Offline Euro POC
— Student Project —

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

Abstract— **Current digital payment solutions are extremely fragile when compared to traditional cash. Their critical dependency on an online service used to perform and validate transactions is a weakness by design. The entire digital payment solution is void and useless if this service is unreachable. This leads to financial exclusion for users in areas with unreliable network coverage. Moreover, no transaction can be executed during server malfunctions or power outages. The latter scenario will occur more frequently as climate change increases the likelihood of extreme weather, leading to more power outages. Another problem with today's digital payment options offered by banks is that they offer little to no privacy. This is an inherent result of their account-based design. People desire more privacy regarding their financial decisions as seen by the rise in popularity and adoption of cryptocurrencies. This weakens the influence that central banks have on economic policies. The critical dependency and lack of privacy can be resolved with a Central Bank Digital Currency that can be used offline, similar to cash. This thesis proposes a design and a first implementation for an offline-first digital euro. The protocol uses transferable tokens and offers complete privacy during transactions using zero-knowledge proofs. Furthermore, transactions can be executed offline without an active third party 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. Possible extensions to the protocol would be to use EBSI services at endpoints to identify users with a passport-grade level of identification for more secure Know-Your-Customer principles.**

I. Introduction

For the past decade, the share of digital payments has increased and the number of cash payments has declined [1]. However, the dependency on a connection to an online infrastructure during the transaction has increased. 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 goods. Additionally, the money must be transferred from the payer's account to the one of the payee.

Other digital payment options, such as most cryptocurrencies, have the same dependency on being connected. In the case of Bitcoin [2], a connection to the ledger is needed to verify whether the transaction is included in the global blockchain.

The result of this dependency on online infrastructure is that the payment options are unusable whenever the servers are unreachable. This could for example be in regions with no internet coverage, when the servers of a bank are down or during a power outage.

The number of outages has increased for the past years [3] and it is expected that the likelihood of power outages will increase in the future [4, 5]. A significant share of these outages are caused by extreme weather events, such as heatwaves, blizzards, hurricanes and floods [6, 7, 8, 9].

Due to climate change, the likelihood and extremity of these weather events have increased [10, 11, 12], which could cause more frequent outages. To have a digital payment option available during those conditions, the transaction must be possible in an offline manner. This implies that no other party but the payer and payee can be involved during the transaction.

Another issue with the current digital payment methods is that they are not privacy-protecting. The bank has a complete list of all transactions involving the account holders and their balances. In case of a breach, this data could be abused.

For most cryptocurrencies, transactions are stored in a public ledger, using a wallet address as a pseudonym. Some of those cryptocurrencies, like Ethereum [13], users have a fixed wallet address. If you know which address belongs to someone, the transactions executed with that wallet can be traced. For other cryptocurrencies like Bitcoin [14] it is feasible to change the wallet addresses with every transaction. However, an address becomes tainted with each transaction and can be tracked with a taint analysis [15].

Another digital payment option that could be used offline and with more privacy is electronic cash (e-cash). Depending on the protocol, e-cash has similar properties to physical cash. Comparable to regular cash, a user must first withdraw money from the bank. In e-cash, this money is a digital token and can be stored on a device. At a later stage, the holder can spend the token(s) by transferring the tokens to the receiver. Finally, the receiver can deposit the tokens at the bank to redeem the value of the tokens.

In an offline e-cash scheme, no bank, ledger, or other third party is involved in the transaction between the spender and the receiver. Therefore, the transaction can be executed in an offline manner.

Many central banks have expressed their interest in e-cash. Some central banks are already providing digital versions of their currencies as e-cash. 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 [16]. Examples of CBDCs in circulation are: *e-Naira* (Nigeria), *Sand Dollar* (The Bahamas) and *JAM-DEX* (Jamaica). Several CBDCs of countries in the G20 that are currently in the pilot phase are: *Digital Yen* (Japan), *e-CNY* (China) and *eAUD* (Australia).

