaction (digital payment) has increased, as these digital payment systems require a connection to such an infrastructure to complete the payment. For example, when you pay at a store with a debit or credit card, a connection to your bank is needed to verify whether you have enough balance to pay for the products or services you want to buy. Additionally, the balance must be transferred from the payer's account to the one of the payee. The dependency on an online infrastructure leads to the first main concern that the digital payment system is unusable whenever its online infrastructure cannot be contacted. The infrastructure could for example be unreachable in regions with no internet coverage, when the bank's servers are down or during a power outage. The latter is increasingly becoming a challenge as the number of power outages has increased for the past years [2] and is expected to increase further [3, 4].

The critical dependency on an online infrastructure for digital payments is not limited to banks, as cryptocurrencies have the same dependency. For example, in the case of Bitcoin [5], payments are stored in a data structure that is maintained by all participants that maintain the online infrastructure, a blockchain. This blockchain is used to verify if payments are valid and completed or not. If this blockchain cannot be reached, the receiver cannot verify if he correctly received the payment.

Cryptocurrencies that support offline digital payments, like Zcash [6], still require the spender to prepare the transaction online to be signed later. This requires spenders to know the exact transaction details up-front. These details include the receiver's account number (a ”wallet address”) and the value of the transaction. This makes the payment method impossible to use for unexpected transactions or transactions where the value cannot be determined up-front.

The second main concern for digital payment systems is that third parties can observe balance transfers. Current mainstream digital payment methods offer less privacy than cash payments. We consider digital payments through banks, cryptocurrencies and e-cash. We consider payments through banks. Banks have a complete list of all transactions involving the account holders and their balances. In case of a breach, a full list of transactions could be found and potentially abused. However, when using cash, the bank only knows who withdrew cash when or deposited cash when.

For most cryptocurrencies, the second type of digital payment system that we consider, payments are stored in a blockchain, using a wallet address as a pseudonym. A wallet address is a unique identifier of a cryptocurrency wallet that is used to retrieve and send cryptocurrencies. The wallet addresses of users may change or remain the same. For instance, in Ethereum [7], users have a fixed wallet address, whereas in Bitcoin [8] users can change the wallet addresses with every transaction. If you know which address belongs to someone, the payment executed with that wallet address can be traced and linked to their identity. Even with changing wallet addresses, payments and identities can be traced. By marking outgoing payments of a wallet address and any subsequent payments from the wallet address that received the payment, flows of payments between wallet addresses can be constructed. This flow of payments exposes links between payments and wallet addresses owned by a single identity. The method of marking payments is called taint analysis [9].

The last digital payment option that we consider is e-cash. e-cash offers users more privacy, compared to bank payments and cryptocurrencies. Similar to regular cash, a user must first withdraw e-cash from the bank, before it can be used. e-cash is represented as a digital token and can be stored on a device. At a later stage, the holder uses the token(s) to pay by transferring the tokens to the receiver. Finally, the receiver can deposit the tokens at the bank to redeem the value of the tokens.

The goal of e-cash is to provide properties similar to physical cash. There are three key properties: anonymity, unforgeability and unlinkability. In e-cash, anonymity is said to be provided when an attacker cannot link an identity to a payment [10]. Unforgeability states that it should be impossible for users to create fake tokens, that appear valid, in the issuer's name. Unlinkability states that it should be impossible to link any payment to a user, even if the user's identity is known.

In contrast to digital payment systems based on banking infrastructure or cryptocurrencies, some e-cash protocols also support offline payments. In a fully offline e-cash scheme, no bank, ledger, or other third party is involved in the payment between the spender and the receiver. Because e-cash provides fully offline privacy-preserving payments, it satisfies both main concerns (online infrastructure and exposed payments) of the current digital payment infrastructure. Therefore we explore it further in this work.

Many central banks have expressed their interest in e-cash and some central banks are already using e-cash as a digital version of their currency. These digital versions of currencies backed by a central bank are named Central Bank Digital Currencies (CBDCs). In December 2023, 130 countries, contributing to 98% of the global GDP, have expressed their interest in a CBDC, are researching and developing it, or have a CBDC in circulation [11]. A survey from the International Monetary Fund (IMF) [12] found that most CBDCs in development can only be used online. This implies that they have the same critical dependency on an online infrastructure as most other digital payment systems. Moreover, compared to online CBDCs, offline-capable CBDCs promote financial inclusion, have lower transaction costs and improve the user experience [13].

The CBDCs that can be used offline typically rely on tamper-resistant hardware to maintain the integrity of the CBDCs stored on a device [12]. However, as Liu et al. [14] and Lee et al. [15] have shown, even the current state-of-the-art hardware can be breached and thus fails to achieve integrity through tamper-resistance. Relying solely on hardware to prevent users from misusing the e-cash, is thus not sufficient. Therefore, the CBDC should guarantee integrity through software. This can be done using cryptographic protocols, such as proofs and signatures, to maintain the protocol's integrity, as is done in [16, 17, 18, 19, 20, 21]. This approach makes it possible to detect fraud retrospectively and to revoke malicious users their anonymity.

The European Central Bank (ECB) is currently in the 'preparation stage' of designing the (offline) digital euro [22]. Two of the main design goals of the digital euro are protecting privacy as much as possible and support for offline transactions [23]. This thesis will look at a software implementation of the digital euro, that fulfils the two design goals set by the ECB.

Following the approach of the Office of Science and Technology Policy [24] for a digital U.S. Dollar and the recommendation of the European Data Protection Supervisor [25], our implementation will use digital pseudonyms and zero-knowledge proofs. A zero-knowledge proof is a type of cryptographic proof used to prove knowledge of a secret, without revealing the secret. In our proposed protocol for the digital euro, zero-knowledge proofs are used as payment proofs. This way other users and banks can verify the proofs and the transaction without being able to identify the previous spenders when they receive a token. To further protect the users' privacy, participants in our system operate under digital pseudonyms.

This thesis proposes a design for the payment protocol of the digital euro that supports fully offline payments, whilst protecting the users' privacy. The protocol relies on zero-knowledge proofs and signatures to protect its integrity and to provide retrospective fraud detection. To demonstrate the protocol's correctness and functionality, the protocol is also implemented in an open-source minimum viable product. More specifically our contributions are as follows:

1) We propose a payment protocol that supports fully offline privacy-preserving payments and could be used by the ECB as implementation for the digital euro. The protocol uses zero-knowledge proofs and signatures providing privacy and retrospective fraud detection (Section VI).
2) We provide an open-source minimum viable product of the protocol to demonstrate the correctness and functionality (Section VII).

II. Problem description

Balancing privacy and fraud prevention, double spending and token transferability are three major problems regarding e-cash. The first major problem is the trade-off between privacy and fraud prevention. Fraud can be trivially detected and prevented by making the transactions traceable and fully linkable to users. This would, however, require all participants in the system to give up their privacy and reveal sensitive information regarding their spending behaviour. When the protocol is implemented this way, the e-cash has the same concern regarding privacy preservation as banking payments and payments done with cryptocurrencies.

The second problem, double-spending [19, 20], results from having too much privacy. Receivers of an e-cash token cannot check if the same token was used in an earlier payment. The spender could thus copy a single token and spend the copy at multiple places. In an analogy to cash, this would mean copying a banknote and spending that forged banknote at different locations. In a fully anonymous setting, a third party cannot link an identity to the fraudulent payment. This makes the e-cash scheme more vulnerable when too much privacy is given to the users.

