Poster: Rayls: A Novel Design for CBDCs

Mario Yaksetig and Mahdi Nejadgholi and Stephen Yang and Nicholas Zanutim

Parfin

Rayls

Introduction

- Central Bank Digital Currencies (CBDCs) are currencies issued from a central bank with a direct liability to such bank. The goal of a CBDC is to improve on the inefficiencies of money.
- Getting a CBDC right is a hard challenge. For example, the most popular commercial bank in India has 500 million users.
- Approaches that rely on a single monolithic blockchain are doomed to fail. No single blockchain can support the throughput for that bank, let alone for a big country.
- The intuition for layer-2 scaling is partially correct, yet is lacking privacy.

Why not a token on a public blockchain?

A CBDC cannot simply be instantiated on a public blockchain.

Why:

- **Privacy** The entire transaction data of a nation should not be available for everyone to see.
- **Scalability** - No blockchain can process the transactions to fulfil the needs of a nation.

L2 scaling

Each bank can be a rollup (or validium). This, however, reveals the balance sheets of all the banks in real-time, which is not acceptable.

Improvements vs Previous Work

- **Zether** [1] provides no auditability of transactions without revealing the spending key.
- **zkLedger** [3] provides an off-chain solution and relies on zero-knowledge proofs with linear verification time and linear proof size.

Our contributions

We introduce a design that provides:

- **Privacy** - Balances of commercial banks are private. Transactions between commercial banks are also private.
- **High performance** - Each PL processes the transactions from their users. This allows for high throughput and modularity.
- **Decentralized Verifier** - We use ZK-SNARKs (i.e., Groth16 [2]) and a smart contract verifier
- **Auditability** - A central bank can audit the transactions that take place.

System Architecture

- **Users (U)** - Clients of each financial institution (i.e., PL).
- **Privacy Ledger (PL)** - Single node blockchain with a private state.
- **Relayer (R)** - Connects PL to Commit Chain. Each relayer has a wallet address on the CC.
- **Commit Chain (CC)** - Public chain to which all the privacy ledgers connect to.
- **Admin (A)** - Issuer of the CBDC on the public chain.

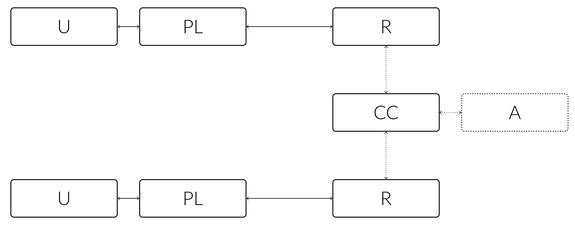
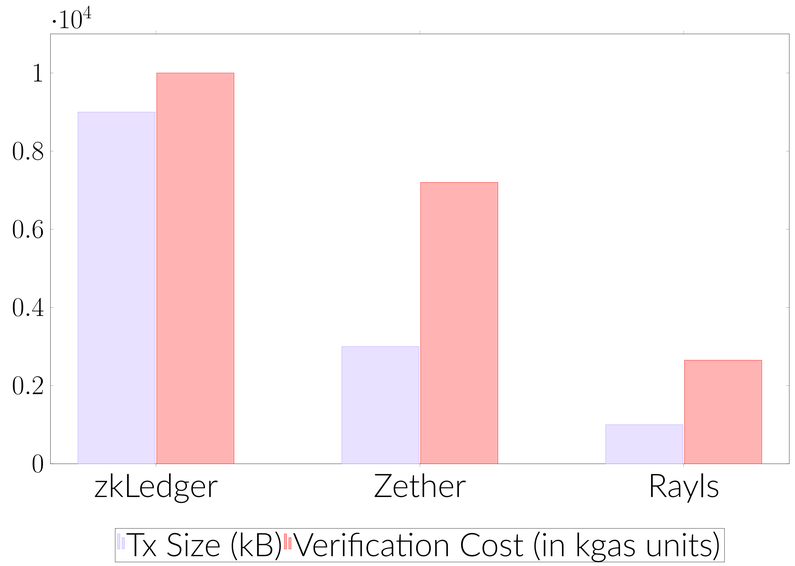