However, a survey from the International Monetary Fund (IMF) [17] found that most CBDCs in development can only be used online. The ones that can be used offline typically rely on tamper-resistant hardware to maintain the integrity of the CBDCs stored on a device. As Liu et al. [18] and Lee et al. [19] have shown, even the current state-of-the-art tamper-resistant, secure hardware can be breached. Therefore, the design of the CBDC must rely on established cryptographic protocols to maintain the system's integrity, rather than 'tamper-resistant' hardware.

Currently, the European Central Bank (ECB) is in the preparation stage of designing the digital euro [20]. Two of the main design goals of the digital euro are protecting privacy as much as possible and support for offline transactions [21]. Michalopoulos et al. [22] state that an offline CBDC also promotes financial inclusion, lower transaction costs and an improved user experience, which could generate trust.

This thesis proposes a design for the digital euro, fulfilling these goals. The protocol relies on zero-knowledge proofs to protect the privacy of users. By using zero-knowledge proofs, other users and banks can verify a transaction without being able to identify the participants. Furthermore, participants in the system operate under digital pseudonyms to protect their privacy. This approach is being explored by the Office of Science and Technology Policy [23] for a digital U.S. Dollar and recommended by the European Data Protection Supervisor [24].

The anonymity of users could be further protected by integrating the European Blockchain Services Infrastructure (EBSI) wallet to handle digital identity. This would allow users to act under a passport-grade pseudonym while banks could still apply the Know-Your-Customer (KYC) principles.

II. Problem description

Balancing privacy and fraud prevention, double spending and transferability are three major problems regarding e-cash. Fraud can be trivially detected and prevented by making the e-cash scheme linkable and traceable. This would, however, require all participants in the system to give up their privacy and reveal sensitive information regarding their spending behaviour. Therefore offline e-cash transactions should provide anonymity and be untraceable.

Unfortunately, they are more prone to malicious actions since no central party can be reached to verify transactions. Malicious users can not be identified or even detected if the protocol offers too much anonymity. One example of such action is double-spending. Receivers of an e-cash token cannot check if the same token is spent in an earlier transaction. Thus in a fully anonymous setting, malicious users could freely duplicate e-cash and spend the tokens at different places. In the literature, there are two ways to mitigate double-spending.

Several e-cash schemes, such as [25, 26, 27, 28, 29], prevent double-spending utilizing secure and tamperproof hardware or software. Those implementations rely on the hardware or software to remove or mark a token used after the transaction. However as Liu et al. [18] and Lee et al. [19] have proven, such hardware and software are not fully secure and tamperproof and can thus be breached. This allows malicious users to freely double-spend their e-cash.

The other solution relies on cryptographic principles to detect double spending and revoke the anonymity of the malicious user. This often occurs when the e-cash is deposited, such as in [30, 31, 32, 33, 34, 35].

In most offline e-cash schemes the tokens are not transferable. This means that a token can only be used for one transaction. After that transaction, the receiver must deposit at the bank and cannot use it for another transaction. This implies that during a longer period in which the bank cannot be reached the number of transactions is limited by the number of e-cash circulating.

On the other hand, transferable e-cash can be used in multiple transactions like physical cash. Whenever someone receives e-cash it can be reused for the next transaction. This reduces the dependency on the infrastructure of the bank. Furthermore, it allows for a more efficient implementation of e-cash with multiple denominations since users can use the change they receive in future transactions. However, the downside of transferable e-cash is that every transaction must be included with the e-cash to detect double-spending. This implies that the size of the e-cash grows with every transaction [36]. This also makes hiding the identity of spenders more complex.

Some e-cash schemes [33, 34] are based on a combination of several difficult cryptographic principles, making them efficient and powerful, but also very complex and hard to understand. As simplicity can be a key factor in generating trust in a system [37], the protocol of the digital euro must be transparent and understandable. This trust in the system could play a vital role in the adaptation of CBDCs by the masses.

III. Related work

Since the introduction of blind signatures in 1983 by Chaum [38] and the first offline e-cash protocol by Brands [30], 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 is trivial, however, breaking this integrity would also invalidate the entire e-cash scheme.