Currently available solutions to the problem of double-spending, show a necessary restriction to our system's implementation. As mentioned earlier, the secured hardware can be breached, thus the implementation is restricted to retrospective double-spending detection and anonymity revocation. As an additional restriction, it should be possible to complete fully offline transactions. This means that the double spending detection cannot rely on third parties involved in the transaction.

The third problem is token transferability. Token transferability is a property of e-cash that allows users to reuse a token they received in an earlier payment. This means that a token can have multiple holders, and be used in multiple transactions, before being deposited. To satisfy unlinkability, token holders should not be able to find any information regarding the identity of previous holders in the token.

Combining token transferability with anonymity revocation introduces a new challenge as tokens can have multiple holders. To find the correct identity that committed double spending, details of every transaction must be included with the token. This is because the transaction details include the identity of the spender. As a result, the size of the token grows with every transaction [26].

III. Related work

Since the introduction of blind signatures in 1983 by Chaum [27] and the first offline e-cash protocol by Brands [16], there has been little (recent) research on offline e-cash schemes that do not rely on hardware or trusted software to prevent double-spending. Relying on such hard- or software to avoid double-spending, such as is done in [28, 29, 30, 31, 32], is an unrealistic assumption and breaking this hard- or software's integrity would also invalidate the entire e-cash scheme.

The other method to prevent double-spending is to rely on cryptographic principles to detect double spending and revoke the anonymity of the malicious user. This occurs when the tokens are deposited at the bank [16, 17, 18, 19, 20, 21]. Whenever the bank detects two tokens with the same identifier, the transaction details can be used to revoke the identity of the double-spender.

A subset of the protocols that rely on cryptography is unpractical as it allows shops to mint tokens ([33]) or non-functional in a fully offline scenario ([34]), especially when intended to be used as a basis of a CBDC (Section III-A). Moreover, other research regarding offline e-cash introduces functionality (token expiration [35, 17, 36]), which does not solve and potentially worsens the problem it was intended to solve (Section III-B).

To fully benefit from the offline functionality, token transferability (Section III-C) is highly desirable. With transferable tokens, it is not required to withdraw and deposit a token for each payment. Having the possibility to spend tokens multiple times without the bank's involvement would thus reduce the communication costs between the bank and the users [19]. A prototype of an offline transferable digital euro, EuroToken, was proposed earlier [37, 38] (Section III-D). However, this proposed scheme is fully traceable and offers little privacy to the users and thus conflicts with one of the main design goals (preserving privacy) set by the European Central Bank [23].

A. *Real world (un)useabilty*

Besides the integrity of the e-cash scheme, its useability must be considered. Some proposed e-cash schemes make assumptions or functionality that are infeasible or undesirable. For instance, Osipkov et al. [34] claim to prevent double-spending without trusted hard- or software in an offline setting. However, they make use of the assumption that the merchants (receivers) have a functional peer-to-peer network. This scenario, where the network is partially offline does not offer a solution to pay in areas with no network coverage. Furthermore, the peer-to-peer network does not satisfy our fully offline constraint.

Another example is the scheme proposed by Batten et al. [33], which provides spare change by giving reputable shops, such as Target, the authority to mint cash. Given that our protocol is intended to be used for a CBDC, it is undesirable that shops can mint tokens.

B. *Expiring e-cash*

Eslami and Talebi [35] introduced expiring e-cash by attaching an expiration date to the e-cash description. This scheme was later improved by [36] and [17]. The main reason for this expiration date is a storage reduction for the bank.

The question remains if schemes, that support token expiration, solve the problem of storage required to detect double-spending. The option to recover expired e-cash will not lead to a decrease in transactions, given that users withdraw e-cash intending to spend it. Whether or not the e-cash has an expiration date, users would need to pay for the same amount of goods or services, the same number of times. Therefore, the number of deposits does not change. This means that the storage size needed to store all deposited tokens will not be reduced by adding an expiration date. Tokens that have been deposited and expired after can not be removed from the storage, because they are needed to check if a token has been spent when it is sent for exchange, as is designed in [35, 36, 17]. Furthermore, by offering an exchange service for expired tokens, the bank should store the exchanged tokens leading to a larger required storage to detect double-spending. Given that adding an expiration date to tokens increases the storage size required, our protocol does not include token expiration.

C. Transferable e-cash

Transferability is a highly desired property that e-cash should have. Transferable e-cash makes it possible to spend the e-cash received by other users without depositing and withdrawing new e-cash first. This reduces the dependency on reaching the bank even further and does not limit the number of transactions to the number of withdrawals.

The downside of transferability is that it requires the e-cash to grow in size with each transaction. This is because storing information about every transaction is needed to reveal the double spender's identity [26].

Sarkar [39] tried to achieve this property using bitwise XORs. However, the protocol uses an unspecified distributive operator over XOR to detect double-spending [40]. Furthermore, Barguil [40] also proves that the security claims made by Sarkar do not hold.

Baldimtsi et al. [19] proposed a transferable e-cash scheme using malleable signatures. This type of signature is used to sign transactions whilst keeping the bank's signature valid. Double-spending is detected by making use of cryptographic tags. For each transaction, a *double spending (DS) tag* and a *serial number (SN) tag* are created and added to the token during a transaction. The SN tag is used in the next transactions as the identifier of the token and is computed using the DS tag of the previous transaction. The DS tag is constructed using the SN tag used in the transaction and is used by the bank to detect double-spending. A bank could detect double spending upon deposit when it receives two tokens, having the same SN tag with a different DS tag.

The protocol of Baldimtsi et al. is improved by Bauer et al. [20] by replacing the inefficient malleable signatures with a commit-and-proof scheme. With this scheme, the tags to detect double spending are also randomized in each transaction. Our protocol has a similar approach with a commit-and-proof scheme. However rather than using two tags, proofs are added to the token to detect double-spending.

Jianbing et al. [21] tried to take the transferability one step further by proposing a transferable e-cash scheme that allows the receiver to be anonymous and thus provides dual anonymity. However, they used a different definition of transferable, as the protocol requires users to contact the bank to re-randomize a received token after each transaction. Having to re-randomize a token at the bank before you can reuse it, removes the benefits (reducing the dependency on the bank) of transferable e-cash.

D. Eurotoken

Blokzijl [37] and Koning [38] of the Tribler Lab [1] and the Delft University of Technology did earlier work regarding a CBDC for the ECB, named Eurotoken. This work was done in collaboration with the Nederlandsche Bank. This thesis serves as a continuation of their research.

In the scheme of Blokzijl and Koning, the bank mints a token by defining a serial number, a face value and a nonce. Upon withdrawal, the bank sends the user the minted token, a tuple of the receiver's public key and a signature of the bank on the minted token and the receiver's public key.

The signature tuple is the start of a chain of proofs of ownership. This chain of ownership is sent with the token and is extended with each transaction. As the bank's signature includes the withdrawer's public key, the withdrawer can prove he owns the token. When the user spends the token, the user will send the token and extend the chain of ownership with a tuple of the receiver's public key and a signature, singing the previous proof of ownership and the recipient's public key. The deposit of the token is similar to a transaction between users. However, now the bank is the receiver of the token. After the bank has received the token it can verify the proof chain and check for double-spending.

Token holders can verify the chain of ownership after k transactions starting from the bank's signature. This signature can be used to find the public key of the first receiver. The found public key can then be used to validate the next proof and to find the next recipient's public key. After k transactions the last found public key maps to the current holder of the token.

