香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

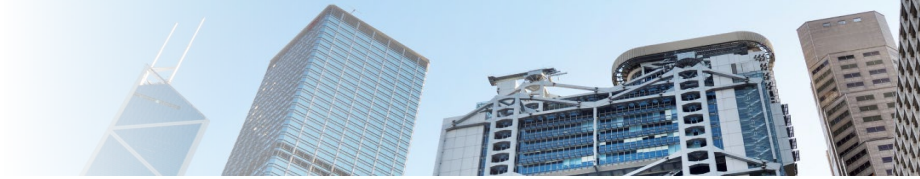
An integrated system for privacy-preserving, and auditable transactions on Hyperledger Fabric

MPhil Thesis Defence

Presented by Martin Yip
Supervised by Prof. Dimitrios Papadopoulos

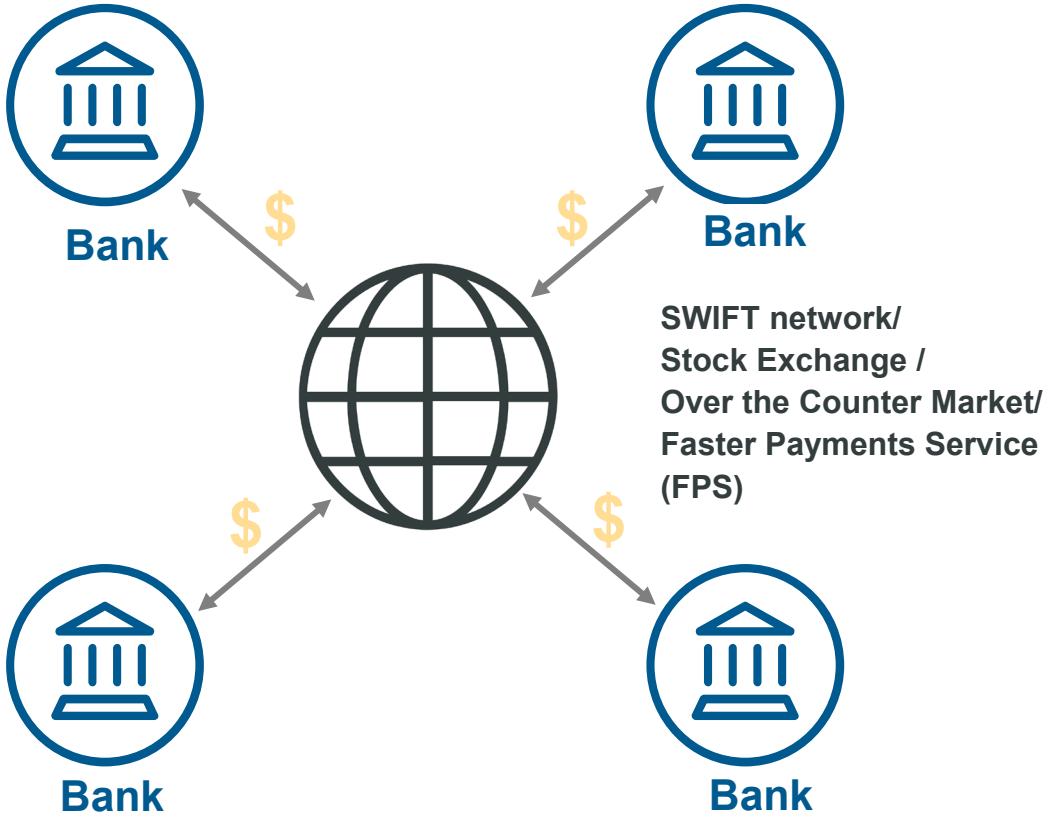
Introduction

Financial Service Industry



The financial service industry employs centralized inter-bank transaction systems [HKMA .01] to facilitates communication, transactions, and settlement among various banks through a central authority or platform.

Centralized Inter-Bank Transaction System:



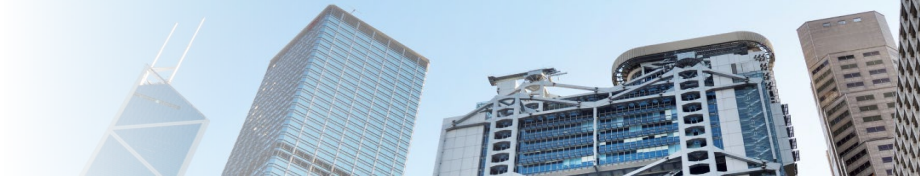
[Benefit]

- Standardization
- Facilitate global trade
- Fraud detection

[Deficiency]

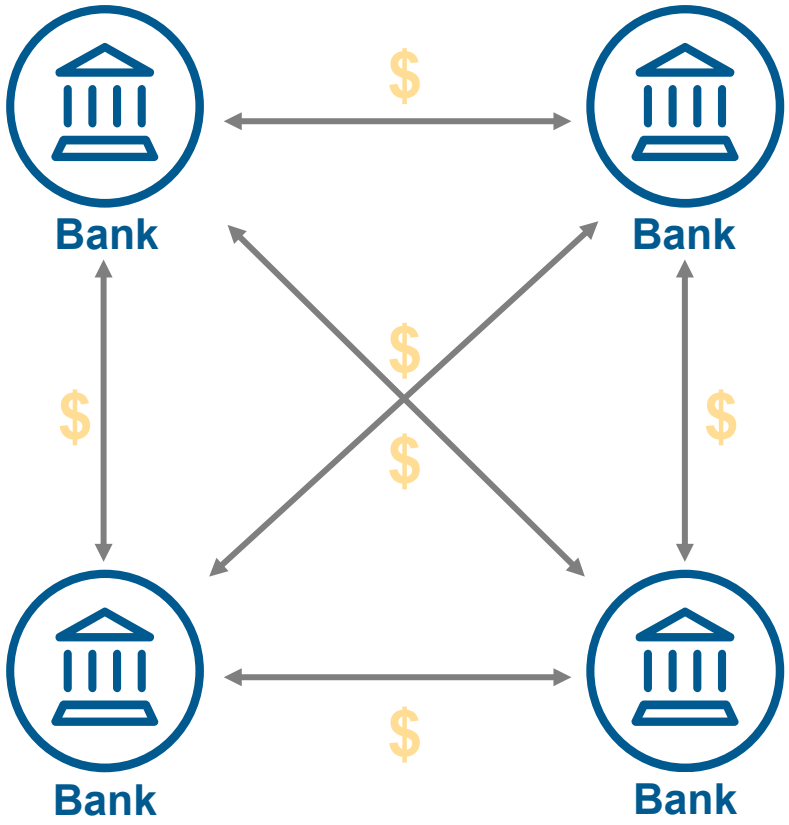
- Limited transparency
- Less flexibility
- Single point of failure

Financial Service Industry



The Hong Kong Monetary Authority (HKMA) has launched the e-HKD Pilot Programme to study the feasibility of retail central bank digital currencies as part of its "Fintech 2025" strategy [HKMA .23], including using blockchain based platform for transactions and settlement of tokenized assets.