A subset of the research that relies on cryptography is unpractical or even nonfunctional in a real-world scenario, especially when intended to be used as a basis of a CBDB. Moreover, other research regarding offline e-cash introduces functionality (token expiration), which does not solve and potentially worsens the problem it was intended to solve.

To fully benefit from the offline functionality, token transferability is highly desirable. This would reduce the dependency on reaching the bank and lower the number of communication actions. However, not all protocols proposed in the literature support this.

A prototype of an offline digital euro, EuroToken, was proposed earlier. However, this proposed scheme is fully traceable and offers little privacy to the users and thus conflicts with some of the design goals set by the European Central Bank.

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. [39] 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, is unlikely to occur and does not offer a solution to pay in areas with no network coverage. Another example is the scheme proposed by Batten et al. [40], which provides change by giving reputable shops, such as Target, the authority to mint cash.

B. *Expiring e-cash*

Eslami and Talebi [41] introduced expiring e-cash by attaching an expiration date to the e-cash description. This scheme was later improved by [42] and [31]. The main reason for this expiration date is a storage reduction for the bank.

However, the question remains if these schemes 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. Therefore, the number of deposits does not change. This means that the size of the deposit table will not be affected 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. 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.

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 [36].

Sarkar [43] tried to achieve this property using bitwise XORs. However, the protocol uses an unspecified distributive operator over XOR to detect double-spending [44]. Furthermore, Barguil [44] also proves that the security claims made by Sarkar do not hold.

Baldimtsi et al. [33] 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 tags.

This protocol is improved by Bauer et al. [34] 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.

Jianbing et al. [35] tried to take the transferability one step further by proposing a transferable e-scheme that allows the receiver to be anonymous and thus provides dual anonymity. However, they used a much less useable definition of transferable, as the protocol requires users to contact the bank to re-randomize a received token after each transaction.

D. *Eurotoken*

Blokzijl [45] and Koning [46] of the Tribler Lab [1] and the Delft University of Technology did earlier work regarding a CBDC, named Eurotoken, that the EU could use. 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.

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

[1] https://www.tribler.org/

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. Moreover, all transactions are visible to the bank. This makes it possible for the bank to construct a graph which can be used to trace the payment system.

Privacy is an important factor in why people use cash for payments [47]. 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 bonus of paying offline will cost you your privacy. Moreover, the provided protocol does not align with the main design goal of the ECB, namely privacy protection [48].

IV. Security assumptions

The protocol proposed by this thesis relies mainly on two security assumptions to guarantee unforgeability and anonymity. These assumptions are the *Discrete logarithm problem* and the *Computational Diffie-Hellman* assumption.

A. Discrete logarithm problem

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.

B. Computational Diffie-Hellman

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

The 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 [38] 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 more anonymity. In this thesis, an implementation of a hash-based blind Schnorr signature (BSS) is used. However, any blind signature protocol could be 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 and the signing party. Furthermore, the signing party chooses a random private key $x \in_R \mathbb{Z}_q^*$ and publishes public key $y = g^x$. 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: $c \overset{?}{=} H(r_v||M)$.

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

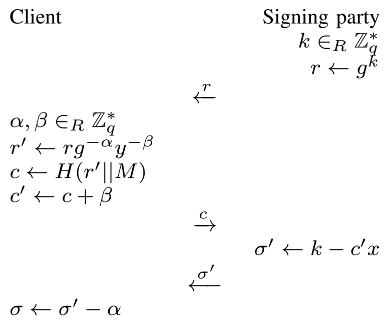
Client
Signing party
$k \in_R \mathbb{Z}_q^*$
$r \leftarrow g^k$
$\xleftarrow{r}$
$\alpha, \beta \in_R \mathbb{Z}_q^*$
$r' \leftarrow rg^{-\alpha}y^{-\beta}$
$c \leftarrow H(r'||M)$
$c' \leftarrow c + \beta$
$\xrightarrow{c}$
$\sigma' \leftarrow k - c'x$
$\xleftarrow{\sigma'}$
$\sigma \leftarrow \sigma'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

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:

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

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:

  $$\begin{aligned} e(P+Q, R) &= e(P,R) \cdot e(Q,R) \\ e(P, R+S) &= e(P,R) \cdot e(P,S) \end{aligned}$$

  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 $h_1, h_2 \in H$:

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

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 [49] 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 used to be very efficient 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 $G_1 \neq G_2$ the pairing is asymmetric.