The bank can detect double spending upon deposit of the tokens. Whenever the bank has received two tokens with the same first proof double spending must have occurred. The bank can then compare the chain of proofs of ownership to find the double spender. After some i proofs there must be two proofs where proof $i+1$ from the first chain differs from proof $i+1$ from the second chain. This implies that proof i is used in two transactions and thus doubly spent. The identity of the double spender can then easily be found, as that is the receiver's public key used to create proof i.

The problem with this proposal is that it offers no privacy and the token's history is fully traceable. Whenever someone receives a token, all the public keys of the previous holders can be found. Malicious people who know which public keys map to which identity could use and abuse that information to obtain sensitive personal information when receiving a token.

Privacy is an important factor in why people use cash for payments [41]. The current implementation of Eurotoken offers less privacy than the online payment infrastructure of banks. This combined will have a detrimental effect on the adoption rate of the CBDC, as the option of paying offline will come at the cost of user privacy. Moreover, the provided protocol does not align with the main design goal of the ECB, namely privacy protection [42].

IV. Security assumptions

The protocol proposed by this thesis relies mainly on two security assumptions to guarantee unforgeability and anonymity.

[1] https://github.com/Tribler/tribler/wiki#current-items-under-active-development

These assumptions are the *Discrete logarithm problem* and the *Computational Diffie-Hellman* assumption. The Discrete logarithm problem states that given a finite cyclic group G, generator $\langle g \rangle$ of G and $h \in G$, it is hard to find an integer a, such that $g^a = h$. This hardness will be used to create unforgeable signatures and proofs of ownership.

The Computational Diffie-Hellman assumption states that given a finite cyclic group G, generator $\langle g \rangle$ of G, g^a and g^b, it is computationally hard to compute g^{ab}, without knowing the values of a and b. This assumption is used to verify knowledge of the private key and as a basis for the security of bilinear pairing cryptography.

V. Signatures and Groth-Sahai proofs

Our system has two main cryptographic components, blind signatures and Groth-Sahai proofs. The blind signature is applied to prevent the bank from linking the withdrawn digital euro to the first holder. The Groth-Sahai proofs are used to create a zero-knowledge proof of a transaction, to provide anonymity between transactions. These proofs are constructed with bilinear pairings.

A. Blind signatures

Chaum [27] first introduced blind signatures in 1983. A blind signature scheme can be used to obtain a valid signature on a message M, without the signer knowing the exact content of M. This makes it possible for e-cash to have a valid signature of a bank for an unknown token. When this token is deposited later, the bank cannot recognize which user has withdrawn the token. This makes it impossible for the bank to link the user who withdrew the token to the user who deposited it, proving anonymity. In this thesis, an implementation of a hash-based blind Schnorr signature (BSS) is used.

As the (blinded) Schnorr signature scheme is based on groups, there should be a group g with order q known by both the client (a user) and the signing party (the bank). Furthermore, the signing party chooses a random private key $x \in_R \mathbb{Z}_q^*$ and shares the public key $y = g^x$ and a cryptographic hash function H with the clients. A BSS on message M can then be obtained as follows:

1) The signing party chooses a random $k \in_R \mathbb{Z}_q^*$ and sends $r = g^k$ to the client.
2) The client picks random blinding factors $\alpha, \beta \in_R \mathbb{Z}_q^*$ and calculates r' as $r' = rg^{-\alpha}y^{-\beta}$.
3) With that the client computes the challenge c for message M: $c = H(r'||M) \mod q$, and sends blinded challenge $c' = c + \beta$ to the signing party.
4) The signing party then signs the blinded message as: $\sigma' = k - c'x$ and returns σ'.
5) To obtain the signature on message M the client computes: $\sigma = \sigma' - \alpha$. The Schnorr signature is then defined as (σ, c)
6) Other parties can verify the validity of the signature on message M by computing $r_v = g^\sigma y^c$ and checking: $c \stackrel{?}{=} H(r_v||M)$.

A mathematical protocol description of the BSS protocol can be found in Figure 1.

$$\begin{array}{lcr}
\text{Client} & & \text{Signing party} \\
 & & k \in_R \mathbb{Z}_q^* \\
 & & r \leftarrow g^k \\
 & \xleftarrow{r} & \\
\alpha, \beta \in_R \mathbb{Z}_q^* & & \\
r' \leftarrow rg^{-\alpha}y^{-\beta} & & \\
c \leftarrow H(r'||M) & & \\
c' \leftarrow c + \beta & & \\
 & \xrightarrow{c} & \\
 & & \sigma' \leftarrow k - c'x \\
 & \xleftarrow{\sigma'} & \\
\sigma \leftarrow \sigma' - \alpha & &
\end{array}$$

Fig. 1: Blind Schnorr signature protocol to obtain a blind signature (σ, c) on message M

The blind signature is done over the hash of the message to prevent malicious clients from creating more valid signatures from an earlier received signature. Due to the multiplicative homomorphic property, malicious clients could also compute valid signatures on multiples of message M without the hash.

Given that the hash function is collision-resistant, it is hard for a malicious client to find the message corresponding to the malled signature. Therefore it is impossible to create more valid signatures, based on an earlier received signature.

B. Bilinear pairings

A bilinear map e is an operation that takes two elements from, potentially, different elliptic curve groups of order p and maps them to an element of a third group, the target group. More formally, given source groups G, H and target group G_T, a bilinear map is denoted as:

$$e : G \times H \to G_T$$

Additionally, the pairing must satisfy the following three properties:

- **Bilinearity:** For all items $P, Q \in G$ and $R, S \in H$, the following holds:

  $$e(P + Q, R) = e(P, R) \cdot e(Q, R)$$
  $$e(P, R + S) = e(P, R) \cdot e(P, S)$$

  Moreover, given generators g, h such that $G = \langle g \rangle$ and $H = \langle h \rangle$, for all $a, b \in \mathbb{Z}$, the following holds:

  $$e(g^a, h^b) = e(g, h)^{ab}$$

- **Non-degeneracy:** $e(P, R) \neq 1$.
- **Efficient computability:** There must be an efficient method to calculate the pairing efficiently.

An extended bilinear map E is a mapping of two elements of G and two elements of H to four elements of G_T:

$$E : G^2 \times H^2 \rightarrow G_T^4$$

As an example, given $g_1, g_2 \in G$ and $h_1, h_2 \in H$:

$$E\left(\begin{pmatrix} g_1 \\ g_2 \end{pmatrix}, \begin{pmatrix} h_1 & h_2 \end{pmatrix}\right) = \begin{pmatrix} e(g_1, h_1) & e(g_1, h_2) \\ e(g_2, h_1) & e(g_2, h_2) \end{pmatrix} \quad (1)$$

Similarly to regular bilinear maps, the extended bilinear maps are also bilinear, using entry-wise product operations for the vectors and matrices. Given $g_1, g_2, g_3, g_4 \in G$ and $h_1, h_2 \in H$:

$$E\left(\begin{pmatrix} g_1 \\ g_2 \end{pmatrix}\begin{pmatrix} g_3 \\ g_4 \end{pmatrix}, \begin{pmatrix} h_1 & h_2 \end{pmatrix}\right) = E\left(\begin{pmatrix} g_1 \\ g_2 \end{pmatrix}, \begin{pmatrix} h_1 & h_2 \end{pmatrix}\right) E\left(\begin{pmatrix} g_3 \\ g_4 \end{pmatrix}, \begin{pmatrix} h_1 & h_2 \end{pmatrix}\right)$$

C. Groth-Sahai proofs

In 2008, Groth and Sahai [43] presented a proof framework that can be used to efficiently create non-interactive zero-knowledge (NIZK) proofs and non-interactive witness-indistinguishable (NIWI) proofs. Before this, NIZK proofs were inefficient and thus not useable. The Groth-Sahai (GS) proofs are designed to prove statements in pairing-based equations.