Decentralized Inter-Bank Transaction System (Permissioned Blockchain):



Onyx (J.P. Morgan)
R3's Corda (HSBC)
Ripple
(Standard Chartered)

[Benefit]

- System resilience
- Improved transparency
- Cost management

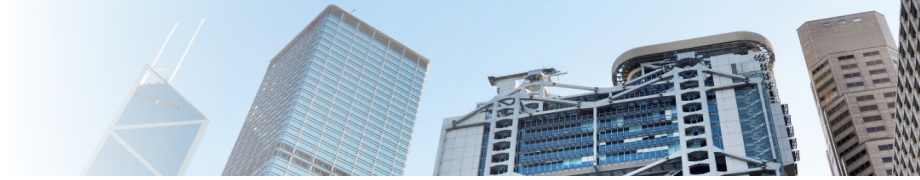
[Deficiency]

- Less scalable
- Adoption challenges
- Privacy concern

[Privacy Concern]

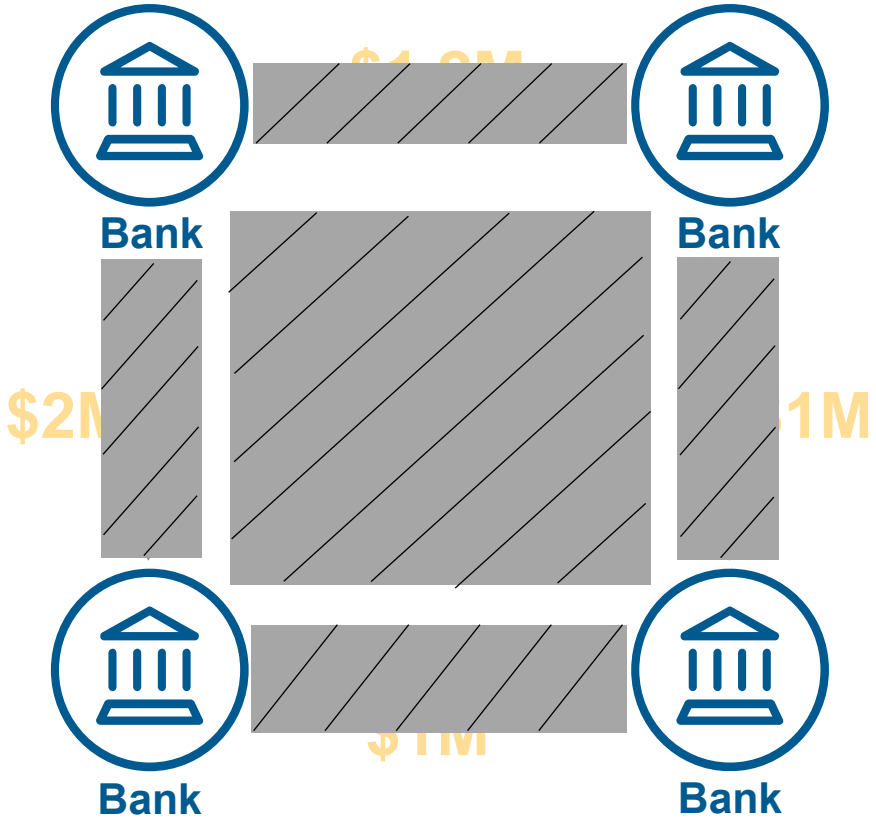
Banks concern that the transparency from collaborating will reveal sensitive trading strategy information, potentially compromising their competitive advantage and privacy.

Privacy-Preserving Transactions



Banks would use privacy-preserving transactions to hide the transaction amounts and sender/receivers' identities.

Clear transactions → **Privacy-preserving transactions**



Transfer transaction	
Spending bank	SCB
Receiving bank	Goldman Sachs
Transfer amount	$ 1,000,000

Transfer transaction	
Spending bank	
Receiving bank	
Transfer amount	

Any problem if all of the transaction information is obscured?

Auditing & Privacy

Regardless of which transaction system the banks use, financial audits are required to be conducted regularly, which cover various aspects of a bank's operations, including its balance sheet to fulfill regulatory requirements. However, it will be challenging to audit ledgers with privacy-preserving transactions.

ID	Date	Asset	From	To	Amount
23	30 Nov 2023	$	HSBC	SCB	1,000,000
24	30 Nov 2023	$	BOC	Goldman Sachs	2,000,000
25	30 Nov 2023	$	UBS	Citibank	1,000,000

[Privacy-Preserving Transactions]

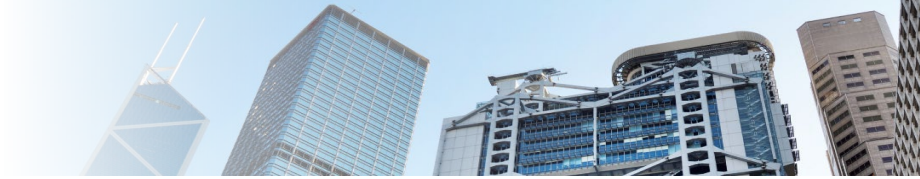
ID	Date	Asset	From	To	Amount

Auditors will examine the ledger to verify to following financial invariants:

- ✓ Consent to transfer
- ✓ Has assets to transfer
- ✓ Assets neither created nor destroyed

How can the auditor verify the transactions now?

Related Works



Privacy-preserving transactions with auditability is a significant area of research in the blockchain and cryptography domains. Here are some research papers that focus on auditable privacy-preserving transactions in blockchain:

Paper	Benefit	Deficiency
zkLedger [Narula et al. 18]	a privacy-preserving, auditable distributed ledger system, allowing for regulatory compliance and auditability	has not been implemented on a blockchain platform.
MINILEDGER [Chatzigiannis et al. 21]	a private and auditable payment system with storage costs independent of the number of transactions.	has not been implemented on a blockchain platform.
FabZK [Kang et al. 19]	a privacy-preserving framework for Hyperledger Fabric, a permissioned blockchain	does not support a rich set of auditing primitives.
smartFHE [Solomon et al. 23]	a framework to support private smart contracts using fully homomorphic encryption	does not support a rich set of auditing primitives.
Zether [B¨unz et al. 20]	a confidential payment mechanism compatible with Ethereum and other smart contract platforms.	does not support a rich set of auditing primitives.
Solidus [Cecchetti et al. 14]	a protocol for confidential transactions on public blockchains	does not support private auditing
Zerocash [Ben-Sasson et al. 14]	a protocol that extends Bitcoin enabling anonymous and non-traceable transactions	does not support private auditing

Auditing with zkLedger

zkLedger proposed a privacy-preserving ledger system to enable confidential transactions while still allowing for auditability.