Figure 1. Rayls Simplified Architecture with two banks

Results



Protocol Overview

- **Setup** - PLs generate Diffie-Hellman key pairs $(\mathsf{sk}, \mathsf{pk}_i)$.
- **Registration** - PLs register a Diffie-Hellman public key.
- **Key Agreement** - Perform a Diffie-Hellman key agreement with all the other PLs in the system.
- **Sending Transactions** - Using the shared secret, the sending PL generates a private transaction along with a private message signaling tag.
- **Receiving Transactions** - Each PL performs a private information retrieval lookup from the blocks, which are stored locally.
- **Auditing** - Optionally, the commercial banks can share view keys with the admin. This allows the central bank to see who is transacting, but no spending on behalf of any commercial bank.

Private Transactions

- Each entity in the system has a public key pk_i and a private balance as a (Pedersen) commitment $\mathcal{C}_i$.
- To make a transaction, the sender submits a list of commitments along with a zero-knowledge proof π and a nullifier nf. These commitments represent either a debit or a credit. The nullifier ensures that the same sender is not able to double spend.
- The random factors used in the Pedersen commitments originate from the shared secrets and ensure that each party can quickly check if there are funds for them.

	$(\mathsf{pk}_1, \mathcal{C}_1)$	$(\mathsf{pk}_2, \mathcal{C}_2)$	$(\mathsf{pk}_3, \mathcal{C}_3)$	
tx	$\mathcal{C}_1^*$	$\mathcal{C}_2^*$	$\mathcal{C}_3^*$	(π, nf)
tx_1	$-vG-(r_2+r_3)H$	$vG+r_2H$	$0G+r_3H$	(π_1, nf_1)
⋮	⋮	⋮	⋮	⋮
tx_n	$100G+r_1'H$	$-200G-(r_1'+r_3')H$	$100G+r_3'H$	(π_n, nf_n)

Table 1. Overview of a private transaction. The commit chain acts as a (decentralized) verifier.

References

[1] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020 , Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers*, page 423–443, Berlin, Heidelberg, 2020. Springer-Verlag.

[2] Jens Groth. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Paper 2016/260, 2016.

[3] Neha Narula, Willy Vasquez, and Madars Virza. zkledger: privacy-preserving auditing for distributed ledgers. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, page 65–80, USA, 2018. USENIX Association.

Poster: Rayls: A Novel Design for CBDCs

Mario Yaksetig
Parfin
George Town, Cayman Islands
mario.yaksetig@parfin.io

Mahdi Nejadgholi
Parfin
Montreal, Canada
mahdi.nejadgholi@parfin.io

Stephen Yang
Parfin
São Paulo, Brazil
stephen.yang@parfin.io

Nicholas Zanutim
Parfin
São Paulo, Brazil
nicholas.zanutim@parfin.io

***Abstract*—We introduce a new CBDC design that—similarly to rollups—leverages the use of separate (private) ledgers to allow for higher throughput. In this case, commercial banks can each run their own ledger. We also introduce a new protocol that allows for efficient anonymous transactions between banks. Therefore, the sender, receiver(s), and amount of the transactions are hidden within a variable anonymity set. Our protocol supports 'double' batching. Since each transaction consists of a set of commitments and a zero knowledge proof, each transaction can pay more than 1 bank at once and each of these individual commitments can contain an aggregated transfers from multiple users. For example, a bank can transfer $1M to a different bank and that amount is actually a sum of multiple users making transfers to that bank.**

***Index Terms*—CBDCs, Privacy, Blockchain**

I. Introduction

A blockchain is an immutable distributed chain of blocks, each containing a new state update. Blockchains can be permissionless or permissioned.

Presently, most popular blockchains are permissionless and allow for any entity to join the system. This, however, implies that all the information contained in these blocks is made public for all network participants. Financial institutions, however, hold and regularly process sensitive information. As a result, it is not in the interest of a financial institution to be part of a public network and reveal all their sensitive data. Therefore, these institutions need a solution that allows them to preserve such information private, while allowing for a connection with existing blockchain ecosystems.