As a setup, a (trusted) party must publish a bilinear pairing description and a Common Reference String (CRS).

The bilinear pairing description is defined as:

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

in which G_1 and G_2 are two bilinear groups. 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. When $G_1 \equiv G_2$ the pairing is symmetric and if $G_1 \not\equiv G_2$ the pairing is asymmetric. Functionally, both types of pairings would work for our protocol. Symmetric pairings allow for an easier protocol description [44]. However, asymmetric implementations tend to perform better on higher security levels.

The CRS is constructed with two pairs of four random group elements, four from G_1 and four from G_2 and is defined as:

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

Depending on the structure of the GS proofs, the CRS can be used in a trapdoor function. In some proof structures, this will reveal the input and can thus be used to find the values used to create the committed value (the value to be proven). In our protocol (Section VII) this is used to find the public key of the spender. The setup can be done with public randomness and multiple parties to fully remove the trust needed in a (central) party. Each proof consists of three parts, namely the target T, the commitment values c_1, c_2, d_1, d_2 and proof elements $\theta_1, \theta_2, \pi_1, \pi_2$. The target represents the value that the prover wants to prove. The commitment values are used to randomized encryptions of values with which the proof is constructed. Elements from G_1 are encrypted in c_1 and c_2, whereas elements from G_2 are encrypted in d_1 and d_2. Lastly, the proof elements are used to derandomize the commitment values without revealing the exact values.

In our protocol, the implementation of the Groth-Sahai proofs is as follows. The equation to prove is $e(X, Y) = T$ in which $X \in G_1$ and $Y \in G_2$ and T is the target of the proof. The commitment values are randomized with values $r, s \in Z_p$, and computed as:

$$\begin{array}{ll} c_1 = g_1^r & d_1 = g_2^s \\ c_2 = u^r X & d_2 = v^s Y \end{array}$$

The prover now picks a random value $t \in Z_p$ and computes the proof elements as:

$$\begin{array}{ll} \theta_1 = g_1^{-t} & \pi_1 = d_1^r g_2^t \\ \theta_2 = X^s u^{-t} & \pi_2 = d_2^r v^t \end{array}$$

The full proof is now defined as $(c_1, c_2, d_1, d_2, \pi_1, \pi_2, \theta_1, \theta_2)$ and can be verified by others with Equation 2.

$$E\left(\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, (d_1, d_2)\right) \stackrel{?}{=} E\left(\begin{pmatrix} g_1 \\ u \end{pmatrix}, (\pi_1, \pi_2)\right) E\left(\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, (g_2, v)\right) \begin{pmatrix} 1 & 1 \\ 1 & T \end{pmatrix} \quad (2)$$

The verification can be done elementwise after expanding the extended bilinear maps as in Equation 1. For example, to verify $e(c_1, d_1)$, the following must hold:

$$e(c_1, d_1) \stackrel{?}{=} e(g_1, \pi_1) \cdot e(\theta_1, g_2) \cdot 1$$

If someone knows the exponents used to create u and v from the CRS, one could find the committed values of X and Y. Let $u = g_1^\alpha$ and $v = g_2^\beta$, the committed values can be retrieved with the equations 3a and 3b.

$$X = c_1^{-\alpha} c_2 \quad (3a)$$

$$Y = d_1^{-\beta} d_2 \quad (3b)$$

VI. Digital euro protocol

Our protocol is divided into four phases: initialization, withdrawal, transactions and deposit. The initialization phase is executed only once by a trusted third party (TTP), the bank and the users. The other three phases are related to the cycle of a single digital euro.

A. Initialization

In the initialization phase, the TTP responsible for managing identification publishes a bilinear pairing description and a common reference string (CRS), as found in section V-C. The exponents used to generate the group elements are stored for later use by the TTP but remain private. The participants in the protocol will use the bilinear pairing description and CRS.

Every participant has to register at the TTP as well. Upon registering the user picks a random private key x, calculates the public key $X = g_1^x$ and registers X at the TTP. The user can register at a bank with the public key, certified by the TTP. The bank can use this public key to keep track of the user's balance.

B. Withdrawal

At the start of the withdrawal phase, the user can prove his identity to the bank in the same way as during the initialization phase. After that, the BSS protocol (Section V-A) is used with the generator g_1 of order p of the bilinear group description provided by the TTP.

The message to be signed consists of the serial number and a random group element. The withdrawer can generate a serial number randomly. This serial number is used for tracking but does not need to be unique however the combination with the random element should be unique. For the random group element, the user picks a value $t \in_R \mathbb{Z}_p^*$ and computes $\theta_1 = g_1^{-t}$. This t will be later used in a transaction to demonstrate knowledge of randomization. The serial number and θ_1 can then be converted to bytes and concatenated to be blindly signed by the bank. When the protocol is completed the digital euro is described as:

$$(SN, \theta_1, \sigma, GS)$$

in which, SN is the serial number of the digital euro, θ_1, σ is the blind signature of the bank on SN and θ_1 and GS is an ordered list of Groth-Sahai proofs of previous transactions. Upon withdrawal GS is empty.

C. Transactions

Every transaction a digital euro has undergone must be stored with the euro to combat double-spending. To find the user that double-spent a euro, the details of the malicious transaction must be known to retrieve the identity of the double-spender, as shown in [26]. This scheme stores the required information as a GS proof. By storing the information in a zero-knowledge proof, participants in later transactions, or the bank, cannot deduce any information related to the transaction from the proof. They can, however, verify if the proofs and thus the transactions are valid. During a transaction, the spender and the receiver collaborate to create a GS proof, which is stored with the digital euro.

To start a transaction the receiver generates a random t and sends the randomization elements g_2^t, v^t, g_1^{-t} and u^{-t} to the spender, whilst keeping t secret. This prevents the spender from deciding on all randomness and trying to obfuscate double-spending by using the same randomness for two transactions with the same digital euro. When the same randomization is used in two transactions with the same token, the proofs for those transactions are the same, hiding the double-spending. The spender will use these randomization elements given by the receiver when creating the GS proof for the transaction.

The target of the proof for transaction i, T_i, depends on whether the digital euro is spent earlier. When the euro has not been spent before, the target is $T_0 = e(g_1, g_2)^\sigma$. Otherwise, after i transactions the target can be computed as $T_i = e(g_1, g_2)^{T_{i-1}}$. This way, the targets of the proofs can be used to describe a chain of transactions, in which the current proof links to the previous proof.

With this target, the spender can compute $y = \frac{T_i}{x}$ and $Y = g_2^y$, in which x is the spender's private key. The spender can now use the GS proof, to prove $e(g_1^x, g_2^y)$. Note that g_1^x is equal to the spender's public key. Additionally, due to the property of bilinearity, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy} = e(g_1, g_2)^{T_i}$.