Privacy-preserving Ledger

Asset	From	To	Amount
$			
$			
$			

Auditors can still examine the **Privacy-preserving ledger** to verify to following financial invariants:

- ✓ Consent to transfer
- ✓ Has assets to transfer
- ✓ Assets neither created nor destroyed

[Motivation]

zkLedger currently only exists as an abstract library without a working prototype fully implemented on a blockchain platform.

The practicability of the implementation of zkLedger on a blockchain platform with consensus mechanism has yet to be evaluated.

Thesis Summary

This thesis has presented a comprehensive study on the design and implementation of an integrated system for privacy-preserving, and auditable decentralized transaction system on Hyperledger Fabric [Androulaki et al. 18].

Objective

i.) Permissioned blockchain for inter-bank transactions will reveal sensitive information.

ii.) zkLedger has yet to be implemented on a blockchain platform with consensus mechanism.

Solution

i.) We have implemented zkLedger on top of Hyperledger Fabric, on a set of AWS EC2 VMs.

ii.) The two approaches are proposed as trade-offs between security and efficiency:

Approach 1 - complete security properties but requires serial transaction processing.

Approach 2 - concurrent transactions with a delayed proof.

iii.) The performance of the integrated system such as the throughput and the latency of the two approaches was evaluated.

Output

i.) Performance would degrade as the number of participants increases in the network for both approaches.

ii.) The second approach allows concurrent transactions, the system would more practical with a higher throughput.

Technical Background

Hyperledger Fabric

Hyperledger Fabric [Androulaki et al. 18] is a widely adopted enterprise-grade permissioned blockchain platform, developed under the Linux Foundation's Hyperledger project. It adopts Practical Byzantine Fault Tolerance (PBFT) where the nodes in the system will act on the consensus reached by the majority.

1. Submit Transaction

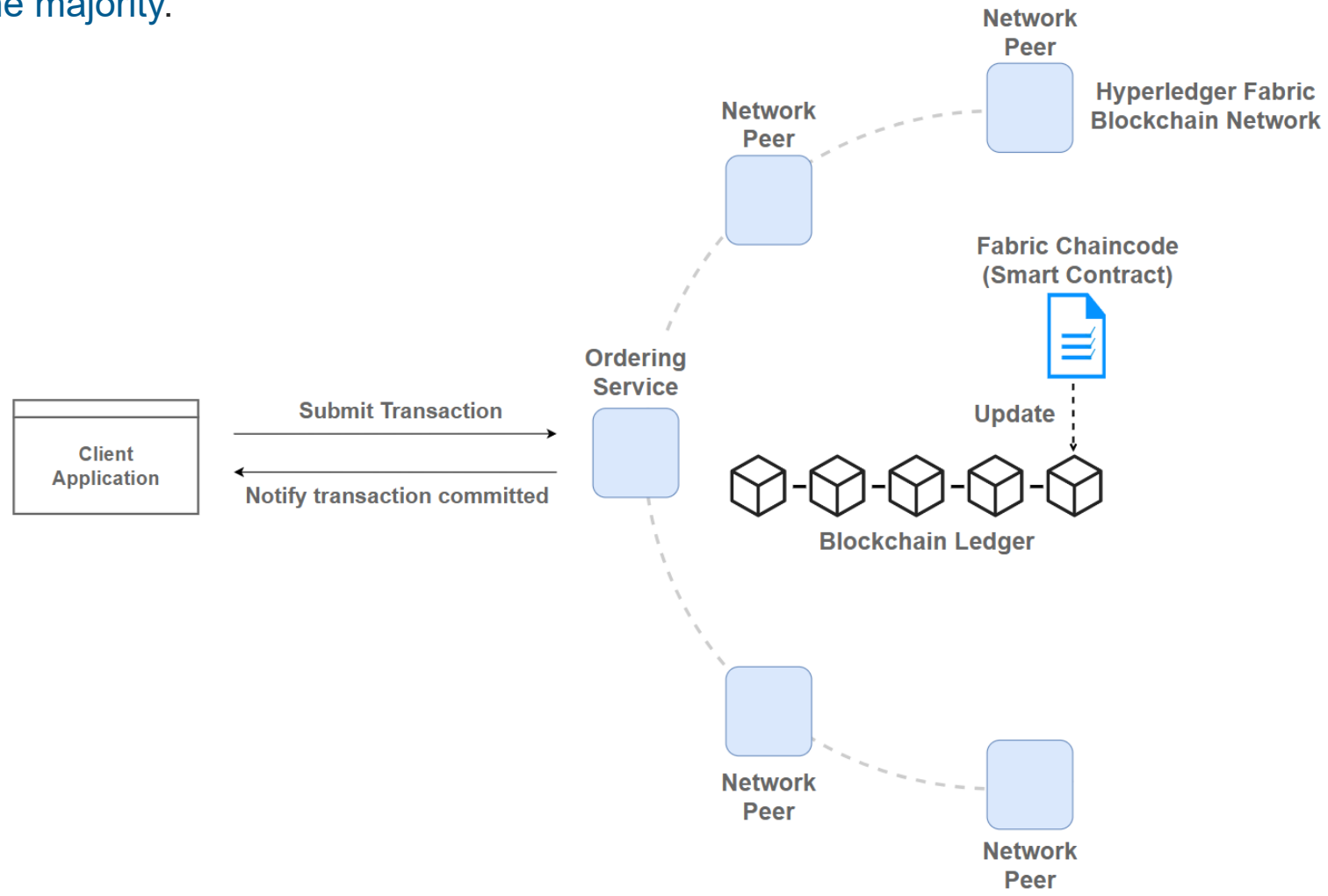
An application submits a transaction to the blockchain network.

2. Create a Block

Once the orderers have created a block, they broadcast it to the network for validation by the peers.

3. Validate Transactions

Each peer validates the transactions and appends the block to its local copy of the ledger.