II. System Model

We now describe our architecture and adversarial model.

A. System Architecture

We describe a system that consists of a set of users, a set of privacy ledgers, and a blockchain.

Consider a privacy ledger (PL) with a secret internal state. This ledger is equivalent to a single node blockchain that is managed (and validated) by an individual operator from a financial institution. The state of this blockchain is kept private and is exclusive to the financial institution. Customers of this financial institution are directly connected to the PL that banks them. Each privacy ledger is connected to a public blockchain, which we denote as commit chain (CC). Each user has a wallet address on their corresponding privacy ledger. Each privacy ledger has a wallet address and a balance on this public blockchain.

We also assume an admin role present in the commit chain. For example, a central bank issues a private token and distributes the balance to each PL accordingly. This is omitted from the diagram below.

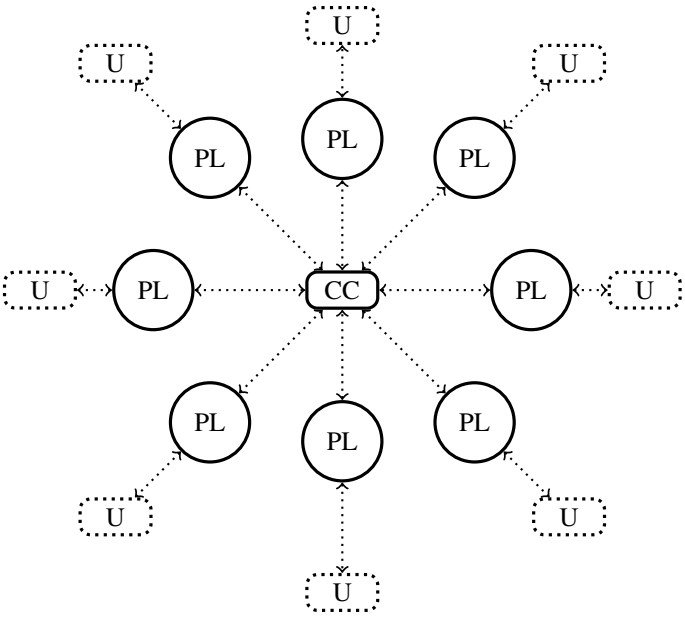


Fig. 1. Simplified System Architecture. We assume a commit chain (CC) and a set of privacy ledgers (PLs). Each privacy ledger is controlled by a commercial bank and can offer high performance to its end users. To perform transactions between PLs, each bank uses a novel privacy protocol.

B. Adversarial Model

We assume an honest admin (e.g., Central Bank), and a set of malicious PLs. The goal of each is to try to perform a double spend or spend more money than their balance sheet.

The adversary is motivated to break the privacy of the transactions in the system. Moreover, we assume that the state of the commit chain is public to everyone. Therefore, no sensitive information should be posted on the blockchain.

Even though it may be reasonable to assume that the PLs are expected to act honestly and rationally due to the legal repercussions attached to conducting fraud as a financial institution, we assume a stronger adversarial model to capture, for example, settings where a financial institution may about

to implode financially and may try to perform malicious mints and/or malicious transactions.

III. Protocol Description

We extend on the previously exposed architecture and add an additional role: the relayer. As a result, between each privacy ledger and the public blockchain, there is a relayer that performs read/write operations involving the privacy ledger and the public blockchain. The separation, however, allow for further potential decentralization of different infrastructure components and for a clearer analysis of the system.

We assume that all operators must register on the blockchain to then be able to start transacting.

A. Setup Phase

First, node operators initiate their own private ledger and generate an elliptic-curve key pair.

Second, the relayer initiates their own instance that acts as a listener for events that take place in the private ledger. The relayer also generates a post-quantum key pair to be used for non-interactive key exchanges (NIKE) with other relayers in the network. We note that this step is extremely sensitive as the wrong choice of post-quantum primitive can result in catastrophic failure [1]. We also note that the recent results from Chen [2] may lead to a re-evaluation of these algorithms.

