PEReDi: Privacy-Enhanced, Regulated and Distributed Central Bank Digital Currencies

Citation for published version:

Digital Object Identifier (DOI):
10.1145/3548606.3560707

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
ABSTRACT
Central Bank Digital Currencies (CBDCs) aspire to offer a digital replacement for physical cash and as such need to tackle two fundamental requirements that are in conflict. On the one hand, it is desired they are private so that a financial "panopticon" is avoided, while on the other, they should be regulation friendly in the sense of facilitating any threshold-limiting, tracing, and counterparty auditing functionality that is necessary to comply with regulations such as Know Your Customer (KYC), Anti Money Laundering (AML) and Combating Financing of Terrorism (CFT) as well as financial stability considerations. In this work, we put forth a new model for CBDCs and an efficient construction that, for the first time, fully addresses these issues simultaneously. Moreover, recognizing the importance of avoiding a single point of failure, our construction is distributed so that all its properties can withstand a suitably bounded minority of participating entities getting corrupted by an adversary. Achieving all the above properties efficiently is technically involved; among others, our construction uses suitable cryptographic tools to thwart man-in-the-middle attacks, it showcases a novel traceability mechanism with significant performance gains compared to previously known techniques and, perhaps surprisingly, shows how to obviate Byzantine agreement or broadcast from the optimistic execution path of a payment, something that results in an essentially optimal communication pattern and communication overhead when the sender and receiver are honest. Going beyond "simple" payments, we also discuss how our scheme can facilitate one-off large transfers complying with Know Your Transaction (KYT) disclosure requirements. Our CBDC concept is expressed and realized in the Universal Composition (UC) framework providing in this way a modular and secure way to embed it within a larger financial ecosystem.

CCS CONCEPTS
• Security and privacy → Distributed systems security; Public key (asymmetric) techniques; Formal security models; • Applied computing → Electronic commerce.

KEYWORDS
Privacy; Cryptography; CBDC; Distributed Ledgers; Regulatory Compliance; KYC; AML; CFT; KYT; Universal Composition.

1 INTRODUCTION
The development of cryptocurrencies provided a strong motivation for the development of "central bank digital currency" (CBDC) systems. A CBDC is central bank money but more widely accessible and transferable than central bank reserves and banknotes (see e.g., Bank of England [27] for an overview of the basic principles of such systems). This type of money can also be interest bearing (with a different rate than that on reserves) [12] and has a different operational structure than other forms of central bank money [37].
It was early on observed that CBDCs solve a different problem than general cryptocurrencies such as Bitcoin and/or Ethereum. The first construction that exploited this distinction is RSCoin [22] which was followed by designs explored by a number of central banks [1, 11, 20]. In such systems the verification of transactions relies on a distributed set of independent authorities (we call them "maintainers"). Such entities are empowered to enforce the monetary and regulatory policies of the system that are dictated by the central bank and regulatory entities. A distinguishing characteristic of CBDC systems compared to cryptocurrencies is that the monetary policy is decoupled from the monetary exchange system. The integrity and soundness of the former remains in the purview of the central bank, while the integrity of the latter is distributed across a set of entities. Therefore, the CBDC system’s state is maintained in a distributed manner by the maintainers such that the central bank as well as any regulatory entities can be offline during the time users transact.
A common concern expressed in the context of CBDCs is that, contrary to other forms of central bank money, a CBDC may transform the central bank into a “panopticon” that is continuously aware of all transactional data. Such concerns have also been highlighted in the context of cryptocurrencies. First generation cryptocurrencies such as Bitcoin and Ethereum are only pseudonymous in the sense that a user’s transactions are linkable to a (set of) pseudonym(s) that the user can generate. Privacy enhanced cryptocurrencies (e.g., ZCash [9] or Monero [31]) were developed to hide the value of transactions and offer unlinkable transactions to a certain degree or under plausible assumptions. Note that such systems enjoy a level of anonymity that does not reveal directly any information about payment counterparties and transaction values and, hence, may be attractive and be used for illegal activities such as money laundering, financing terrorism, and so on. As a result, privacy-preserving systems using such techniques can be problematic in settings where comprehensive regulatory compliance is required. CBDCs constitute such setting and hence it is imperative to have built-in features by which, while full anonymity can be offered for most circumstances, at the same time conditional disclosure to regulators and law enforcement in case of misbehavior can be facilitated, cf. [4].

Privacy in payment systems can interfere with three main regulatory obligations: (1) Know-Your-Customer (KYC), which requires the positive identification of counterparties before they are able to transact. (2) Anti-Money Laundering (AML), which requires that sources of funds should be legitimate. (3) Combating Financing of Terrorism (CFT), which requires that the recipients of funds should not engage in terrorism. To appreciate the way such requirements interfere with privacy, it helps to imagine the set of all payments as a hidden directed graph where vertices correspond to counterparties and edges to payments between them weighted by their value. Using this abstraction, it follows that introducing vertices in the graph should be subject to KYC, while it should be possible to reveal the incoming or outgoing edges to any vertex which is suspected for illicit or terrorism activity, as well as trace selectively particular paths in the graph from source to destination and vice-versa to address AML and CFT considerations. Beyond these opening and tracing operations it is widely recognized in the CBDC context, cf. [1, 7, 11], that it is desirable to restrict both the volume of payments that a particular vertex can make (so that “hoarding” CBDC currency is tempered) as well as limit the amount of value that can be transferred between two counterparties in a single transaction, without triggering additional auditing regarding the funds of the sender (what is referred to as KYT - know your transaction, cf. [3]). Unfortunately, currently no existing CBDC design offers privacy combined with such “regulation friendly” capabilities.

Our Results. We put forth a model and construction that for the first time addresses all the issues identified above simultaneously. In PEReDi each user has an account which is approved during on-boarding (i.e., it undergoes KYC) and can subsequently be issued currency by the central bank (following its monetary policy) as well as receive or transmit funds to other users. Our design approach applies a novel combination of cryptographic primitives and distributed organization that, perhaps surprisingly, shows how we can remove the requirement for (byzantine) agreement or broadcast from the optimistic path of payment execution. PEReDi features an encrypted ledger maintained separately by each maintainer, transactions are identified by transaction identifiers and leave encrypted fingerprints in the ledger of each maintainer that under normal circumstances are completely opaque. Transaction senders and receivers independently update their private accounts, leaving the above traces, while only in the case of a transaction abort the maintainers need to engage in an agreement protocol to ensure consistency. In this way, PEReDi offers a digital equivalent of physical cash: payments do take place with double-spending prevention without anyone in the system becoming aware of the precise value transferred or the counterparties involved. Moreover, both sender and receiver need to engage for the payment, something that prevents “dusting” attacks\(^1\). At the same time (and contrary to physical cash) the transaction value is subject to constraints in terms of sending and receiving limits of the two counterparties and maximum transaction size, while the counterparties themselves are preconditioned to proper KYC onboarding. Tracing and opening operations are accommodated by the design elements of the encrypted ledgers.

Given adequate evidence about suspicious activities of a specific user or a particular transaction (indexed by its unique transaction identifier), the authorities can trace transactions made by that user or reveal the metadata of a given transaction by unlocking the real world identities of the counterparties or the total value transferred. Combining these opening and tracing operations, authorities can identify the labels of specific vertices in the payment graph as well as trace paths of payment from source to destination and vice-versa. We stress that such operations require a quorum of entities to agree and hence cannot be unilaterally invoked by any individual entity hence precluding a single point of failure.

To summarize, our contributions are as follows:

1. To the best of our knowledge, this is the first time that a fully privacy-preserving and comprehensively regulated CBDC is modeled formally. Our formal model is in the Universal Composition (UC) setting [16]. This modeling enables the composition of the system as payment infrastructure within larger systems.
2. We review the regulatory compliance in the context of payment systems (KYC, AML, CFT, auditing, etc.) and argue how our ideal functionality for CBDCs captures such requirements.
3. We put forth a distributed construction that realizes our CBDC ideal functionality in an efficient manner based on standard cryptographic assumptions. Notably our construction demonstrates that neither Byzantine broadcast nor agreement is needed in the optimistic execution path of a payment instance, resulting in an optimal communication pattern and message size in the case when both sender and receiver are online and willing to finalize a payment.
4. We introduce a novel simulatable approach for tracing suspicious users in the auditing protocol which is employed for double-spending prevention as well and may be of independent interest as it is more efficient than previously known techniques in the broader context of tracing users in conditionally anonymous payment systems. Moreover, the introduced auditing mechanism does not require Byzantine agreement or broadcast.

---

\(^1\)Dusting attacks were observed in 2022 after the ruling of OFAC to blacklist the anonymization service Tornado Cash [32].
5. We describe how our efficient CBDC construction can facilitate additional features such as protocol support for concurrent digital currency issuance by the central bank for different users, aborting transactions, and Know Your Transaction (KYT) operations. It is worth noting that even though we describe our results in the context of CBDCs, it is immediate that our system can be used to implement any “stablecoin” or more generally fungible digital token which has a centrally managed supply. In such case, the role of the central bank is played by the issuer of the digital token, who is capable to introduce new tokens increasing the supply as determined by the issuer’s policy. It is also straightforward to return such tokens to the issuer by sending them to a designated account for that purpose.