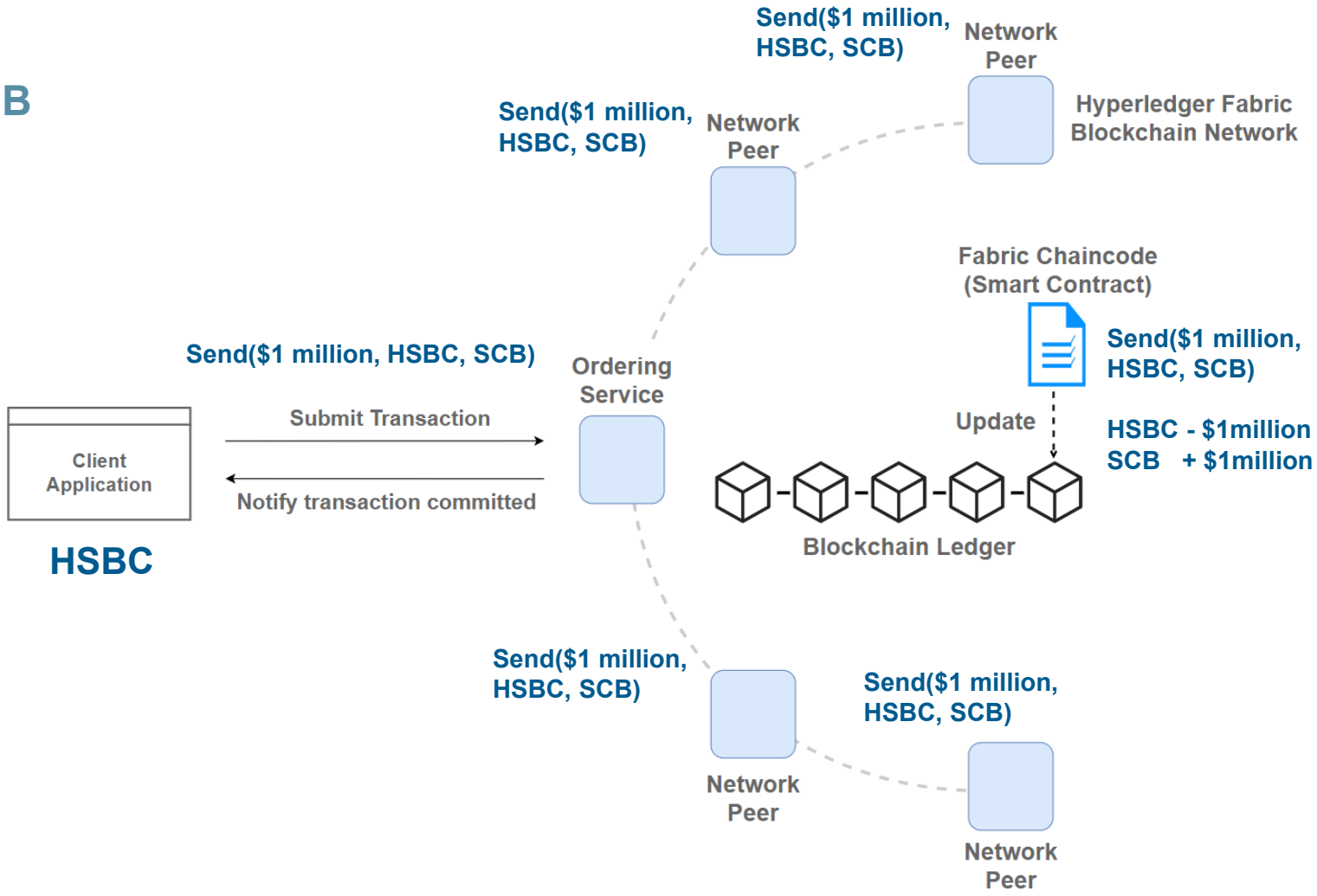
Hyperledger Fabric

However, the transaction content is clear to the participants in the network, meaning all parties have complete transparency on the network.

1. HSBC wants to send $1 million to SCB

2. HSBC submits the request (transaction) to Hyperledger Fabric

Send($1 million, HSBC, SCB)



zkLedger

zkLedger [Narula et al. 18] presents the first system for distributed ledgers to support privacy-preserving transactions with Non-Interactive Zero-Knowledge ("NIZK") proofs, and auditability.

[Commitment Schemes]

Sender creates the Pedersen commitments:

$$cm(v) = g^v h^r$$

v	value
r	randomness
g	generator
h	generator

Hiding & Binding

The **value** committed to is **hidden** from anyone who does not know the random number used in the commitment and computationally **infeasible to change** .

Homomorphically combined

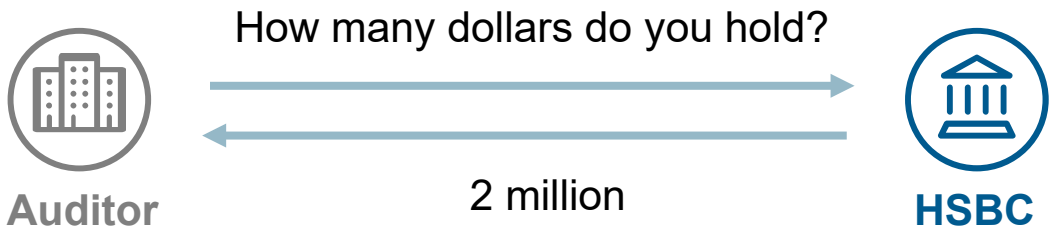
Additively homomorphic, commitments to values and randomness can create a commitment product.

$$cm_1 * cm_2 = g^{v_1} h^{r_1} * g^{v_2} h^{r_2}$$
$$cm_{1+2} = g^{v_1+v_2} * h^{r_1+r_2}$$

[Tabular Structure]

Sender creates the latest row (SCB)

Asset	SCB	HSBC	BOC
$	cm(-1M)	cm(1M)	cm(0)
$	cm(2M)	cm(0)	cm(-2M)
$	cm(-1M)	cm(1M)	cm(0)



Open cm(1M) x cm (0) x cm (1M) to 2 million

zkLedger

zkLedger enables public verifiability where the general public can mathematically verify the integrity of each transaction recorded in the ledger by using Non-Interactive Zero-Knowledge ("NIZK") proofs. NIZK proofs provide Verifiability without requiring interaction between the prover and verifier.

[Non-Interactive Zero-Knowledge Proofs]

Zero-knowledge proof allows one to prove knowledge of a secret without revealing any information about the secret itself.

Proof of Assets π^A

This proof ensures no party could create a transaction without owning enough assets to transfer.

Proof of Balance π^B

This proof ensures that a single transaction in a row conserves the overall balance in the system, meaning there is no increase or decrease in the overall assets in the system.

Proof of Consistency π^C

This proof ensures that the spending organization provides the correct randomness used in the commitment to the receiving organization.

zkLedger

[Proof of Assets] π^A

SCB	HSBC	BOC
cm(-1M)	cm(1M)	cm(0)
cm (2M)	cm(0)	cm(-2M)
cm(-1M)	cm(1M)	cm(0)

The **sum of the assets** in the column should be **greater than 0**.

[Proof of Balance] π^B

SCB	HSBC	BOC
cm(-1M)	cm(1M)	cm(0)
cm(2M)	cm(0)	cm(-2M)
cm(-1M)	cm(1M)	cm(0)

The **sum of the commitments** of the row should return to a commitment to a value of **0**.

[Proof of Consistency] π^C

SCB	HSBC	BOC
cm(-1M)	cm(1M)	cm(0)
cm(2M)	cm(0)	cm(-20M)
cm(-1M, r_1) $Token_1$	cm(1M, r_2) $Token_2$	cm (0, r_3) $Token_3$

The **randomness** used in cm and the audit Token should be the same.

$$Token = pk^r$$

[Final Transaction Construction]

Asset		HSBC	SCB	BOC	⋯	Citibank
$		$cm(-10^6, r_1)$, $Token_1$, $\pi_1^A, \pi_1^B, \pi_1^C$	$cm(10^6, r_2)$, $Token_2$, $\pi_2^A, \pi_2^B, \pi_2^C$	$cm(0, r_3)$, $Token_3$, $\pi_3^A, \pi_3^B, \pi_3^C$	⋯ ⋯ ⋯	$cm(0, r_n)$, $Token_n$, $\pi_n^A, \pi_n^B, \pi_n^C$

Challenges

Challenge from Hyperledger Fabric

Multiversion concurrency control ("MVCC") is a concurrency control mechanism used in Hyperledger Fabric to manage concurrent access to shared data. It allows multiple transactions to access the same data simultaneously without conflicts, it makes sure no double spending or inconsistency in data will occur.