B. Joining the network

To join the network, the commit chain enforces their own set of acceptance rules. This is orthogonal to our design. For example, agreements involving financial institutions, such as banks, may involve the signing of legal contracts. This set of rules is very flexible and depends on the application. Therefore, this is outside the scope of our design. The key part is that privacy ledgers are able to register a public-key pair to be used as a wallet address and for different Diffie-Hellman key exchanges with other privacy ledgers.

C. Transacting in the network

Private ledgers can perform two types of transactions: internal and external.

Internal transactions are transactions signed by the wallets of the users that hold accounts in the private ledger. These can be seen as traditional day to day payments. These transactions are private as the private ledger operator never exposes this information to any external entities.

External transactions represent moving funds from one privacy ledger to another ledger. Essentially, a bank wants to send funds to a different bank. We assume a private token issued on the commit chain where each balance is shielded and turned into a Pedersen commitment. A bank can aggregate multiple transactions from users to a specific bank. We explain this type of transaction in the next section.

IV. Private Transactions

Each privacy ledger in the system has a public key identifier pk_i and a corresponding private balance as a (Pedersen) commitment $\mathcal{C}_i$. To make a transaction, the sender submits a list of commitments along with a zero-knowledge proof π and a nullifier nf. These commitments represent either a debit or a credit. The nullifier ensures that the same sender is not able to double spend. This generic transaction type is described in tx, the first transaction row.

In tx_1, we denote entity 1 sending an arbitrary balance v to entity 2 and a 0 balance to entity 3. We note that these commitments are indistinguishable from the perspective of an outsider (i.e., anyone not sending/receiving funds). Moreover, we highlight that the random factors used in the Pedersen commitments originate from the Diffie-Hellman key agreements and are used in the Pedersen commitments to ensure that each party can quickly check if there are funds for them by removing the corresponding random factor. The used r values are different for every new block.

In tx_n, we show a concrete transaction of 200 from entity 2 to both entity 1 and 3 to highlight that the system supports transactions to multiple entities simultaneously. For simplicity, we show below only three entities in this example. This approach can support an arbitrary number of involved parties.

	$(\mathsf{pk}_1, \mathcal{C}_1)$	$(\mathsf{pk}_2, \mathcal{C}_2)$	$(\mathsf{pk}_3, \mathcal{C}_3)$	
tx	$\mathcal{C}_1^*$	$\mathcal{C}_2^*$	$\mathcal{C}_3^*$	(π, nf)
tx_1	$-vG - (r_2 + r_3)H$	$vG + r_2H$	$0G + r_3H$	(π_1, nf_1)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
tx_n	$100G + r'_1H$	$-200G - (r'_1 + r'_3)H$	$100G + r'_3H$	(π_n, nf_n)

TABLE I
Overview of a Rayls private transaction.

V. Implementation

We implemented an initial ZK implementation using Zokrates. Elliptic curve operations use the Baby Jubjub curve. Creating a transaction with an anonymity set of 6 using an AWS c5.2xlarge takes approximately 2 seconds. We expect this number to go down significantly. We also note that this is an affordable instance that costs less than $\$0.5$ per hour.

VI. Conclusion

Using zkLedger [3] as inspiration, we introduce a new design that paves the way for financial institutions to move their existing infrastructure to blockchain rails and connect to existing blockchain ecosystems and conduct efficient private transactions without leaking their sensitive data.

References

[1] W. Castryck and T. Decru, "An efficient key recovery attack on sidh," Cryptology ePrint Archive, Paper 2022/975, 2022. [Online]. Available: https://eprint.iacr.org/2022/975

[2] Y. Chen, "Quantum algorithms for lattice problems," Cryptology ePrint Archive, Paper 2024/555, 2024. [Online]. Available: https://eprint.iacr.org/2024/555

[3] N. Narula, W. Vasquez, and M. Virza, "zkledger: privacy-preserving auditing for distributed ledgers," in *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'18. USA: USENIX Association, 2018, p. 65–80.