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 structures, this will reveal the input. However, in other structures, no secret information can be found. 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.

A full proof can be verified with an equation similar to Equation 2:

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

More specifically, 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:

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

In this thesis, 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:

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

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

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

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. 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.

$$X = c_1^{-\alpha} c_2 \tag{3a}$$

$$Y = d_1^{-\beta} d_2 \tag{3b}$$

VI. System architecture overview

The protocol is divided into four phases: Initialization, withdrawal, transactions and deposit. The initialization phase is executed only once by the trusted third party (TTP) 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 EBSI identification service can be used to prove the user's identity. 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. 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 the 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 [36]. 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. Furthermore, t is used to prove knowledge of the randomization elements used in the previous transaction, as the t, will be used to determine randomization in the next transaction. The spender will use these randomization elements when creating the GS proof for the transaction.

The target of the proof, T, depends on whether the digital euro is spent earlier. When the euro has not been spent before, the target is $T = e(g_1, g_2)^\sigma$. 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 $y = \frac{T}{x}$ 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, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy} = e(g_1, g_2)^T$.

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 can be computed as:

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

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 implementation can be found on GitHub [2]. The Java Pairing Based Cryptography (JPBC) library [50] 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. The prototype is built as a mobile application to mimic the current payment options. 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.

A. 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 [51]. 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.

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

Curve	Initial size (kB)	50 transactions (kB)	Grotwh (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 I: Digital euro growth with transactions ($r = 160$).

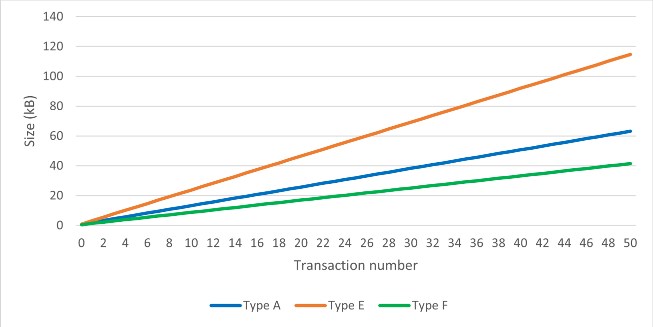

Fig. 2: Digital euro growth per transaction ($r = 160$).

B. 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. Pairings of type A and E are symmetric and the ones of type F are asymmetric. The difference between A and E is the size of the fields. During the test, the prime order used (r) remained constant ($r = 160$).

Table I 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 different element sizes of groups G_1 and G_2. Due to the size of the fields, 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. The first transaction, however, has a slightly larger growth (+250 bytes) due to the initialization of the list structure for the transaction proofs. From Equation 5 the growth of the digital euro was expected to grow linear, depending on the size of the group elements. This is visualised in Figure 2.

C. Transaction verification performance

The time it takes to verify a transaction is a major factor in adopting digital currencies and their useability 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 near instantly [52], this payment option is unusable in most scenarios.

To test the transaction verification performance of the digital euro, a test is done that measures the time it takes to deposit a digital euro used in 50 transactions. To verify this transaction the bank must check three Schnorr signatures, the current transaction proof and the chain of 50 previous proofs. The results of this experiment are listed in Table II.

Curve	Single proof (ms)	50 proof chain (ms)	Deposit protocol (ms)
Type A	68	4334	4626
Type E	259	17084	18046
Type F	1073	68617	69934

TABLE II: Digital euro verification of proofs and deposit protocol ($r = 160$).