Concurrency Issue: If the transactions of the same bank are created in the same block

Initial State:

HSBC : 10 M
BOC : 10 M
SCB : 10 M

HSBC wants to send $2 million to SCB

HSBC's Client Application → Request Tx 1

SCB wants to send $1 million to BOC

SCB's Client Application → Request Tx 2

[Same Block]

Validate against initial state

HSBC : 10 M
BOC : 10 M
SCB : 10 M

✓ ✓

Tx 1 can proceed to execute

HSBC : **8** M
BOC : 10 M
SCB : **12** M

✓ X

Tx 2 cannot be executed

Challenge: cannot **read / write** on data that have changed in the **same block**.

Challenge from zkLedger

The commitments are homomorphically combined to obtain a commitment to the value of the total assets. Therefore, transactions must be created serially to ensure that the sum of the assets is correct when producing the Proof of Assets.

Concurrency Issue: If the banks are creating transactions almost simultaneously

HSBC wants to send $2 million to SCB

HSBC's Client Application → Request Tx 1

SCB wants to send $1 million to BOC

SCB's Client Application → Request Tx 2

[Committed Transactions]

SCB	HSBC	BOC
cm(2M)	cm(0)	cm(-2M)
cm(-3M)	cm(3M)	cm(0)

[Upcoming Transactions]

cm(2M)	cm(-2M)	cm(0)

✓

"Able to create the correct Proof of Assets π^A since I know all the previous transactions"

cm(-1M)	cm(0)	cm(1M)

X

"**NOT** able create the correct Proof of Assets π^A since I do **NOT** know all the previous transactions"

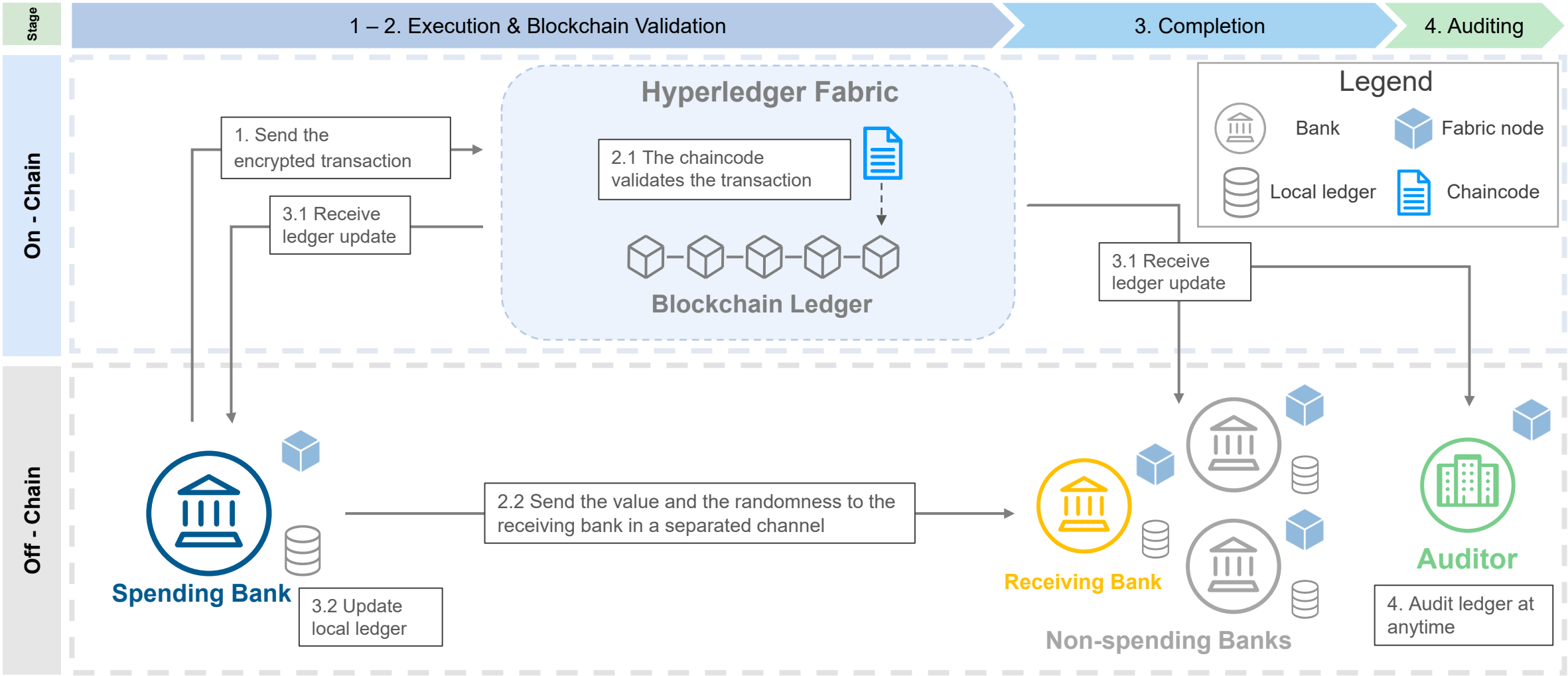
Challenge: cannot create a **correct Proof of Asset** π^A if a sender has just received a transaction.

Our Solution:

An integrated system for privacy-preserving, and auditable transactions on Hyperledger Fabric

System Design

Our design features a secure transaction system that operates on a Hyperledger Fabric blockchain network with bank clients. Each bank's machine has a node, and a client application leveraging the zkLedger cryptographic library.



Objective

zkLedger currently only exists as an **abstract library** without a working prototype of the consensus mechanism included. Additionally, together with Hyperledger Fabric, it has **concurrency issues** that need to be addressed for it to be a viable solution. Hence, we are proposing two approaches to address the challenges.

Approach 1 - Serial Transactions

This approach involves implementing zkLedger on Hyperledger Fabric without altering the security properties, resulting in a fully secure but less scalable system due to the serial transaction processing requirement.

This approach may be suitable for applications where security is of paramount importance, and transaction throughput is not a critical concern.