Related Work. The first system for anonymous electronic cash was introduced by Chaum [19] and focused on sender anonymity, while disclosing the recipient’s identity and the amount transferred. The system also required users to hold information linear in the number of coins that they possess, a performance consideration that was addressed in follow up work [13, 18]. Regarding the problem of revealing the transaction value to the bank, transferrable e-cash [6, 15] introduced a mechanism for double-spending prevention. In this mechanism, coins can be transferred to various users without communicating with the bank. Hence, coins expand in size depending on how frequently they are used, which might be inefficient for retail payments. Additionally, in these schemes coins are distinguishable based on the number of transfers performed. Camenisch et al. [14] proposed a token-based e-payment solution in which the bank can enforce simple rules such as per-user payment limits. Privacy of senders of transactions is preserved, nonetheless, the recipient identity and payment amount are leaked.

Considering blockchain-ledger-based anonymous payment systems like Zerocash [9], Garman et al. [24] addressed how regulation rules could be enforced in such constructions. The disadvantage of privacy systems similar to the Zerocash approach is that they result in privacy-preserving transactions that are unsuitable for resource-constrained users. Users should prove knowledge of the path of a transaction output in a Merkle tree, hence, they must maintain an up-to-date version of this tree. Moreover, users are supposed to download the whole ledger and decrypt all transactions to conclude whether they are recipients of transactions. Instead, in our construction there is no need to download the ledger. The necessity for users to be up-to-date with the whole ledger makes distributed blockchain-ledger based constructions less efficient than our scheme which is based on signatures of distributed (known) maintainers on the updated account of each user (this technique eliminates the need to synchronize with the ledger state, which is only necessary for auditing).

Danezis and Meiklejohn, [22] introduced RSCoin, a central bank currency framework which is built around an efficient broadcast mechanism. In RSCoin, the central bank delegates the responsibility of verifying transactions to a set of entities called mintettes. Different from traditional cryptocurrency miners, in their framework mintettes are known and may eventually be held responsible for any misconduct. RSCoin focuses on the scalability of broadcast rather than privacy or regulatory compliance. Performance was improved further with the Fastpay design [8], even though privacy remained unaddressed. Wüst et al. [36] proposed an anonymous payment scheme called PRCash in which transactions are verified in a distributed manner. It achieves privacy and some degree of regulatory compliance. However, the main drawbacks of PRCash are that it does not meet full anonymity as validators can link different transactions and it does not have auditability. Hence, the authorities cannot investigate suspicious transactions or counterparties on demand.

Androulaki et al. [5] introduced a privacy-preserving auditable token management system. Their proposed scheme uses a UTxO model in a permissioned blockchain. In contrast to our construction which is account-based, they target business-to-business scenarios, and they do not offer a comprehensive approach to regulatory compliance as we do. Damgård et al.’s work [21] addressed the problem of balancing accountability with privacy. Nevertheless, their work is in the identity layer for blockchain systems, and they do not study various features necessary for a CBDC system (e.g., currency issuance, transactions between users, financial and regulatory policies, and so on) in their transaction layer framework. The tracing mechanism in [21], for each account generation, requires the account holder to compute a pseudorandom-function PRF using its secret key. There is no concrete implementation for tracing in their work as they use a secure multi-party computation for PRF in a black-box manner. More importantly, the input of PRF is only restricted to be in a range of values making tracing inherently inefficient as authorities are supposed to generate the PRF values for all possible inputs in the range. In contrast, we achieve tracing complexity, per user, proportional to the actual number of transactions issued by that specific user.

Wüst et al. [35] introduced Platypus which is a privacy preserving and centralized payment system. Platypus relies on a single authority, our scheme is distributed such that it is robust against single points of failure with respect to regulation enforcement, and can work even if the central bank is completely offline. Furthermore, our scheme offers encrypted (distributed) ledgers which allow compliance with regulation like AML and CFT, by enabling the set of authorities to trace a malicious user and to discover the transfer value and identities of the counterparties in any suspicious transaction. Platypus [35] does not offer such capability. We stress that it is quite delicate to add efficient tracing and opening mechanisms to a CBDC design as various attacks such as man-in-the-middle attacks where the sender’s transaction information is not tied to the receiver’s identity and vice versa can take place and should be addressed by careful design and modeling choices as we do here. Moreover, the security properties of a CBDC system in their work are defined via a game-based approach something which may limit the composability of their construction, cf. [17]. Finally, another drawback of Platypus [35] is that the technical details on their regulation approach, currency issuance by the central bank, and addressing concurrent and aborted transactions are not formally studied within their security model.

Tomescu et al. [34] introduced a decentralized payment system called UTT. Their construction rely on Byzantine fault tolerant infrastructure. However, PEReDi obviates Byzantine agreement and byzantine broadcast from the optimistic execution path of a transaction. Hence, we have an essentially optimal communication pattern and communication overhead when transaction participants are
The receiver of a transaction has to scan all transactions on a ledger similar to blockchain-ledger-based anonymous payment systems to be able to successfully receive the currency which increases the load on users’ sides. Regarding regulation enforcement, the amount of money that can be anonymously sent in UTT setting is limited by a monthly budget. PEReDi, on the other hand, allows for comprehensive regulatory compliance, and can also enforce them from the recipient’s standpoint.

2. CBDC DESIDERATA AND MODELING
We abstract a CBDC system to three separate classes of entities: the central bank, a set of maintainers (e.g., commercial banks and financial institutions), and users. Role separation is an important element in CBDC design, cf. [1]. The description of these roles together with the relevant assumptions made about them are as follows.

1. Central Bank: The central bank issues the digital currency and is responsible for monetary policy. The monetary supply at any given time is in the purview of the central bank. However, the state of all users’ accounts is not under its control. Moreover, due to the potential threat of mass surveillance [20], the central bank is also not trusted for privacy, i.e., it has no ability to deanonymize the sender or recipient of a transaction or reveal the transferred values associated with a specific transaction. Finally, the central bank is not responsible for enforcing the regulatory rules that govern payments. We refer to [11], and [20] for more context on the role of central banks.

2. Maintainers: The authority of validating transactions and facilitating various auditing operations needed for regulatory compliance is delegated to a number of approved institutions that we call the maintainers. As a result, the central bank and regulator are not needed to be active in any of the system’s day to day operations (except for issuing currency for the former). The maintainers share the state of system and are responsible for continuously updating it as users issue transactions. In a real world deployment, maintainers can be organizations with an existing connection to the central bank for instance, commercial banks, financial institutions, and etc. Note that contrary to e.g., miners in a cryptocurrency blockchain, the set of all maintainers is public and known to all network participants. The basic properties of the system such as the integrity, regulatory compliance and privacy of transactions emanate from the actions of the maintainers. We note that the system’s security and liveness objectives will be met as long as the adversary controls less than a certain threshold number of maintainers. In any financial system, there exist various operations that are subject to regulatory rules. Examples of relevant entities developing and/or enforcing such rules are the Financial Conduct Authority (FCA) in the UK or the Securities and Exchange Commission (SEC) in the US. One important aspect of regulatory compliance is KYC; in our CBDC system abstraction, we assume maintainers are responsible for onboarding users to the system, i.e., all accounts in the system that are introduced subject to the approval of the maintainers.

3. Users and Payment interface Providers (PIPs): As any digital currency system, in a CBDC system, the users can act as either the sender (a.k.a. buyer, payer, or customer) or the recipient (a.k.a. seller, payee, or merchant) of digital currency in a transaction. Users of the currency can be private individuals or organizations. Note that users engage with the system through software and/or hardware provided by a PIP. The distinction between users and PIPs will not be essential for our analysis and modeling, and we will not pursue it further. We assume that any number of users of the system are untrusted, i.e., they may behave maliciously against honest users or other system entities. Privacy of payments should be satisfied between an honest sender and an honest receiver in a transaction.

2.1 CBDC Security Requirements
In this section, we informally define security requirements that will be captured by our CBDC ideal functionality. Note that the CBDC system should be resilient against broad types of attacks (e.g., Sybil attacks, man-in-the-middle attacks etc.), however, the focus of this section is on explaining requirements which are more specific to payment systems and CBDCs; these are as follows.

1. Financial and Regulatory Integrity. No one should be able to update the account of another user. Furthermore, currency in circulation or the amount of CBDC that is used to conduct transactions between consumers and businesses does not change as the system evolves over time except when the central bank decides to create new money (digital currency). Double-spending prevention is a crucial requirement for any payment system. A specific balance of a user should not be used in two transactions without being updated each time. In addition, after a successful payment between two users, the account of both of them should be updated correctly considering all parameters that are included in users’ accounts for the purpose of checking financial and regulatory rules.