The value of s in the proof is set to the inverse of t_{prev}, the t used in the previous transaction to provide the randomization elements. This implies that the spender must know the value of t used during the last transaction and cannot generate a valid proof if he does not. For the first transaction, no t_{prev} is available. However, the spender in the first transaction can use the t used in the withdrawal phase as he is the withdrawer.

To prevent the receiver from creating valid proofs by changing the values of t after the transaction, the spender also computes an additional signature. This is a Schnorr signature constructed with signing key r used to create GS proof and signs the value of g_1^{-t}. This signature only has to be shown in the next transaction. The next receiver can verify this signature as the decryption key g^r is provided in the GS of the current transaction as c_1.

The spender sends the values of v^s and Y together with the proof elements, the signature received in the previous transaction and the signature of the current transaction to the receiver. With these, the receiver can verify the proof, if $e(X, Y) = T$, check if d_2 is constructed correctly and verify the signatures.

Additionally, the receiver must check if the previous proofs included with the digital euro are correct and verify the links between the proofs. Given the proofs for transaction $i-1 = j$ and i as:

$$(c_{1j}, c_{2j}, d_{1j}, d_{2j}, \theta_{1j}, \theta_{2j}, \pi_{1j}, \pi_{2j}, T_j)$$

and

$$(c_{1i}, c_{2i}, d_{1i}, d_{2i}, \theta_{1i}, \theta_{2i}, \pi_{1i}, \pi_{2i}, T_i)$$

the equations 4a and 4b must hold:

$$T_i \overset{?}{=} e(g_1, g_2)^{T_j} \tag{4a}$$

$$e(\theta_{1j}, d_{1i}) \overset{?}{=} e(g_1, g_2)^1 \tag{4b}$$

Equation 4b must hold to verify that every spender knew the randomization element t in the previous transaction. As g_1 and g_2 are part of the bilinear pairing description and thus constant, the equation expands to $e(g_1^{-t_j}, g_2^{s_i})$, which is equal to $e(g_1, g_2)^{-t_j s_i}$. For the transaction to be valid s should be the inverse of t of the previous transaction, implying that $-t_j s_i = 1$. This results in the verification form $e(g_1, g_2)^1$.

D. Deposit

A digital euro can be deposited to the bank in the same way as a digital euro is transferred between users in section VI-C. However, in this case, the bank is the receiver. As the user that wants to deposit the euro has to share their public key, the bank knows to which account the balance should be added. The bank also checks if the digital euro is doubly spent or not.

E. Double spending detection

The bank detects double spending when two digital euros DE and DE' with the same signature σ_{sn} are deposited. There are two possible scenarios in this case.

The first trivial case is when GS of DE equals GS of DE', excluding the last proof created in section VI-D. This occurs if, and only if, the same user tries to deposit the same digital euro twice. To deposit the euro the user must identify himself, therefore the identity of the double spender is revealed.

In the second scenario, when GS of DE does not equal GS of DE', the bank must take additional actions to reveal the identity of the double spender. Given that the two lists of proofs are different, there must be an index i, such that $GS_{DE}[i] \neq GS_{DE'}[i]$. Assuming that the odds that the double spender retrieved the randomization elements generated by the same t are extremely unlikely, the proofs have, at least, different values for the θ_1 and θ_2 proof elements.

The bank can then send both proofs to the TTP. The TTP can extract the public key X with equation 3a, for both proofs and check if X is the same for both proofs sent by the bank. If they are the same, the TTP can retrieve the legal identity, registered with this public key, and return it to the bank. Otherwise, this transaction is no occurrence of double-spending. This could for example occur when the double spender did receive the same randomization parameters.

F. Efficiency analysis

As mentioned earlier, the size of the digital euro must grow to detect double spending and revoke the anonymity of the double-spender. As seen in section VI-C, every transaction included in the digital euro is defined in a GS-proof. This means that the size of the digital euro grows with 8 or 9 group elements for each transaction. The number of group elements depends on whether the value of T is explicitly included in the proofs. Given that the target T can be calculated from the proof elements of the previous proof, it can be omitted for size optimizations. This means that the size of the digital euro after n transactions (using a symmetric pairing) can be computed as:

$$size = |SN| + |G| + |\sigma| + n \cdot 9|G| \tag{5}$$

in which $|SN|$ denotes the size of the serial number, $|\sigma|$ the size of the signature of the bank and $|G|$ the size of a group element.

VII. Implementation

The described protocol is implemented in Kotlin as a proof of concept. The open-source implementation can be found on GitHub [2]. The Java Pairing Based Cryptography (JPBC) library [45] is used for group and bilinear map operations. As this is a proof of concept, it is not a fully implemented financial system and users can freely withdraw and deposit digital euros without affecting their balances. This does not affect the results as the balance of the user's bank account does not influence the digital euro protocol. This prototype was used to test the protocol for correctness, growth size and verification performance. The tests were performed on a desktop with an Intel Core i5-4590 (3.30GHz) processor and 8 GB of RAM.

Curve	$\lvert G_1\rvert$	$\lvert G_2\rvert$	$\lvert G_T\rvert$	$\lvert GS\rvert$
Type A	128	128	128	1152
Type E	256	256	256	2304
Type F	40	80	240	720

TABLE I: Size of group elements and a single GS proof in bytes per curve type ($r = 160$).

A. Prototype

The prototype is built as a mobile application, in which participants can select their role (TTP/Bank/User). Screenshots of the running application can be seen in Figure 2. For demonstration purposes, users (Figure 2c) can easily double-spend digital euros.

The device communication is implemented using the Kotlin implementation of the peer-to-peer communication protocol IPv8[3]. This library is also used during the transaction between users. Even though this library uses an internet connection, the (authenticated) messaging is done directly between two peers without a third party involved. As the library is solely used to transfer bytes between devices, the prototype can be extended with offline communication protocols, such as QR codes, Bluetooth or NFC. As our fully offline constraint required transaction verification without needing a third party, the constraint is still satisfied and guaranteed this way.

B. Different curves using JPBC

JPBC can be configured to use different types of underlying elliptic curves. This difference is the equation used to generate the bilinear map. Moreover, with JPBC it is also possible to set the security parameter giving more flexibility regarding the size of the group elements. The curves and their properties are listed in [46]. The implementation has been tested with multiple underlying elliptic curves, both symmetric and asymmetric, and security parameters. For the different tested parameters, the protocol remained functional. This shows that the protocol is not tied to a specific curve type or security parameter.