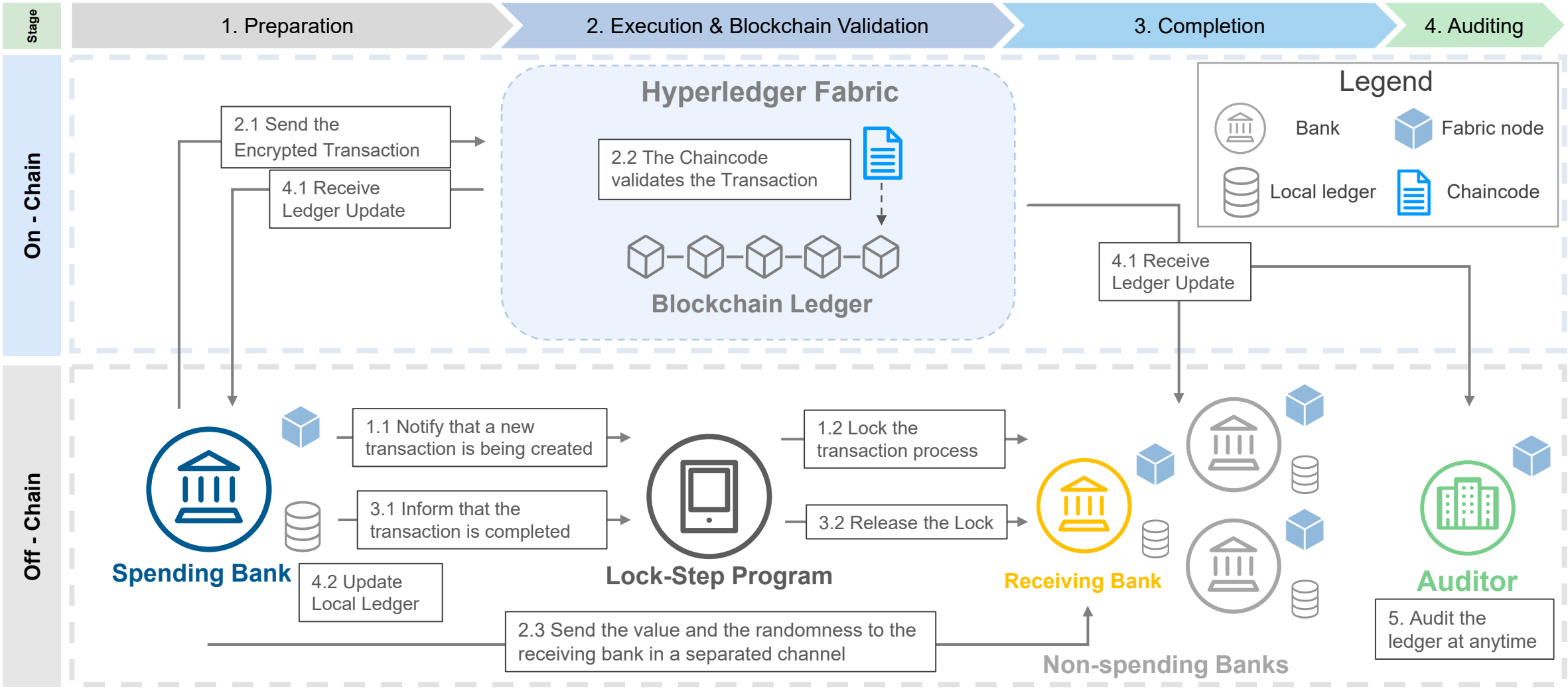
Approach 2 - Full Concurrency

This approach introduces a delayed verification of one of the zkLedger security properties, Proof of Assets, allowing participants to create transactions concurrently.

This design significantly improves the performance of the system, as participants no longer need to wait for the completion of the previous transaction before initiating new ones.

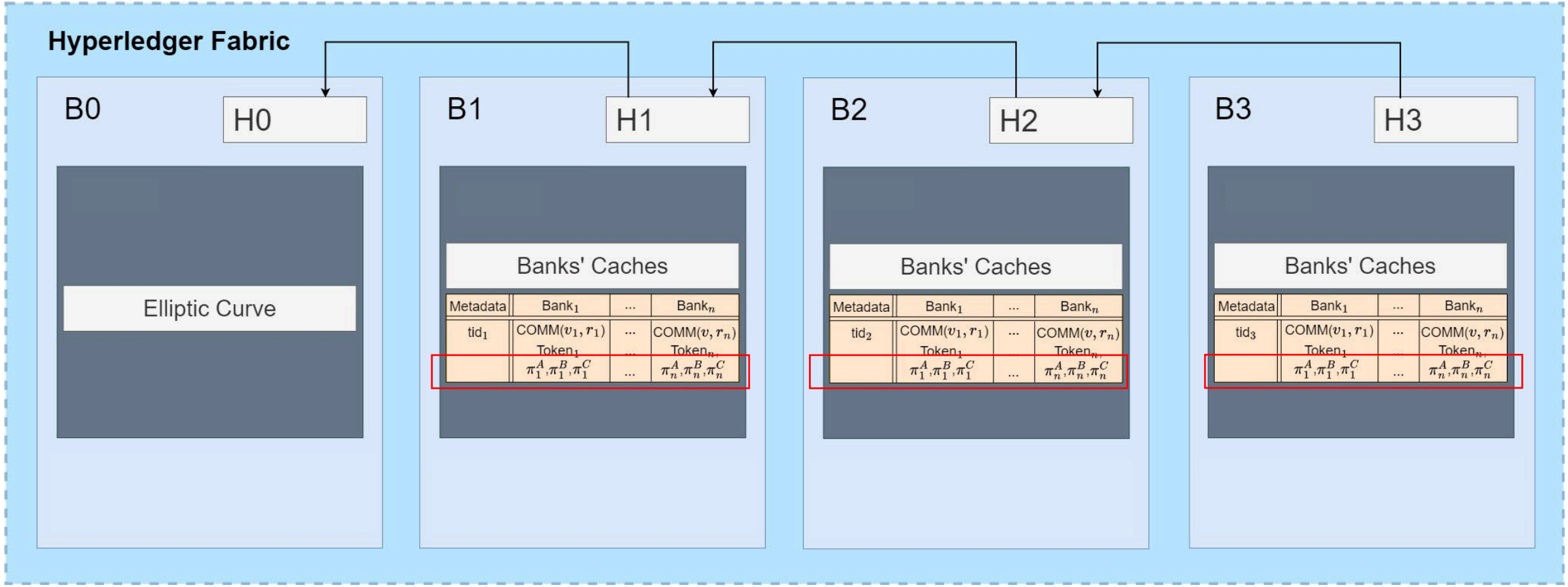
Approach 1 - Serial Transactions

This approach enables the system to attain **complete verifiability** by validating all NIZK proofs. A lock-step program is established to facilitate communication among banks, ensuring the accurate sequencing of transactions.