2. Comprehensive Regulatory Compliance. This term means achieving all the following four items at the same time.

(1) Balance Limit: It limits the amount of funds that a particular user can possess in a specific period of time. Bank of England [1] and a report from several Central Banks (that details the principles, motivations, and risks of CBDC) [11] have mentioned that balance limit can help prevent bank runs and evasion of wealth tax. Moreover, the Bank of England (BoE) [1] and the European central bank (ECB) [10] have addressed that to manage the implications of a CBDC for financial stability, limits of how much CBDC any individual can hold is necessary.

(2) Receiving and Sending Limit: It limits the amount of received and sent funds that a particular user can receive or send in a specific period of time. The sent and received amounts should not exceed a predefined threshold. European central bank [7], and several central banks [11] have mentioned that limiting receiving and sending values can help achieve AML and prevent tax evasion.

(3) Transaction Value Limit and KYT: Reporting requirements and disclosure of source of funds for large value transactions are typically required (e.g., in the US filing a report is required for transactions in cash exceeding $10,000). To reflect this, we have a limit on the value of each transaction. Furthermore we discuss how it is possible to comply with more complex KYT policies where users should disclose additional information for large value transactions.

(4) Auditable: In cases of suspicious activities, additional auditing actions are needed (for e.g., filing suspicious activity reports called SARs [2]). The auditing functionality has two components:
Privacy Revocation: Given an anonymous transaction, authorities can reveal the real world identities of involved parties and the transferred value of that transaction. Tracing: Given a real world identity of a user, authorities can trace anonymous payments in which the user has engaged (as a sender or recipient).

3. Full Privacy. This property means achieving all the following three items at the same time.

1. Identity Privacy: It means for any given transaction the real world identities of either the sender or the receiver cannot be revealed (except when auditing). Furthermore, given the identity of a specific user no one can find the transactions in which the user has involved as a sender or receiver.

2. Transaction Privacy: The transferred value by the sender to the recipient cannot be revealed (except when auditing) and given a specific amount of transferred value no one can find the transactions that match that same (or related) value. Only the sender and recipient should know the value of the transaction. Moreover, the account information of users (e.g., sum of all sent and received values) are hidden from all network entities.

3. Full Unlinkability: It contains two parts that are as follows. Identity Unlinkability: Given an anonymous payment’s real world identities of the sender or receiver it should not be possible to link the sender or receiver’s other transactions to the given transaction. Transaction Unlinkability: Given a transaction, it should not be possible to link any past transaction that resulted in the possession of the funds used by the current transaction.

4. Accountability. When a user makes a payment it should not be able to deny it later — there is an obligation to accept the responsibilities that come with a finalized transaction.

2.2 Notations

In this paper, for uniquely identifying parties, we denote the central bank by B, the user and its key pair with U and (pk_u, sk_u) respectively. U also has another secret key a used for generating per-transaction tracing tag. This tag is denoted by T. We denote the account of U by acc. The notation M_j is used for the j-th maintainer and M for the set of all maintainers. Each maintainer (e.g., M_j) has two pairs of keys for threshold encryption (pk_{1,j}, sk_{1,j}) and (pk_{2,j}, sk_{2,j}). M_j also has a pair of key for threshold signature (pk_j, sk_j). We assume |M| = D and there are two thresholds, α is the threshold number of maintainers required for verifying transactions on behalf of the central bank and the regulator, and β is the threshold number of maintainers required for executing the Auditing protocol. Maintainers of which β number is required for executing the Auditing protocol is called audit committee. Set of honest and malicious maintainers are denoted by H and C, and their associated identifiers (indexes) by H and C respectively. We assume |C| = t. Honest maintainer is denoted by M_w and malicious maintainer is denoted by M_m.

L_j denotes the j-th ledger maintained by j-th maintainer M_j which is initially empty. We denote the user record which is saved in L_j with U. The sender and receiver of a payment are denoted by U_s and U_r respectively. Hence, for instance the key pair of the sender is (pk_s, sk_s) and its tracing key is a_s. The value of transaction that is transferred from a sender (B or U_s) to a recipient is denoted by v and the transaction identifier is denoted by t_{id}.

The balance of U is denoted by B, and sum of all sent and received values of U by S and R respectively. B_{max}, S_{max}, R_{max}, and V_{max} are regulatory limits on maximum allowed: balance, sum of all sent values, sum of all received values, and transaction value respectively. We denote transaction counter of a user which is incremented for each transaction (Currency Issuance or Payment) by x (the statement of Zero-Knowledge is denoted by x). The notation \{e_j\}_{j=1}^N is used to denote a set \{e_1, ..., e_N\} with N elements. If for every positive polynomial p, there is an integer i_0 where for all integers i > i_0, negl(i) < \frac{1}{p(i)} holds, the function negl is negligible. We use F_q to denote a field with q elements. PPT stands for probabilistic polynomial time.

2.3 CBDC Formal Model

We formalize the objectives of a CBDC system as an ideal functionality in the Universal Composition framework [16]. The central bank digital currency scheme consists of six main sub-protocols: User Registration, Currency Issuance, Payment, Abort Transaction, Privacy Revocation and Tracing. The last two are called Auditing. Valid transactions are recorded in the ledger L of each maintainer M. Hence, there is a history of all verified transactions accessible by anyone who is permissioned to audit private transactions.

F_{CBDC} is parameterized by D, t, V_{max}, B_{max}, S_{max}, and R_{max} where D = 3t + 1 holds. The functionality F_{CBDC} maintains the following tables and mappings: T(U) outputs 0 if U has not been traced and 1 if it has been traced. Initially, T(U) = ⊥ meaning that for non-registered users T(U) outputs ⊥. Users to their accounts’ state: W = (B, S, R, x) ⇐ E(U). Initially, E(U) = ⊥. U(O) outputs pid if the user U has ongoing transaction with pid. Once the transaction is finalized (in the real world the user receives α valid signature shares on its new account) U(O) is set to ⊥ meaning that user is in the Idle state, therefore, can start a new transaction. Payment identifiers pid to transaction identifiers t_{id}: t_{id} ⇐ P(pid). Set of maintainers who engage in a specific transaction whose identifier is t_{id}: M(t_{id}). Users to their most recent transaction metadata or transaction identifier (U_s, U_r, t_{id}, o) ⇐ Tid(U) where U = U_s or U = U_r, or (B, U, t_{id}, o) ⇐ Tid(U). Initially, Tid(U) = ⊥. Transaction identifiers to transaction metadata (U_s, U_r, o) ⇐ Rvk(t_{id}). Users to all their transaction identifiers and their role in each of them \{t_{id'}^{role_i}\}_{i=1}^{t_{id}} ⇐ Trc(U).

We note that session identifiers are of the form sid = (B, m, sid’) such that m = \{M\}_j^{D} \cdot 1. Initially, init ← 0 where init ∈ \{0, 1\}. At the end of Initialization init is set to 1. Afterwards in the beginning of all parts of the functionality (namely User Registration, Currency Issuance, Payment, Abort Transaction, Privacy Revocation and Tracing) it is checked whether init has been set to 1. If it has not been set to 1, F_{CBDC} ignores the received message. In F_{CBDC}, by sending a message m to m via delayed output, we mean the following. F_{CBDC} provides m and unique identifiers of all maintainers in the set m to the ideal-world adversary A. F_{CBDC} lets A decide the order of maintainers in the set m who receives the message m. Also it can delay the message delivery and prevent delivering a message to a maintainer.
**Functionality $\mathcal{FCBDC}$**: part I: Registration and Issuance

**Initialization.**
1. Upon input ($\text{Init}, \text{sid}$) from party $P \in \{B,M\}$: Abort if $\text{sid} \neq (B,M, \text{sid}')$. Else, output ($\text{InitEnd}, \text{sid}, P$) to $\mathcal{A}$. Once all parties have been initialized, set init $\leftarrow 1$.

**User Registration.**
1. Upon receiving a message ($\text{GenAcc}, \text{sid}$) from $U$: If $\mathbb{E}(U) = \bot$, output ($\text{GenAcc}, \text{sid}, U$) to $\mathcal{A}$. Else, ignore.
2. Upon receiving ($\text{ok,GenAcc}, \text{sid}, U$) from $\mathcal{A}$: Output ($\text{AccGenEd}, \text{sid}, U$) to $\mathcal{M}$ via public-delayed output. Output ($\text{AccGenEd}, \text{sid})$ to $U$ via public-delayed output and set $\mathbb{E}(U) \leftarrow (0,0,0,0)$ and $T(U) \leftarrow 0$ when delivered.