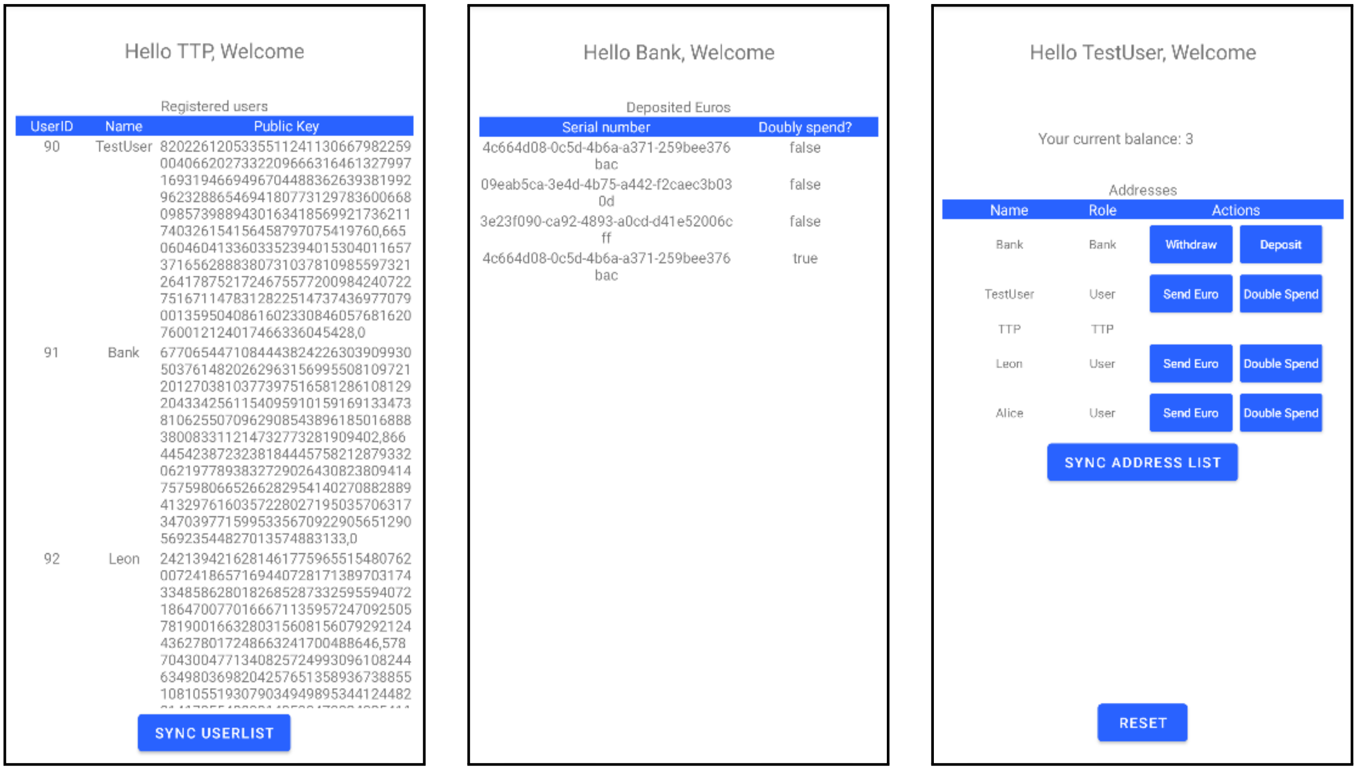
C. Growth in size

As mentioned earlier, the size of the digital euro must grow for each transaction. The growth size depends on the elliptic curve and security parameter used. To test the difference in growth, a test is done to measure the size of the digital euro after each transaction, used in 50 transactions.

Three elliptic curves, A, E and F, were used for the test. To construct those curves, the curve generator of JPBC was

[2] https://github.com/LeonKempen/trustchain-superapp/tree/master/offlineeuro

[3] https://github.com/Tribler/kotlin-ipv8



Hello TTP, Welcome

Registered users

UserID	Name	Public Key
90	TestUser	820226120533551124113066798225900406620273322096663164613279971693194669496704488362639381992962328865469418077312978360066809857398894301634185699217362117403261541564587970754197 60,66506046041336033523940153040116573716562888380731037810985597321264178752172467557720098424072275167114783128225147374369770790013595040861602330846057681620760012124017466336045428,0
91	Bank	677065447108444382422630390993050376148202629631569955081097212012703810377397516581286108129204334256115409591015916913347381062550709629085438961850168883800833112147327732819094 02,86644542387232381844457582128793320621977893832729026430823809414757598066526628295414027088288941329761603572280271950357063173470397715995335670922905651290569235448270135748831 33,0
92	Leon	242139421628146177596551548076200724186571694407281713897031743348586280182685287332595594072186470077016667113595724709250578190016632803156081560792921244362780172486632417004886 46,578704300477134082572499309610824463498036982042576513589367388551081055193079034949895344124482

SYNC USERLIST

(a) TTP

Hello Bank, Welcome

Deposited Euros

Serial number	Doubly spend?
4c664d08-0c5d-4b6a-a371-259bee376bac	false
09eab5ca-3e4d-4b75-a442-f2caec3b030d	false
3e23f090-ca92-4893-a0cd-d41e52006cff	false
4c664d08-0c5d-4b6a-a371-259bee376bac	true

(b) Bank

Hello TestUser, Welcome

Your current balance: 3

Addresses

Name	Role	Actions	
Bank	Bank	Withdraw	Deposit
TestUser	User	Send Euro	Double Spend
TTP	TTP		
Leon	User	Send Euro	Double Spend
Alice	User	Send Euro	Double Spend

SYNC ADDRESS LIST

RESET

(c) User

Fig. 2: Screenshot of the application per role

Curve	Initial size (kB)	50 transactions (kB)	Growth (kB)
Type A	0.567	63.217	1.248
Type E	0.823	114.673	2.272
Type F	0.391	41.441	0.816

TABLE II: Digital euro growth with transactions measured in serialized bytes ($r = 160$).

used, with the default settings [4]. Pairings of Type A and E are symmetric and the ones of Type F are asymmetric. Curves of Type A are used in Charm [47] and curves of Type F are BN-curves, introduced in [48]. BN-curves are used in Ethereum and Zcash [49]. As curves of Type E have larger elements, compared to Type A, and are less optimized, they are used less often [50]. The sizes of the group elements [50] can be found in Table I. The size of the GS proof is computed as $4 \cdot |G_1| + 4 \cdot |G_2| + |G_T|$. During the test, the prime order used (r) remained constant ($r = 160$).

Table II shows that the growth rate of the digital euro significantly depends on the elliptic curve. The asymmetric pairing (Type F) has the lowest growth rate, resulting from the smaller element sizes as found in Table I. Due to the size of the elements, the digital euro will grow faster when elliptic curves of Type E are used, compared to curves of Type A.

The growth of the digital euro is constant for each transaction, hence the growth is linear to the number of transactions. In Figure 3, the size of the digital euro is plotted after each transaction. The plot shows that the digital euro grows linearly with each transaction as expected from Equation 5.

Comparing the size of the expected growth ($|GS|$ from Table I) and the actual growth per transaction (Growth in Table II), are similar in byte sizes. The difference between the expected and actual growth can be a result of serialization, overhead of the used list structure or how group elements are converted into bytes by JPBC. The first transaction, however, has a slightly larger growth (+250 bytes) due to the initialization of the list structure for the transaction proofs.

D. Transaction verification performance

The time it takes to verify a transaction is a major factor in adopting digital currencies and their usability in everyday transactions. For example, on average, a Bitcoin transaction takes 10 minutes to confirm and waiting for more confirmation blocks for larger transactions is recommended. As financial transactions are expected to be completed nearly instantly [51], this payment option is unusable in most scenarios.

To test the transaction verification performance, a test measures the time it takes to verify a transaction using a digital euro for up to fifty transactions. To verify the final transaction the user must check three Schnorr signatures and the chain of fifty proofs.

The test is set to measure the time it takes to verify a transaction. The digital euro is then used to create the next transaction details. This experiment is done using the

[4] http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html

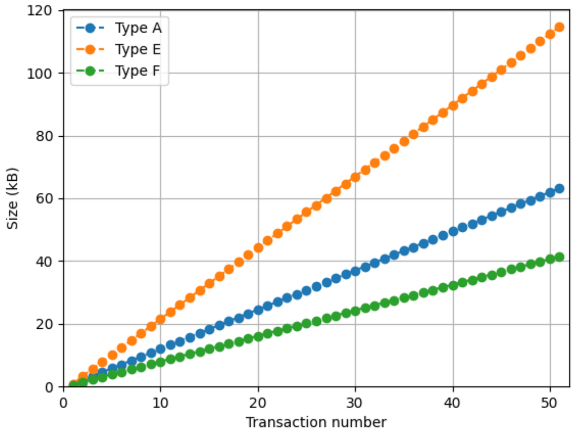


Fig. 3: Serialized digital euro growth per transaction ($r = 160$).