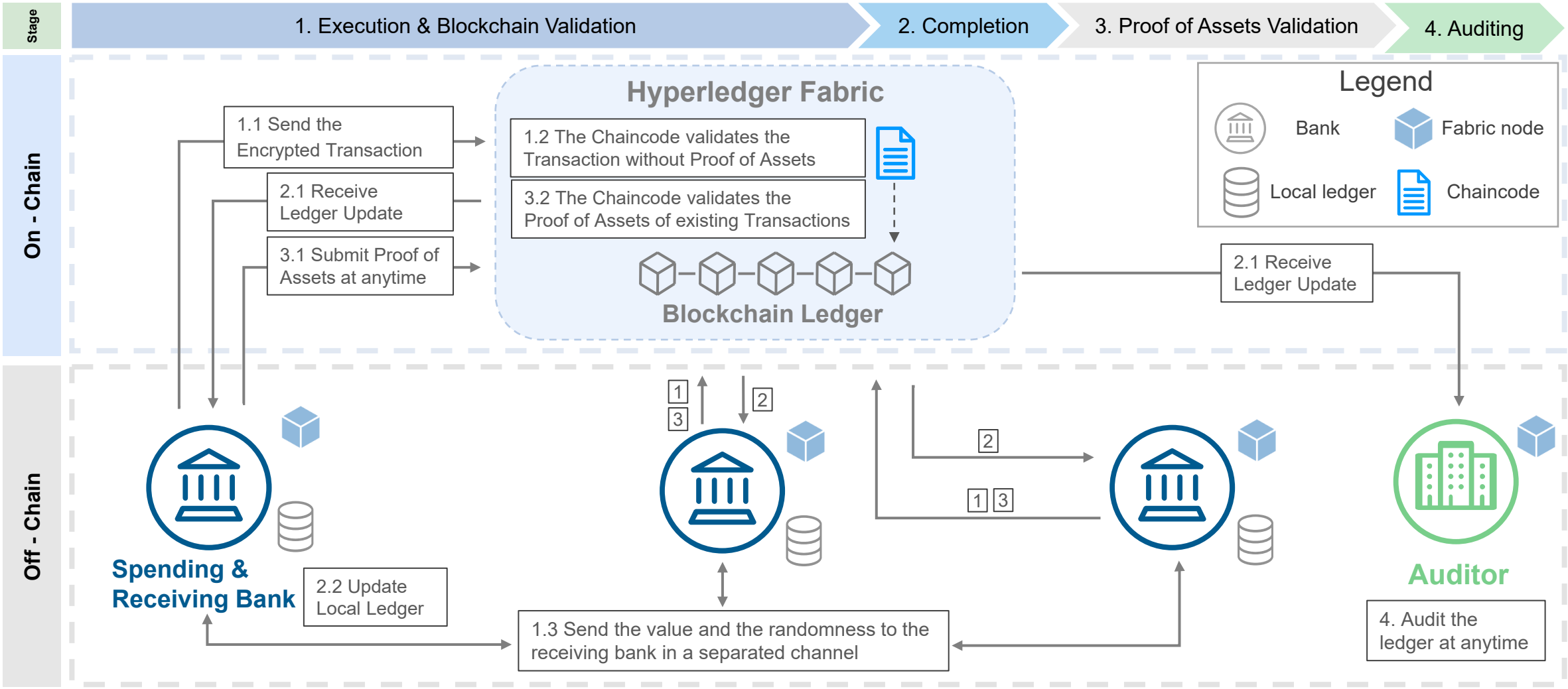
Approach 1 - Serial Transactions

A high-level block data structure for the blockchain in the Serial Transactions approach. The block data consists of a Fabric transaction, which includes the banks' commitment caches and audit token caches, and one zkLedger transaction containing all the proofs.



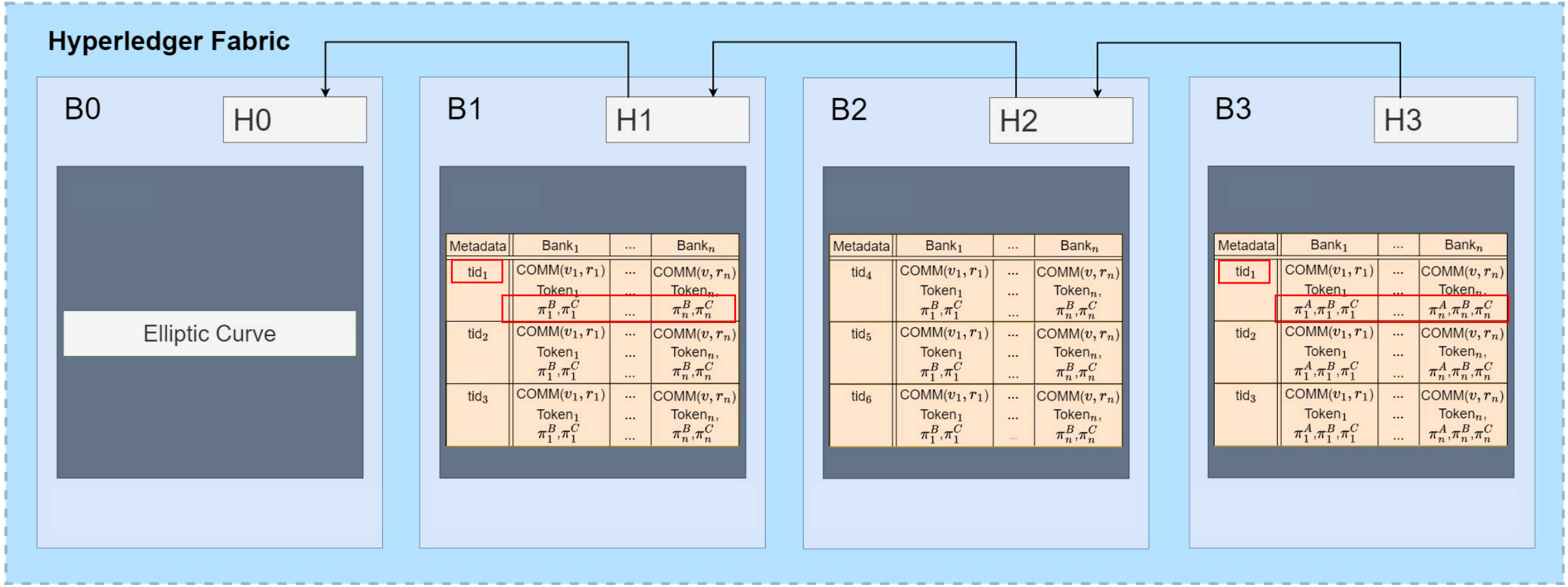
Approach 2 - Full Concurrency

This approach enables the system to attain full concurrency by delaying the on-chain validation of Proof of Assets π^A due to its serialized characteristic. The transaction structure remains the same, ensuring provability and publicly verifiability.



Approach 2 - Full Concurrency

A high-level block data structure for the blockchain in the Full Concurrency approach. The block data includes a Fabric transaction containing multiple zkLedger transactions without Proof of Assets π^A or an update of the previous zkLedger transaction with Proof of Assets π^A attached.



Approaches Comparison

The two approaches has mainly demonstrated the trade-offs below in the design and implementation of such a privacy-preserving transaction system.