**Currency Issuance.**
1. Upon receiving a message ($\text{iss}, \text{sid}, U, v$) from $B$: Ignore if $B$ not in sid. Else, generate a new pid. If $U$ is corrupted, output ($\text{iss}, \text{sid}, \text{pid}, U, v$) to $\mathcal{A}$. Else, output ($\text{iss}, \text{sid}, \text{pid})$ to $\mathcal{A}$.
2. Upon receiving ($\text{AcceptIss}, \text{sid}, \text{pid}, v, U$) from $U$: If $\mathbb{E}(U) = \bot$ or $U(\mathbb{E}(U)) \neq \bot$, ignore. Else, retrieve $W \leftarrow \mathbb{E}(U)$. If $B + v > B_{\text{max}}$ or $R + v > R_{\text{max}}$, ignore. Else, set $U(\mathbb{E}(U)) \leftarrow \text{pid}$ and retrieve $T(U)$: (a) If $T(U) = 0$, output ($\text{AcceptIss}, \text{sid}, \text{pid})$ to $\mathcal{A}$. (b) Else, output ($\text{AcceptIss}, \text{sid}, \text{pid}, U$) to $\mathcal{A}$.
3. Upon receiving ($\text{GenTxn}, \text{sid}, \text{pid}, t_{\text{id}}$) from $\mathcal{A}$: If already exits a pid’ $\neq \text{pid}$ where $t_{\text{id}} \leftarrow P(\text{pid}')$, ignore. Else, if $P(\text{pid}) = \bot$, set $P(\text{pid}) \leftarrow t_{\text{id}}$. Else, retrieve $t_{\text{id}}' \leftarrow P(\text{pid})$, ignore if $t_{\text{id}}' \neq t_{\text{id}}$. Set $T(\text{id}) \leftarrow (B, U, t_{\text{id}}, v)$.
4. Upon receiving ($\text{GenTxn}, \text{sid}, \text{pid}, M_k$) from $\mathcal{A}$: Ignore if $\bot \leftarrow P(\text{pid})$. Else, retrieve $t_{\text{id}} \leftarrow P(\text{pid})$. Set $M(t_{\text{id}}) \leftarrow M(t_{\text{id}}) \cup M_k$ and output ($\text{TxnDone}, \text{sid}, t_{\text{id}}$) to $M_k$ via public-delayed output. Once $|M(t_{\text{id}})| \geq \beta$: Set $\mathbb{E}(U) \leftarrow (B + a, S, R + a, x + 1)$, $\text{Rvk}(t_{\text{id}}) \leftarrow (B, U, v)$, $\text{Trc}(U) \leftarrow \text{Trc}(U)$, $\text{Trc}(U) \leftarrow (t_{\text{id}}, \text{receiver})$. Once $|M(t_{\text{id}})| \geq \alpha$: Output ($\text{TxnDone}, \text{sid}, B, v$) to $U$ via private-delayed output and set $U(\mathbb{E}(U)) \leftarrow \bot$ when delivered.

In more details the components of our functionality are as follows. \(\mathcal{I}\) **Initialization.** This step merely ensures that the relevant parties ($B$ and all $D$ maintainers $M$) have been activated; the functionality keeps a record of all parties that have been initialized for the scheme. \(\mathcal{U}\) **User Registration.** At the registration phase, a user $U$ should get their account ratified by the system. If the user $U$ has already been registered by maintainers $M$, it cannot be registered again. Note that as it is common in the Universal Composition setting, we allow the adversary $\mathcal{A}$ communications and hence also block registration (i.e., we do not model denial of service attacks). The balance and regulation-related information of user $W$ is set to initial values which are zero for balance $B$, sum of all sent values $S$, sum of all received values $R$ of user, and transaction counter $x$. The maintainers are notified for each successful user registration. \(\mathcal{C}\) **Currency Issuance.** In the Currency Issuance process different from Payment, only the central bank $B$ is allowed to be the payer and there are no limits imposed to the funds that the central bank possesses. First of all, functionality $\mathcal{FCBDC}$ checks whether the receiver of digital currency $U$ is a valid registered user in the system or not which means if the user $U$ has not already been registered by maintainers $M$, it cannot obtain any digital currency. The functionality imposes the regulatory restrictions of $B + v \leq B_{\text{max}}$ and $R + v \leq R_{\text{max}}$, where $v$ is the amount of currency that is issued following the central bank’s instructions. We remark that based on different regulatory rules in each jurisdiction, some of the restrictions such as upper bounding the value central bank $B$ issues $v \leq v_{\text{max}}$ can be easily captured or the mentioned checks $B + v \leq B_{\text{max}}$ and $R + v \leq R_{\text{max}}$ can be ignored for currency issuance transactions (note that as our construction is account-based rather than token-based; adding or removing such regulatory compliance constraints is relatively straightforward). Currency issuance is not a unilateral action from the central bank but as it also needs the activation of the user $U$ who receives the digital currency. This highlights one of the distinctions of our setting compared to blockchain systems: the recipient of funds $U$ is online during transaction and the protocol is interactive. The state of the receiver’s account is updated after each currency issuance action. As before, the adversary $\mathcal{A}$ may block the currency issuance from going forward. A successful currency issuance will increase the balance of the receiver $U$ by the indicated amount $v$. Transaction value, and identity of the receiver is hidden from the adversary. The ideal-world adversary $\mathcal{A}$ is also required to assign a unique transaction identifier $t_{\text{id}}$ and all transaction metadata are stored by the functionality in a table $\text{Rvk}(t_{\text{id}})$ while the $t_{\text{id}}$ is stored in $\text{Trc}(U)$, where $U$ is the recipient. \(\mathcal{P}\) **Payment.** As in the case of Currency Issuance, the Payment process involves both the sender $U_s$ and the receiver $U_r$ being activated. Contrary to issuance transaction, the functionality during payment performs the important check that the sender $U_s$ has sufficient balance to fund the payment $B_s - v \geq 0$. Interactive payment is necessary as we claim that $\mathcal{FCBDC}$ captures regulatory compliance (e.g., AML, CFT) considering both parties which means both of them are supposed to know with whom they are making a payment. Hence, it is vital for the receiver $U_r$ to actively engage in each payment. A successful payment protocol will increase the balance of the receiver $U_r$ by the indicated amount $v$ as well as subtract that amount from the balance of the sender $U_s$. Additionally, account information of each user is updated to capture different regulation policies. As in the case of issuance a unique transaction identifier $t_{\text{id}}$ is determined by the ideal-world adversary $\mathcal{A}$ and the transaction metadata are stored in table $\text{Rvk}(t_{\text{id}})$ while the $t_{\text{id}}$ is stored in $\text{Trc}(U_r)$ and $\text{Trc}(U_s)$, where $U_r$ and $U_s$ are the sender and recipient of the payment. Note that the adversary $\mathcal{A}$ is not aware of the transaction value, and identities of sender and receiver (unless one of them is malicious) and the $t_{\text{id}}$ is selected independently of them. \(\mathcal{A}\) **Abort Transaction.** The user initiates aborting transaction by which it requests an update on its account’s state. The update which user acquires on its account (either in Currency Issuance or Payment). The engagement corresponds to signing the account of the user in the real world. If less than or equal to $t$ (either malicious or honest) maintainers have engaged with the most recent transaction of the user, the user’s transaction gets rejected meaning that the state of the account is not changed (only $x$ is updated). Otherwise, the transaction is confirmed and the state of the sender and receiver’s account is updated (in case of Currency Issuance transaction only
the receiver’s account is updated). \textcircled{6} Privacy Revocation. Privacy revocation is initiated by the maintainers who submit the transaction identifier of a fully anonymous payment they wish to revoke. If a sufficient number of them (this is set to \( \beta \)) agrees on the revocation of a specific transaction the functionality will recover the metadata of the specific transaction and return them to the maintainers and adversary. \textcircled{7} Tracing. As in the case of revocation, the maintainers have to agree they want to trace a specific user. If the quorum is reached (requiring \( \beta \) maintainers) then the set of transaction identifiers that correspond to the agreed users will be returned to the maintainers and adversary.