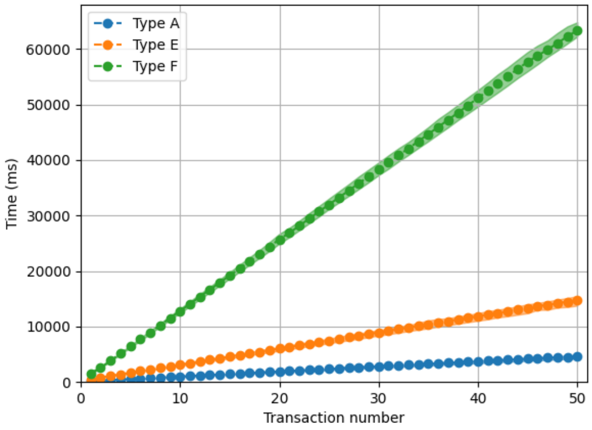


Fig. 4: Transaction verification time ($r = 160$).

same curves as mentioned in Section VII-C. Furthermore, the experiment is executed 10 times for each curve. The results of this experiment are listed in Table III and visualized in Figure 4. In the figure, the minimum and maximum time needed to verify the transactions is also shown.

It is clear from the results that the verification of the proofs is the major part of the verification process. Similar to the growth of the digital euro, the time it takes to verify increases linearly with the number of transactions. The elliptic curve used in the protocol significantly impacts the performance of the transaction verification.

With the current implementation, the Type A curves outperform the other two curve types. Combining that with the results from the growth rate (Table II), the curves of Type A seems to be the most favourable option. Even though proofs created with the curves of Type F are more compact, the time it takes to compute the pairings makes them less usable after multiple transactions.

It is important to note that the implementation is not optimized for the best performance of the protocol. Therefore several steps can be taken to reduce the verification time of the transaction. The authors of JPBC mention that pairings in JPBC without preprocessing are roughly 5.5 times slower than the PBC [5] framework it ported to Java. Furthermore, the verification process is now single-threaded but could be parallelized as verifying the proofs themselves is not dependent on the other proofs. Other performance boosts could be preprocessing elements of the CRS or storing digital euros (partly) precomputed. However, more research is needed for this.

VIII. Limitations and Future Work

The current protocol relies on a TTP to revoke the anonymity of users in case double spending is detected. However, the TTP can revoke anyone's identity based on a single transaction. This makes it possible for a malicious TTP to fully trace transactions when it receives a digital euro with the full list of proofs. In most literature, the TTP requires two proofs of the double-spend transaction to revoke the user's anonymity. Even though this protocol offers more privacy and anonymity than the traditional banking system, a 'once concealed twice revealed' approach might be more desirable. Such an approach might be feasible by using a different type of GS-proof. For example, by changing how the targets of the proofs are constructed. If it is possible to create the proofs such that two targets generated for the double-spending transaction would reveal the identity of the double-spender a commitment scheme that always hides the spender's identity can be used.

The ability to revoke the anonymity from one transaction has a legal advantage. When a perpetrator would only spend e-cash obtained through theft or a forced money transfer once, the perpetrator can be identified. Without this possibility, the perpetrator would not be identifiable from a valid transaction.

To further protect users' privacy, the CRS used in the protocol can be constructed by a collaboration of multiple parties. The ability of a single party to revoke the anonymity of all users is then removed. To revoke the anonymity of users all parties are needed.

Another limitation is that users can recognize e-cash, which they had before. The signature and transaction proofs are not randomized with each transaction. Therefore if a user notices that it had the same e-cash before, it is possible to gain some knowledge regarding the traceability of the e-cash. This knowledge allows the user to link the receiver of the earlier transaction to the spender from whom the user received the e-cash and the number of transactions in between. This linkability could be avoided by randomizing both the signature and transaction proofs for every transfer as is done in [19] and [20].

More research is needed to determine which curve type is most optimal. This curve must balance the growth per transaction, the verification performance and the application's

[5] https://crypto.stanford.edu/pbc/

Curve	First transaction (ms)	50th transaction	Average time increase per transaction (ms)
Type A	165	4545	91
Type E	485	14470	292
Type F	1530	62255	1259

TABLE III: Digital euro verification performance ($r = 160$).

security. A more optimized version of the protocol is needed for this. This optimization could be achieved by implementing the pairing in a more efficient framework. Moreover, other improvements could be preprocessing, parallelization and (partial) precomputation.

IX. Conclusion

This thesis proposes an offline transferable e-cash scheme that could be used for the offline digital euro. Furthermore, this thesis introduces an open-source implementation of this scheme. The protocol offers fully offline transactions with transferable tokens and more privacy than current digital payment options or the earlier proposed EuroToken.

Our implementation and tests show that the protocol is correct and that the number of transactions, in which the digital euro is used, affects the size of the digital euro and the transaction verification time in a linear manner. However, more research and optimizations are needed to find the best curve to use in the final implementation of the digital euro.

In conclusion, our implementation of the digital euro will enhance the digital payment ecosystem and solve the two main concerns with the current digital payment systems, namely the dependency on an online infrastructure and the fact that third parties can observe transactions. Our digital euro can make the economic system more durable and stable in areas with low coverage or during power outages, whilst providing more privacy than the current alternatives.

References

[1] DNB. "Use of cash lower in Euro Area Countries". In: *De Nederlandsche Bank* (Dec. 2022). URL: https://www.dnb.nl/en/general-news/dnbulletin-2022/use-of-cash-lower-in-euro-area-countries.

[2] Narayan Bhusal et al. "Power system resilience: Current practices, challenges, and future directions". In: *Ieee Access* 8 (2020), pp. 18064–18086.

[3] Adam X Andresen et al. "Understanding the social impacts of power outages in North America: a systematic review". In: *Environmental Research Letters* 18.5 (2023), p. 053004.

[4] ATD Perera et al. "Quantifying the impacts of climate change and extreme climate events on energy systems". In: *Nature Energy* 5.2 (2020), pp. 150–159.

[5] Satoshi Nakamoto. "Bitcoin whitepaper". In: *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

[6] *Zcash*. Accessed: 2024-06-05. URL: https://z.cash/learn/.

[7] *Ethereum*. Accessed: 2024-03-04. URL: https://ethereum.org/en/.

[8] bitcoin.org. *Protect your privacy*. Accessed: 2024-03-04. URL: https://bitcoin.org/en/protect-your-privacy.

[9] Tin Tironsakkul et al. "Context matters: Methods for Bitcoin tracking". In: *Forensic Science International: Digital Investigation* 42 (2022), p. 301475.

[10] Jannik Dreier, Ali Kassem, and Pascal Lafourcade. "Formal analysis of e-cash protocols". In: *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*. Vol. 04. 2015, pp. 65–75.

[11] Atlantic Council. *Central Bank Digital Currency Tracker*. Accessed: 2024-03-04. URL: https://www.atlanticcouncil.org/cbdctracker/.