It is clear from the results that the verification of the proofs is the major part of the verification process. During the deposit protocol, verifying the proofs took 97% of the time on average. The elliptic curve used in the protocol significantly impacts the performance of the transaction verifications.

With the current implementation, the type A curves outperform the other two curve types. Combining that with the results from the growth rate (Table I), the curves of type A seem 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 unusable in multiple transactions for now.

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 [3] 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.

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

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

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 of the protocol 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 [33] and [34].

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 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 transferrable e-cash scheme that could be used as a prototype for the CBDC of the ECB. The protocol is based on bilinear pairings through GS-proofs. Using these proofs, the identity of the users is encrypted into the commit values of the proof. However other users can only verify that the transactions are valid and cannot obtain information from the proofs. As every transaction with the same digital euro is linked to the previous one in two ways, malicious users cannot alter the proof history. Additionally, as the users must know a secret variable used in the previous proof to generate a new valid proof, users cannot spend digital euros which they did not directly receive.

The scheme relies on a TTP to handle the users' identities and to revoke their anonymity when needed. Whenever the bank detects double spending when receiving two tokens with the same serial number and signature, the TTP extracts the identity from the proofs. Even though the TTP only needs one proof to revoke the anonymity of users, the protocol gives more privacy towards the bank than the traditional banking systems and Eurotoken.

Another problem is that users can recognize digital euros they have had before. However, they can extract little information from this recognition. This problem could potentially be mitigated by randomizing the proofs with each transaction. However, this will increase the cryptographic complexity of the system, which could hurt the adoption rate.

The protocol also has a public proof of concept implementation. This implementation can be seen as a real-world example of how the system could be used. Additionally, the proof of concept also makes it easier to reason about bottlenecks and other potential problems in the system.

Initial tests show that the digital euro grows linearly with the size of the group elements. Using different types of curves results in different growth sizes. However, these changes also affected the performance of the verification process. Further work and research are needed to optimize this process and find the optimal curve.

In conclusion, this protocol is an initial implementation for the digital euro, useable by the European Central Bank. The protocol offers a transferrable offline e-cash scheme and more privacy than current digital payment options or the earlier proposed Eurotoken. Therefore, this prototype of the digital euro will enhance the digital payment ecosystem and 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] Satoshi Nakamoto. "Bitcoin whitepaper". In: *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

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

[4] 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.

[5] 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.

[6] Laiz Souto et al. "Identification of weather patterns and transitions likely to cause power outages in the United Kingdom". In: *Communications Earth & Environment* 5.1 (2024), p. 49.

[7] J Schaller and S Ekisheva. "Leading causes of outages for transmission elements of the North American bulk power system". In: *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE. 2016, pp. 1–5.

[8] Stephen A Shield et al. "Major impacts of weather events on the electrical power delivery system in the United States". In: *Energy* 218 (2021), p. 119434.

[9] Joan A Casey et al. "Power outages and community health: a narrative review". In: *Current environmental health reports* 7 (2020), pp. 371–383.

[10] Peter Stott. “How climate change affects extreme weather events”. In: *Science* 352.6293 (2016), pp. 1517–1518.

[11] Kristie L Ebi et al. “Extreme weather and climate change: population health and health system implications”. In: *Annual review of public health* 42.1 (2021), pp. 293–315.

[12] Intergovernmental Panel on Climate Change (IPCC). “Weather and Climate Extreme Events in a Changing Climate”. In: *Climate Change 2021 – The Physical Science Basis: Working Group I Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2023, pp. 1513–1766.

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

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

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

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

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

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

[19] 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.

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

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

[22] 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).

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

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

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

[26] Wen-Shenq 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.

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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`.

[36] 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.

[37] Johannes Strikwerda. “Simplicity and Complexity”. In: *Organized Complexity in Business: Understanding, Concepts and Tools*. Springer, 2023, pp. 31–49.

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

[39] 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.

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

[41] 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.

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

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

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

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

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

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

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

[49] 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.

[50] 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/`.

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

[52] 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.