### Functionality \( F_{CBDC} \), part II: Payment and Auditing

#### Payment.
1. Upon receiving a message \((\text{GenTxnSnd}, \text{sid}, U_r, v)\) from \( U_i \): If \( \mathbb{E}(U_i) = \bot \) or \( \mathbb{E}(U_r) \neq \bot \), ignore. Else, retrieve \( W_r = \mathbb{E}(U_r) \). If \( S_r + v > S_{\text{max}} \) or \( B_r - v < 0 \), or \( v > V_{\text{max}} \), ignore. Else, generate a new pid and set \( U_r \leftarrow \text{pid} \). If \( U_r \) is corrupted, output \((\text{GenTxnSnd}, \text{pid}, \text{sid}, U_r, v)\) to \( \mathbb{A} \). Else, retrieve \( T(U_r) \): (a) If \( T(U_r) = 0 \), output \((\text{GenTxnSnd}, \text{pid}, \text{pid})\) to \( \mathbb{A} \). (b) Else, output \((\text{GenTxnSnd}, \text{pid}, \text{pid})\) to \( \mathbb{A} \).
2. Upon receiving \((\text{GenTxnRcv}, \text{sid}, U_r, v)\) from \( U_r \): If \( \mathbb{E}(U_r) = \bot \) or \( \mathbb{E}(U_r) \neq \bot \), ignore. Else, retrieve \( W_r = \mathbb{E}(U_r) \). If \( B_r + v > B_{\text{max}} \) or \( R_r + v > R_{\text{max}} \), ignore. Else, set \( T(U_r) \leftarrow \text{pid} \) and retrieve \( T(U_r) \): (a) If \( T(U_r) = 0 \), output \((\text{GenTxnRcv}, \text{sid}, \text{pid})\) to \( \mathbb{A} \). (b) Else, output \((\text{GenTxnRcv}, \text{sid}, \text{pid})\) to \( \mathbb{A} \).
3. Upon receiving \((\text{GenTnx}, \text{sid}, \text{pid}, t_{id})\) from \( \mathbb{A} \): If already exits a pid’ \( \neq \text{pid} \) where \( t_{id} \leftarrow P(\text{pid’}) \), ignore. Else, if \( P(\text{pid}) = \bot \), set \( P(\text{pid}) \leftarrow t_{id} \). Else, retrieve \( t_{id} \leftarrow P(\text{pid}) \), ignore if \( t_{id} \neq t_{id} \). Set \( \text{Tid}(U_r) \leftarrow (U_r, t_{id}, v) \) and \( \text{Tid}(U_j) \leftarrow (U_j, t_{id}, v) \).
4. Upon receiving \((\text{GenTnx}, \text{sid}, \text{pid}, M_k)\) from \( \mathbb{A} \): Ignore if \( \bot \leftarrow P(\text{pid}) \). Else, retrieve \( t_{id} \leftarrow P(\text{pid}) \). Set \( M(t_{id}) = M(t_{id}) \cup M_k \), and output \((\text{TnxDone}, \text{sid}, t_{id})\) to \( M_k \) via public-delayed output. Once \( |M(t_{id})| \geq \beta \), set \( \mathbb{E}(U_r) \leftarrow (B_r - v, S_r + v, R_r, x_r + 1) \), \( \mathbb{E}(U_j) \leftarrow (B_r + v, S_r - v, R_r, x_r + 1) \), \( \text{Rvk}(t_{id}) \leftarrow (U_s, U_r, v), \mathbb{E}(U_r) \leftarrow \mathbb{E}(U_j) \cup (t_{id}, \text{sender}) \), and \( \mathbb{E}(U_r) \leftarrow \mathbb{E}(U_j) \cup (t_{id}, \text{receiver}) \). Once \( |M(t_{id})| \geq \alpha \): Output \((\text{TnxDone}, \text{sid}, U_r, v)\) to \( U_r \) via private-delayed output and set \( U_r \leftarrow \bot \) when delivered. Output \((\text{TnxDone}, \text{sid}, U_r, v)\) to \( U_i \) via private-delayed output and set \( U_r \leftarrow \bot \) when delivered.

#### Abort Transaction.
1. Upon receiving a message \((\text{AbortTnx}, \text{sid})\) from \( U_i \): If \( \mathbb{E}(U_i) = \bot \) or \( \text{Tid}(U) = \bot \), ignore. Else, retrieve \((U_s, U_r, t_{id}, v) \leftarrow \text{Tid}(U) \). Send \((\text{AbortTnx}, \text{sid}, t_{id})\) to \( \mathbb{A} \).
2. Upon receiving \((\text{AbortTnxOk}, \text{sid}, t_{id})\) from \( \mathbb{A} \): Set \( \text{Tid}(U) \leftarrow \bot \). (a) If \( |M(t_{id})| < \beta \), set \( \mathbb{E}(U) \leftarrow (B, S, R, x + 1) \), \( \mathbb{E}(U) \leftarrow \mathbb{E}(U_j) \cup (t_{id}, \text{Aborted}) \). Output \((\text{TnxAbort}, \text{sid}, \text{tid})\) to \( U_i \) via public-delayed output and set \( U_r \leftarrow \bot \) when delivered. Output \((\text{TnxAbort}, \text{sid}, t_{id})\) to \( M \) via public-delayed output. (b) Else, given the retrieved tuple \((U_s, U_r, t_{id}, v)\):

Output \((\text{TnxDone}, \text{sid}, U_r, v)\) to \( U_r \) via private-delayed output and set \( U(U_r) \leftarrow \bot \) when delivered. Output \((\text{TnxDone}, \text{sid}, U_r, v)\) to \( U_i \) via private-delayed output and set \( U(U_r) \leftarrow \bot \) when delivered. Output \((\text{TnxDone}, \text{sid}, t_{id})\) to \( M \) via public-delayed output.

#### Privacy Revocation.
1. Upon receiving a message \((\text{RvkAnn}, \text{sid}, t_{id})\) from maintainer \( M_j \): If \( \text{Rvk}(t_{id}) = \bot \), ignore. Else, record \((\text{RvkAnn}, \text{sid}, t_{id}, M_j)\) and output \((\text{RvkAnn}, \text{sid}, t_{id}, M_j)\) to \( \mathbb{A} \). Once \( |\{j|t_{id} = t_{id}\}| \geq \beta \), set \( X \leftarrow t_{id} \).
2. Upon receiving \((\text{RvkAnnOk}, \text{sid}, t_{id})\) from \( \mathbb{A} \): If \( X \) has not already been set to \( t_{id} \), ignore. Else, retrieve \((U_r, U_r, v) \equiv \text{Rvk}(X) \). Output \((\text{AnnRevoked}, \text{sid}, t_{id}, U_r, v)\) to \( M \) via public-delayed output.

#### Tracing.
1. Upon receiving a message \((\text{Trace}, \text{sid}, U_j)\) from maintainer \( M_j \): If \( \mathbb{E}(U_j) = \bot \), ignore. Else, record \((\text{Trace}, \text{sid}, U_j, M_j)\) and output \((\text{Trace}, \text{sid}, U_j, M_j)\) to \( \mathbb{A} \). Once \( |j|U_j = U_j| \geq \beta \), set \( Y \leftarrow U_j \).
2. Upon receiving \((\text{TraceOk}, \text{sid}, U)\) from \( \mathbb{A} \): If \( Y \) has not already been set to \( U \), ignore. Else, retrieve \((B, S, R, x) \leftarrow \mathbb{E}(Y) \). Retrieve \((t_{id}, \text{role})\) from \( \mathbb{A} \). Set \( T(U) \leftarrow 1 \). Output \((\text{Traced}, \text{sid}, t_{id}, \text{role})\) to \( M \) via public-delayed output.

*Either \( U = U_r \) or \( U = U_j \) holds.

### 3 OUR CONSTRUCTION

In our construction, we aim to achieve all the financial, regulatory and security properties described informally in Sec. 2.1 and formally in Sec. 2.3. We assume that the whole number of maintainers are \( 3t + 1 \) and \( t \) of them can be corrupted by the adversary. Hence, we set the thresholds of blind signature scheme and auditing as \( \alpha = 2t + 1 \) and \( \beta = t + 1 \) respectively.

#### 3.1 High-level Technical Overview

Every user in the system has an account acc for storing the current balance \( B \) and other user specific values related to the system’s financial and regulatory restrictions. Users update their accounts when transacting. For each new currency issuance or payment transaction, the involved parties in the transaction engage in a cryptographic protocol with all maintainers \( M \). To this end, users encode the values of accounts into cryptographic one-time objects that fix a unique tag \( T \). When updating an account a user discloses the tag associated to the previous account snapshot acc (which has been signed by at least \( \alpha \) maintainers). A user also discloses \( \sigma_{\text{Acc}} \) that is a re-randomization of the consolidated signature \( \sigma_{\text{Acc}} \) on their previous account snapshot. The disclosed tags are stored by maintainers for the purpose of enforcing users to use their most updated accounts (as in Chaum’s double-spending prevention for online cash [19]). To support tracing, the protocol in fact computes tags pseudo-randomly so that they can be recomputed by the Auditing
Upon sending TI, U’s state is changed to Receiving (from central bank B or from another user U_s) or Sending (to another user U_r).

2. One of the states Receiving or Sending (which means U’s most recent transaction is pending), U ignores Z’s message.

When state is changed from Idle to Receiving or Sending, the transaction can be successful or pending as explained in the following cases:

1. Successful (e.g., payment participants use their newly updated accounts, regulatory compliance is met, and maintainers have received valid transaction information of both payment participants). U receives at least a valid blind signature shares of maintainers on acc\_new.\_B. Upon generating unblinded-consolidated maintainers’ signature on the new account \(\sigma_{\text{ml}}\), state is changed to Idle. Hence, U who, now, has its new account signed is ready to enter into the next transaction.

2. Pending (e.g., the sender-receiver pair has not been generated on sufficiently enough maintainers’ sides). U’s state remains in Receiving or Sending up to the moment when Z instructs U to send an abort request AR.