[12] John Kiff. *Taking digital currencies offline*. July 2022. URL: https://www.imf.org/en/Publications/fandd/issues/2022/09/kiff-taking-digital-currencies-offline.

[13] Panagiotis chalopoulos et al. "Compliance Design Options for Offline CBDCs: Balancing Privacy and AML/CFT". In: *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2024).

[14] Weijie Liu et al. "Understanding TEE containers, easy to use? Hard to trust". In: *arXiv preprint arXiv:2109.01923* (2021).

[15] Jaehyuk Lee et al. "Hacking in darkness: Return-oriented programming against secure enclaves". In: *26th USENIX Security Symposium (USENIX Security 17)*. 2017, pp. 523–539.

[16] Stefan Brands. "Untraceable off-line cash in wallet with observers". In: *Advances in Cryptology—CRYPTO'93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings 13*. Springer. 1994, pp. 302–318.

[17] Chun-I Fan, Wei-Zhe Sun, Hoi-Tung Hau, et al. "Date attachable offline electronic cash scheme". In: *The Scientific World Journal* 2014 (2014).

[18] Joseph K Liu, Patrick P Tsang, and Duncan S Wong. "Recoverable and untraceable e-cash". In: *Public Key Infrastructure: Second European PKI Workshop: Research and Applications, EuroPKI 2005, Canterbury, UK, June 30-July 1, 2005, Revised Selected Papers 2*. Springer. 2005, pp. 206–214.

[19] Foteini Baldimtsi et al. "Anonymous transferable e-cash". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2015, pp. 101–124.

[20] Balthazar Bauer, Georg Fuchsbauer, and Chen Qian. "Transferable E-cash: A cleaner model and the first

practical instantiation”. In: *IACR International Conference on Public-Key Cryptography*. Springer. 2021, pp. 559–590.

[21] Jianbing Ni et al. “Dual-Anonymous Off-Line Electronic Cash for Mobile Payment”. In: *IEEE Transactions on Mobile Computing* 22.6 (2023), pp. 3303–3317. DOI: 10.1109/TMC.2021.3135301.

[22] European Central Bank. *Where does the project stand?* Oct. 2023. URL: https://www.ecb.europa.eu/paym/digital_euro/timeline/html/index.en.html.

[23] European Central Bank. “A stocktake on the digital euro”. In: *Eurosystem* (Oct. 2023).

[24] Office of Science and Technology Policy. *Technical Evaluation for a U.S. Central Bank Digital Currency System*. 2022.

[25] Stefano Leucci, Massimo Attoresi, and Xabier Lareo. *TechDispatch #1/2023 - Central Bank Digital Currency*. 2023.

[26] David Chaum and Torben Pryds Pedersen. “Transferred cash grows in size”. In: *Workshop on the Theory and Application of Cryptographic Techniques*. Springer. 1992, pp. 390–407.

[27] David Chaum. “Blind signatures for untraceable payments”. In: *Advances in Cryptology: Proceedings of Crypto 82*. Springer. 1983, pp. 199–203.

[28] Wen-Sheng Juang. “A practical anonymous off-line multi-authority payment scheme”. In: *Electronic Commerce Research and Applications* 4.3 (2005), pp. 240–249.

[29] Wen-Sheng Juang. “RO-cash: An efficient and practical recoverable pre-paid offline e-cash scheme using bilinear pairings”. In: *Journal of Systems and Software* 83.4 (2010), pp. 638–645.

[30] Eligijus Sakalauskas et al. “A simple off-line E-cash system with observers”. In: *Information Technology and Control* 47.1 (2018), pp. 107–117.

[31] Jia-Ning Luo and Ming-Hour Yang. “Offline transferable E-cash mechanism”. In: *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. Ieee. 2018, pp. 1–2.

[32] Zhexuan Hong and Jiageng Chen. “A Solution for the Offline Double-Spending Issue of Digital Currencies”. In: *International Conference on Science of Cyber Security*. Springer. 2022, pp. 455–471.

[33] Lynn Batten and Xun Yi. “Off-line digital cash schemes providing untraceability, anonymity and change”. In: *Electronic Commerce Research* 19 (2019), pp. 81–110.

[34] Ivan Osipkov et al. “Combating double-spending using cooperative P2P systems”. In: *27th international conference on distributed computing systems (ICDCS’07)*. IEEE. 2007, pp. 41–41.

[35] Ziba Eslami and Mehdi Talebi. “A new untraceable off-line electronic cash system”. In: *Electronic Commerce Research and Applications* 10.1 (2011), pp. 59–66.

[36] Yaser Baseri, Benyamin Takhtaei, and Javad Mohajeri. “Secure untraceable off-line electronic cash system”. In: *Scientia Iranica* 20.3 (2013), pp. 637–646.

[37] Wessel Blokzijl. “EuroToken: An offline capable Central Bank Digital Currency”. In: (2021).

[38] Robbert Koning. “Performance analysis of an offline digital Euro prototype”. In: (2023).

[39] Pratik Sarkar. “Multiple-use transferable e-cash”. In: *Cryptology ePrint Archive* (2013).

[40] João Marcos de Mattos Barguil and Paulo Sérgio Licciardi Messeder Barreto. “Efficient methods for lattice-based cryptography”. In: (2015).

[41] ECB. *The role of cash*. URL: https://www.ecb.europa.eu/paym/digital_euro/timeline/html/index.en.html.

[42] European Central Bank. “The case for a digital euro: key objectives and design considerations”. In: *Eurosystem* (July 2022).

[43] Jens Groth and Amit Sahai. “Efficient non-interactive proof systems for bilinear groups”. In: *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*. Springer. 2008, pp. 415–432.

[44] Sanjit Chatterjee, Darrel Hankerson, and Alfred Menezes. “On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings”. In: *International workshop on the arithmetic of finite fields*. Springer. 2010, pp. 114–134.

[45] Angelo De Caro and Vincenzo Iovino. “jPBC: Java pairing based cryptography”. In: *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1: IEEE, 2011, pp. 850–855. URL: http://gas.dia.unisa.it/projects/jpbc/.

[46] Angelo De Caro and Vincenzo Iovino. “jPBC: Java Pairing Based Cryptography”. In: June 2011, pp. 850–855. DOI: 10.1109/ISCC.2011.5983948.

[47] Joseph A. Akinyele, Matthew D. Green, and Avi D. Rubin. *Charm: A framework for Rapidly Prototyping Cryptosystems*. Cryptology ePrint Archive, Paper 2011/617. https://eprint.iacr.org/2011/617. 2011. URL: https://eprint.iacr.org/2011/617.

[48] Paulo SLM Barreto and Michael Naehrig. “Pairing-friendly elliptic curves of prime order”. In: *International workshop on selected areas in cryptography*. Springer. 2005, pp. 319–331.

[49] Diego F Aranha, Youssef El Housni, and Aurore Guillevic. “A survey of elliptic curves for proof systems”. In: *Designs, Codes and Cryptography* 91.11 (2023), pp. 3333–3378.

[50] Diptendu M Kar and Indrajit Ray. “Systematization of knowledge and implementation: Short identity-based signatures”. In: *arXiv preprint arXiv:1908.05366* (2019).

[51] Nicolas T Courtois, Pinar Emirdag, and Daniel A Nagy. “Could bitcoin transactions be 100x faster?” In: *2014 11th International Conference on Security and Cryptography (SECRYPT)*. IEEE. 2014, pp. 1–6.