Upon Z’s instruction (of the form (Abr Tnx, sid)) for sending abort request AR (which includes U’s refreshed-blinded account acc\_r.B), U sends AR to m\_B (in this case, if the state of U is not Sending or Receiving, it ignores Z’s instruction). Doing so changes U’s state from either Sending or Receiving to Aborting. The two following scenarios are for the case when U is in the Aborting state:

1. If at least \(t + 1\) maintainers have saved a sender-receiver TI pair in their ledgers (which guarantees at least one honest maintainer has the pair), maintainers ignore acc\_r.B and send their signatures for acc\_new.\_B to U. Upon generating unblinded-consolidated maintainers’ signature on the new account \(\sigma_{\text{ml}}\), state is changed to Idle.

2. Else, maintainers sign acc\_r.B, record the pending transaction as aborted and ignore acc\_new.\_B included in TI. Upon generating unblinded-consolidated maintainers’ signature on the refreshed account \(\sigma_{\text{ml}}\), state is changed to Idle.

Furthermore, you can find a pictorial representation of all the sub-protocols of our construction in the full version of the paper [26].

3.2 Details of the Construction

In this section, we describe our CBDC protocol \(\Pi_{\text{PEReDi}}\). We will prove that \(\Pi_{\text{PEReDi}}\) securely realizes \(\mathcal{F}_{\text{CBDC}}\). Our construction uses several concrete cryptographic components and ideal functionalities (see the full version of the paper [26]). Our scheme uses the Coconut Threshold Blind Signature scheme (TBS) [30, 33] and the Threshold ElGamal Encryption (TE) scheme [23, 25, 28] in a mostly blackbox manner. However, we reduce the unforgeability of Coconut to its underlying Pointcheval-Sanders [29] signature component. Throughout this section when we use Coconut we employ its algorithms as described in the full version of the paper [26]. However, whenever possible we merge its 2K proofs with those of the rest of the protocol for improving performance. PEReDi employs the following functionalities: a Key-Registration functionality \(\mathcal{F}_{\text{KR}}\), a communication Channels functionality \(\mathcal{F}_{\text{Ch}}\) (parameterized by different labels, e.g., “sa” for a sender anonymous channel \(\mathcal{F}_{\text{Ch}}^{sa}\)).
a Broadcast functionality $F_{BC}$, a Byzantine Agreement functionality $F_{BA}$, a Random Oracle functionality $F_{RO}$, a Non-Interactive Zero Knowledge functionality $F_{NIZK}$ and a Signature of Knowledge functionality $F_{SK}$.

We will assume that transferring parties communicate through variants of $F_{Ch}$ as specified. We note that some sender-anonymity is necessary for privacy, as otherwise network "leakage" will trivially reveal the counterparties of a transaction irrespective of the strength of cryptographic protections at the transactional level. We note that in a real-world deployment such network leakage may be considered tolerable — our analysis would apply directly to such setting as well, exhibiting the unavoidable concession that the adversary may break privacy via traffic analysis.

Throughout this section we use the notation of Sec. 2.2. Each maintainer $M$ has its own ledger $L$ for storing registration and transaction information. In the Currency Issuance and Payment protocols of the construction below, the sender ($U_i$ or $B$) and receiver ($U_j$ or $U_k$) separately send their transaction information $T_i$ to all maintainers $M$. However, a plausible alternative communication pattern could have the sender sending its transaction information $T_i$ to the receiver and then the receiver sending both the sender’s $T_i$ and its own $T_i$ to $M$. The public key of threshold encryption, the ciphertexts and tracing tags all are from $G$, and we use Bilinear maps for threshold blind signature (see the full version of the paper [26] for more details).

### 3.2.1 Initialization

The key generation algorithm takes the security parameter as input and generates the secret key $sk$ and public key $pk$ for the caller of algorithm as outputs. Participants of the network independently call the key generation algorithm for each underlying cryptographic scheme to generate their keys (see the full version of the paper [26] for key generation algorithms and see Section 2.2 for the notations). The public keys of all parties are maintained in a public-key directory and are assumed to be accessible on demand by calling $F_{Ch}$ with input $(\text{RetrieveKey}, \text{sid}, \text{P})$ for party $P$.

### 3.2.2 User Registration

Maintainers $M$ enroll a user $U$ in the CBDC system by creating a signature on the user’s initial account. Afterwards, $U$ uses the signature to create transactions. For registration, $U$ with a pair of public-secret key $(pk_U, sk_U)$ and a secret key $a$ (used in tag generation) engages in a threshold blind signature $TBS$ protocol with $M$ where $U$ proves honest creation of its initial account to $M$. The output of this protocol is a signed account $σ_{M}$ for $U$ (needed for its first transaction) and the user record $UR$ saved in the ledger $L$ of each maintainer $M$ (required for additional investigation during the Auditing protocol). Every user’s account consists of a tuple of field elements $acc = (B, S, R, sk_U, a^x, a)$. During registration, $U$ sets $B, S, R$ and $x$ to 0.

Upon receiving $(\text{GenAcc}, \text{sid})$ (from $Z$), $U$ who is initially in the Idle state initiates the User Registration protocol (see the full version of the paper [26] for the high-level picture of this protocol) to get the account signed by $M$. $U$ generates its registration information $R_{ij}$ ($a_{ij}, r_{ij}, com_{M}, sk_{U}, π$) as follows:

1. $(acc^U, \{σ_{M}\})$" prepares BlindSign$(acc, \cdot)\$, and calls $(a_{ij})_{j=1}^D \to \text{SSH.Share}^D(\cdot)\$ to secret share $a$ and computes $\cdot \text{com}_j = g^{a_j \cdot h_j}$ for $r_j \to Z_p^n$. It sets $com_{M} = (\text{com}_j)_{j=1}^D$.

2. Calls $F_{NIZK}$ with input $(\text{Prove}, \text{sid}, x, w)$, and receives $(\text{Proof}, \text{sid}, \pi)$ where $\pi$ is a NIZK proof of knowledge for statement $x = (acc^U, \text{com}_M, pk_U)$ and witness $w = (acc, \{a_{ij}\}_{j=1}^D, n_{bacc}, r_{com})$. We denote the randomness used to create the blinded account $acc^U$ and the commitment $com_{M}$ by $b_{bacc}$ and $r_{com}$, respectively and define the relation $R(x, w)$ of NIZK as follows (for formal definition the relation and the associated Sigma protocol see the full version of the paper [26]): $\text{1.}$ The secret key $sk_U$ in the blinded account $acc^U$ is the secret key associated with public key $pk_U$. $\text{2.}$ The secret key $a$ in $acc^U$ is the same as the secret key that can be reconstructed from the shares $(\{a_{ij}\}_{j=1}^D)$ committed in $com_{M}$.

3. Calls $F_{BC}$ with $(Broadcast, \text{sid}, com_{M})$, and then calls $(\text{Send}, \text{sid}, M_j, R_{ij})$ to the secure channel $F_{Ch}$ for $j = 1, \ldots, D$. Specifically, $U$ calls $F_{Ch}$ with the input $(\text{Send}, \text{sid}, M_j, R_{ij})$ (1 $\leq k \leq D = 1$) and waits for $F_{Ch}$ to send back $(\text{Continue}, \text{sid})$ then $U$ proceeds by calling $F_{Ch}$ with the input $(\text{Send}, \text{sid}, M_k, R_{k+1})$.

Each maintainer $(M_j)$:

1. Generates pairs of messages. Each pair contains the received message from $F_{BC}$ and $F_{Ch}$ where both messages have the same identifier $U$ of the user. In other words, it receives $(Broadcast, \text{sid}, U^\prime, com_{M})$ from $F_{BC}$ and $(Receive, \text{sid}, U^\prime, R_{ij})$ from the secure channel $F_{Ch}$. If $U^\prime = U$, $M_j$ generates a pair of messages containing the received messages from $F_{BC}$ and $F_{Ch}$. Else, waits to receive such messages.

2. If $com_{M}$ received from $F_{BC}$ is not equal to $com_{M}$ included in $R_{ij}$ received from $F_{Ch}$, aborts.

3. Else, ignores the message if at least one of the following conditions holds: (i) There already exists a user record $UR'$ in $L_j$ where $U' = U$. (ii) Upon calling $F_{Ch}$ with $(\text{RetrieveKey}, \text{sid}, U)$, it receives $(\text{KeyRetrieved}, \text{sid}, U, pk)$ such that $pk_U \neq pk'$. (iii) Upon calling $F_{Ch}$ with $(\text{Verify}, \text{sid}, x, π)$, it receives $(\text{Verification}, \text{sid}, 0)$. (iv) Given $(a_{ij}, r_{ij})$ received from $F_{Ch}$ included in $R_{ij}$, it computes $g^{a_j \cdot h_j}$ which is not equal to $\text{com}_j$ for $\text{com}_j \in com_{M}$. (v) Know Your Customer (KYC) guidelines for $U$ is not verified.

4. Else, the user record: $UR = (a_{ij}, r_{ij}, com_{M})$, $U$ is saved in $L_j$, $a_j$ $\rightarrow$ BlindSign$(sk_k, \cdot, acc^U, \cdot)$, calls authenticated channel $F_{Ch}$ with input $(\text{Send}, \text{sid}, U, σ^U_j)$, and outputs $(\text{AccSigned}, \text{sid}, U)$ (to $Z$).

$^2$Given acc, it calls $\text{PrepareBlindSign}$ algorithm of the threshold blind signature scheme $TBS$ to obtain a blinded account $acc^U$, $\{σ_{M}\}$, $a_{ij}$ are random field elements. Here, as explained in the beginning of this section, in contrast to the original $\text{PrepareBlindSign}$ algorithm of Coconut the algorithm does not create a proof. All necessary $ZK$ proofs are included in (2).

$^3$Signs associated information of $acc^U$ using the BlindSign algorithm of TBS scheme to obtain blind signature share $σ^U_j$. 
The user U: 1 Receives (Received, sid, M_j, σ^R_j) for different j from the authenticated channel F_{Ch}^ac 2 σ_j ← Unblind((σ_1^R)^j, σ^R_j). 3 σ^R_j ← TBS.Agg((σ_j)^R_j, pk)^5 5 and outputs (AccGened, sid) (to Z).

3.2.3 Currency Issuance. Upon receiving (Iss, sid, U, ρ) (from Z), B initiates Currency Issuance protocol (as shown in the picture in the full version of the paper [26]). To issue a digital currency worth of \( \rho \) of the secret key generated under the public key of maintainers worth of \( \sigma \), the exponent of \( \sigma \) is re-randomization of \( \sigma \) by B. Upon receiving (AcceptIss, sid, pid, ω) (from Z), if U is in idle state, it sends the fresh randomness \( ρ \) to B using the secure-server anonymous channel F_{Ch}^ac with input (Send, sid, M_j, T_{1j}) for j = 1, . . . , D. Specifically, B calls F_{Ch}^ac with the input (Send, sid, M_j, T_{1j}) (1 ≤ k ≤ D − 1) and waits for functionality to send back (Continue, sid), then U proceeds by calling F_{Ch}^ac with the input (Send, sid, M_k+1, T_{1j}).

Upon receiving (Received, sid, U, ρ) from the secure-server anonymous channel F_{Ch}^ac, B also sends its transaction information T_{1k} = ψ to \( m \). The central bank B calls the authenticated channel F_{Ch}^ac with input (Send, sid, M_j, T_{1k}) for j = 1, . . . , D. Specifically, B calls F_{Ch}^ac with the input (Send, sid, M_k, T_{1j}) (1 ≤ k ≤ D − 1) and waits for F_{Ch}^ac to send back (Continue, sid) then B proceeds by calling F_{Ch}^ac with the input (Send, sid, M_k+1, T_{1j}). Each main user (M_j):

1. Receives (Received, sid, T_{1k}, mid) from the sender anonymous channel F_{Ch}^ac and parses T_{1k} as \( (ψ, acc^{new}, σ^R_{1j}, T, π) \). The components of T_{1k} is computed by U which does the following:
   1. Computes threshold ElGamal encryption as setting its public key pk_{U} and \( g^α \) as plaintexts: \( ψ = (ψ_1, ψ_2, ψ_3) = (g^α, pk_{U}, pk_{U}^2 \cdot g^α) \).
   2. Computes acc^{new}, \( σ^R_{1j} \). Similar to acc \( σ_{1j} \) at User Registration protocol, to obtain blind encryption of \( < \text{acc}^{new}, σ^R_{1j}> \) on U’s new account which is as follows: acc^{new} = (\( B^{new}, \text{new acc}, σ^R_{1j}, T, π) \). U should prove that it has a valid signature \( σ_{1j} \) on its previous account acc and request a new signature on its new account acc^{new}.
   3. Computes T = \( g^{α^R} \) that is a tag used for compelling users to use their most updated accounts in which \( ψ \) is an incrementing value per transaction. As we will see, same value is used for tracing the user when it is necessary.
   4. Calls F_{NZK} with input (Proof, sid, x, w), and obtains (Proof, sid, π) from it in which \( π \) is a NIZK proof for the statement \( x = (ψ, acc^{new}, σ^R_{1j}, T) \). We denote the randomness used to create acc^{new}, \( σ^R_{1j} \) and threshold encryption \( ψ \) by \( \text{reg} \). The witness of \( π \) is \( π = (acc, \text{reg}, w) \) for the following relation R(x, w) (for formal definition of the relation and the associated Sigma protocol see the full version of the paper [26]): The secret key sk_{U} used in acc^{new}, \( σ^R_{1j} \) is the secret key associated with public key pk_{U}, in the threshold encryption \( ψ \) generated under the public key of maintainers \( pk_{1L,M} \). T is well-formed, the exponent of \( g \) is the fifth element in acc^{new}. \( σ^R_{1j} \) is re-randomization of \( σ_{1j} \) which is a signature generated by aggregating \( a \) different valid signature shares of maintainers on acc. \( acc^{new}, σ^R_{1j} \) is generated considering acc and \( ψ \) in \( α \). Hence, \( B^{new} = B^{old} + v, \text{new acc} = \text{old acc} + R^{old} + v, sk_{U} = sk_{U}, \sigma^{α+1} = a^α \cdot a, \text{a} = v \).

5. Calls the sender anonymous channel F_{Ch}^ac with input (Send, sid, M_j, T_{1j}) for j = 1, . . . , D. Specifically, U calls F_{Ch}^ac with the input (Send, sid, M_j, T_{1j}) (1 ≤ k ≤ D − 1) and waits for functionality to send back (Continue, sid), then U proceeds by calling F_{Ch}^ac with the input (Send, sid, M_k+1, T_{1j}).
fully anonymous channel \( F_{\text{Ch}}^{fa} \) with input (Send, sid, U, \( \rho_r \)).

Upon receiving (GenTnxRcv, sid, U, \( \sigma \)) (from Z), if \( U \) is in one of the Sending or Receiving state, it ignores the message.

1749

There already exists a transaction identifier \( T_1 \). 1 Upon calling

\[ \Box \]

The witness of \( TI \) is \( R \) from \( \sigma \text{sk} \) is a signature generated by aggregating \( \sigma \text{acc} \) is a signature generated by aggregating

\[ \Box \]

\[ \Box \]

\[ \Box \]


The witness of \( \Box \) for the statement \( x = (\psi_1, \sigma_{\text{acc}}^{\text{new}}, \Box \Box) \) and the message of signature \( \psi_1 \).

We denote the set of all random values \( \sigma_{\text{acc}}^{\text{new}}, \sigma_{\text{sk}}^{\Box \Box} \) on \( \psi_1 \) by \( R \).

The witness of \( \Box \) for the statement \( x = (\psi_1, \sigma_{\text{acc}}^{\Box \Box}, \Box \Box) \) and the message of signature \( \psi_1 \).

The transaction information of \( U_r, T_r \) is similar to \( T_r \) with values associated to \( U_r \) which is \( T_1 = (\psi_r, \sigma_r, \Box \Box) \) for the following relation \( \Box \Box \) (for formal definition of the relation and the associated Sigma protocol see the full version of the paper [26]).

The secret key \( \Box \Box \) is the secret key associated with public key \( \Box \Box \) in the threshold encryption \( \psi_1 \) generated under the public key of maintainers \( \Box \Box \). \( \Box \Box \) is well-formed, the exponent of \( r \) is the fifth element in \( \Box \Box \). \( \Box \Box \) is a re-randomization of \( \sigma_{\text{acc}}^{\Box \Box} \) which is a signature generated by aggregating \( \alpha \) different valid signature shares of maintainers on \( \Box \Box \). \( \Box \Box \) is generated considering \( \alpha \) and \( v \) in \( \psi_1 \). Hence, \( \Box \Box \) is \( \Box \Box \) and \( \Box \Box \) is \( \Box \Box \) and \( \Box \Box \) is \( \Box \Box \). store for \( \Box \Box \). Additionally, \( \Box \Box \) is \( \Box \Box \) for \( \Box \Box \). Additionally, \( \Box \Box \) is \( \Box \Box \).

The sender \( U \) (resp. receiver \( U \)) sends the sender anonymous channel \( F_{\text{Ch}} \) with input (Send, sid, M, \( \Box \Box \)) (resp. (Send, sid, M, \( \Box \Box \))) for \( j = 1, \ldots, D \). Specifically, \( U \) sends an abort request to the \( \Box \Box \) which is the form \( \Box \Box \) (for which \( \Box \Box \)) is a refreshed account of the user and \( \Box \Box \) is a blinded version of it. T = \( \Box \Box \) is the most recent tag used in the user’s most recent transaction. \( \rho \) is a NIZK proof of knowledge. Specifically, the user U acts as follows:

1. Computes \( \Box \Box \) and \( \Box \Box \). \( \Box \Box \) is computed for \( U \)’s refreshed account \( \Box \Box \) using PrepareBlindSign algorithm and \( \Box \Box \) is computed for \( U \)’s previous account \( \Box \Box \) (for which it has consolidated signature \( \Box \Box \)) using the ProveSig algorithm of the TBS scheme.
2. Calls \(\mathcal{T}_{\text{NIZK}}\) with input (Proof, sid, x, w), and obtains (Proof, sid, \(\pi\)) from it in which \(\pi\) is a NIZK proof for the statement \(x = (\text{acc}^\text{new}\_B, \text{acc}^\text{Round}_B, T)\). We denote the randomness used to create \(\text{acc}^\text{new}_B\) and \(\text{acc}^\text{Round}_B\) by \(\mathsf{r}_{\text{abr}}\). The witness of \(\pi\) is \(w = (\text{acc}, \mathsf{r}_{\text{abr}})\) for the following relation \(R(x, w)\) (for formal definition of the relation and the associated Sigma protocol see the full version of the paper [26]):

(1) \(T\) is well-formed, the exponent of \(g\) is the fifth element in \(\text{acc}^i\).

(2) \(\sigma^\text{Round}_B\) is \(\text{re}-\text{randomization of} \sigma^\text{old}_B\) which is a signature generated by aggregating a different valid signature shares of maintainers on acc. \(\text{acc}^\text{new}_B\) is generated considering acc. Hence, \(B' = \text{acc}^{\text{old}}_B, S' = \text{acc}^{\text{old}}_B, R' = \text{acc}^{\text{old}}_B, \mathsf{sk}_1^a = \mathsf{sk}_i^a, z^{a+1}_i = a^i \cdot a\) and \(\sigma' = \text{acc}\).

(3) \(U\) knows the randomness \(\mathsf{r}_{\text{abr}}\).

3. Calls the sender anonymous channel \(\mathcal{T}_{\text{Ch}}\) with input (Send, sid, \(M_i, \mathsf{AR}\)) for \(j = 1, \ldots, D\). Specifically, \(U\) calls \(\mathcal{T}_{\text{Ch}}\) with the input (Send, sid, \(M_i, \mathsf{AR}\)) who has already had a transaction saved in its ledger. This protocol parses as two sub-protocols Privacy Revocation and Tracing which are as follows.

(1) Privacy Revocation: Given a privacy-preserved payment made by a specific sender-receiver pair, the audit committee revokes the privacy of the transaction by decrypting the ciphertexts and identifying transaction participants and value of the transaction. Upon receiving a message (RvkAm, sid, \(\mathsf{t}_{id}\)) from \(Z\) the \(j\)-th maintainer \(M_j\) does the following (as shown in the picture in the full version of the paper [26]):

1. Finds the associated \((\psi_{s,a}, \psi_{r,a})\) saved in its ledger \(\mathcal{L}\) for the given \(\mathsf{t}_{id}\), and computes its decryption shares that are \(\psi_{s,1,1}^{sk_1,1}\) and \(\psi_{s,1}^{sk_{1,1}}\) for \(\psi_{s,1}\) and \(\psi_{r,1}^{sk_{1,1}}\) for \(\psi_{r,1}\), and calls \(\mathcal{T}_{\text{NIZK}}\) with input (Proof, sid, \(x_j, w_j\)), and obtains (Proof, sid, \(\pi_j\)) from it in which \(\pi_j\) is a NIZK proof for the statement \(x_j = (\psi_{s,1}^{sk_{1,1}}, \psi_{s,1}^{sk_{1,1}}, \psi_{s,1}^{sk_{1,1}}, \psi_{r,1}^{sk_{1,1}}, \psi_{r,1}^{sk_{1,1}})\). The witness of \(\pi_j\) is \(w_j = (sk_{1,1}, sk_{2,1}, w_{1,1})\) for the following relation \(R(x_j, w_j)\): \(\log_g \mathsf{pk}_1 = \log_g \psi_{s,1}^{sk_{1,1}}, \log_g \mathsf{pk}_2 = \log_g \psi_{s,1}^{sk_{1,1}}, \log_g \mathsf{pk}_3 = \log_g \psi_{r,1}^{sk_{1,1}}\). For the associated Sigma protocol see the full version of the paper [26].

2. Calls the authenticated channel \(\mathcal{T}_{\text{Ch}}\) with the input (Send, sid, \(M_i, (x_j, \pi_j)\)) for \(i = 1, \ldots, D \land j \neq i\), and considering the equations \(\mathsf{pk}_s = \psi_{s,2}/\Pi_{j \in 1} \psi_{s,1}^{sk_{1,1}}, g^\mathsf{g} = \psi_{s,3}/\Pi_{j \in 1} \psi_{s,1}^{sk_{1,1}}, \mathsf{pk}_r = \psi_{r,2}/\Pi_{j \in 1} \psi_{r,1}^{sk_{1,1}}\)
$\psi_r \cdot \prod_{1 \leq j \leq P} \psi_{r_j}^{ak_{1,j}}$ such that $|l| = \beta$ and $\lambda$ is Lagrange coefficient, upon obtaining $\beta$ valid decryption shares from $\mathcal{F}_{\text{ac}}$ (sent by other maintainers), computes $pk_r, g^\omega$, and $pk$. Validity of shares is checked by calling $\mathcal{F}_{\text{NIZK}}$.
3. Calls $\mathcal{F}_{\text{KB}}$ with (RetrieveID, sid, pk_r) and (RetrieveID, sid, pk_r) to retrieve unique identifiers of users by receiving (IDRetrieved, sid, U_s, pk_s) and (IDRetrieved, sid, U_r, pk_r) from $\mathcal{F}_{\text{KB}}$. Computes $v$ from $g^\omega$. Outputs (AnnRevoked, sid, t_d, U_s, U_r, v) (to $\mathcal{Z}$).

Note that to have an efficient zero-knowledge and signature of knowledge proofs the user sets $g^\omega$ as one of the plaintexts in $\psi$.

One of the system’s regulatory compliance is having a limit on transaction value $\mathcal{C}$, is the commitment to the same $\mathcal{E}$, having $\beta$ ledgers so that there is no possibility that maintainers do not see the computed tag $\gamma$. In this way, tracing authorities know the number of honest maintainers who have the whole ledger $L_j$ that includes $g^\omega$ as a tag $\gamma$, and proceed to step 2 with $\varepsilon - \varepsilon + 1$ and records the associated $t_d$ of computed $T$ and role.

Else, sends a message to all maintainers via calling $\mathcal{F}_{\text{ac}}$ with input (Send, sid, M_i, $(0, g^\omega)$) for $i = 1, \ldots, D \land i \neq j$ (which means it has not seen $g^\omega$ in $L_j$).

4. If it receives $2t + 1$ messages of the form (Received, sid, M_i, $(0, g^\omega)$) in which $g^\omega$ from $\mathcal{F}_{\text{ac}}$, outputs transaction identifiers and corresponding roles (Traced, sid, $(t_{j, i}^{\text{ac}}, \text{role}_{j, i}^{\text{ac}})$) to $\mathcal{Z}$, and aborts. Else, waits for at least $t + 1$ messages of the form (Received, sid, M_i, $(\gamma, \text{pid}_j)$) in which $\gamma = g^\omega^x$ from $\mathcal{F}_{\text{ac}}$ and proceeds from step 2.

For currency issuance transaction $t_d$ only contains tracing tag of receiver and for payment transaction it contains tracing tags of both sender and receiver. Based on the computed tracing tags each maintainer knows that the traced user was sender or receiver of the transaction for which $t_d$ is retrieved (tag of the sender appears first in $t_d$). Hence, $M$ output (Traced, sid, $(t_{j, i}^{\text{ac}}, \text{role}_{j, i}^{\text{ac}})$) (to $\mathcal{Z}$) such that role can be sender or receiver. Note that given the $\{t_{j, i}^{\text{ac}}\}$ values, the counterparts of the suspicious user can be revealed using the Privacy Revocation protocol described above. To make tracing efficient, at User Registration protocol each user proves that $x$ starts from 1 and then increments by one for each transaction.

4 PEReDi SECURITY AND PERFORMANCE

Our main theorem is given below.

Theorem 1. Assuming that Pedersen commitments are perfectly hiding, Pointcheval-Sanders signatures are EUF-CMA secure in the random oracle model, ElGamal encryption is IND-CPA secure, and the $d$-strong Diffie-Hellman problem is hard, there exist two polynomials $p_e$ and $p_a$ such that no PPT environment $\mathcal{Z}$ can distinguish the real-world execution EXEC$_{\text{PEReDi},\mathcal{A},\mathcal{Z}}$ from the ideal-world execution EXEC$_{\text{PEReDi},\mathcal{A},\mathcal{Z}}$ with advantage better than Adv$_{\mathcal{A}}$.

ACKNOWLEDGMENTS

This work was supported by Input Output (iohk.io) through their funding of the Edinburgh Blockchain Technology Lab.
REFERENCES