Approach 1 - Serial Transactions

i. Full Verifiability
The chaincode can validate all of the proofs.

ii. Less Efficiency
Banks must create transactions serially.

iii. Less resource consumption
As the blockchain only process one transaction per block.

iv. Straightforward implementation
The cryptographic components were not modified much from the zkLedger's cryptographic library.

Approach 2 - Full Concurrency

[Comparison]

i. Delayed Verifiability
Proof of Assets, is delayed for verification.

ii. Enhanced Efficiency
Participants can create transactions concurrently.

iii. High resource consumption
The blockchain might process many transactions in one block.

iv. Complex implementation
A great amount of modification was made from the zkLedger's cryptographic library.

Evaluation

System's Performance Evaluation

In our experimental setup, we have designed and implemented a working prototype of the proposed application over Hyperledger Fabric v1.4 as the underlying blockchain platform, deployed on a set of AWS EC2 instances.

System: AWS EC2 c5.4xlarge instance type, 16 vCPUs with 3.6GHz clock speed, 32GB of RAM

OS: 64-bit Linux 5.4.0-1051-aws kernel with Ubuntu 18.04.1 as the operating system

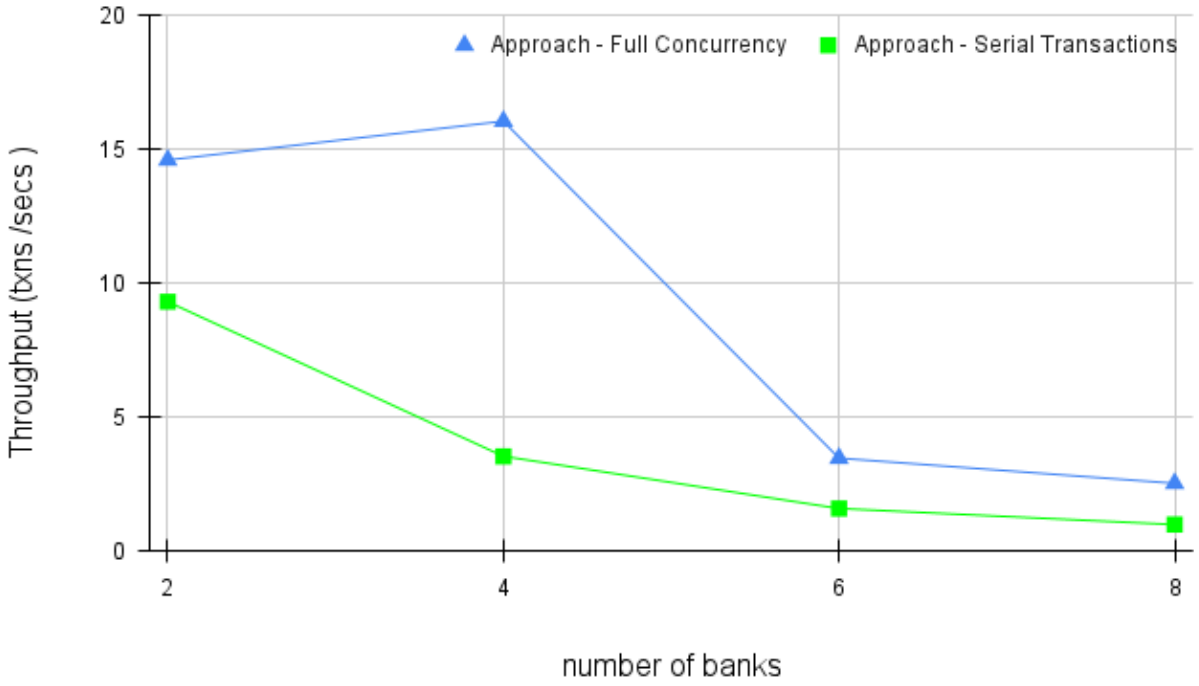
Crypto libraries: zkSigma, btcec, secp256k1

Client application and chaincode: Go version 1.9

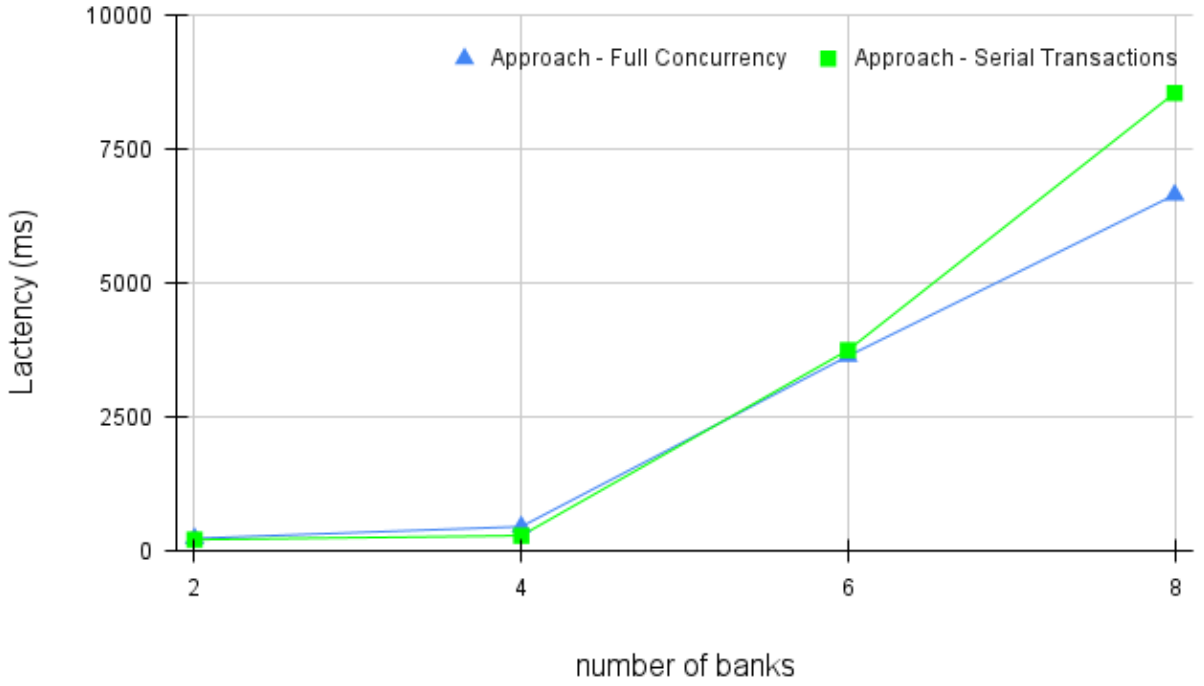
System's Performance Evaluation

The below graphs show the throughput and the end-to-end latency measured in both the Serial Transactions and Full Concurrency approaches.

In the Serial Transactions approach, the banks send one transaction at a time across participants.
In the Full Concurrency approach, the banks send N transactions simultaneously where N refers to the number of participating banks.



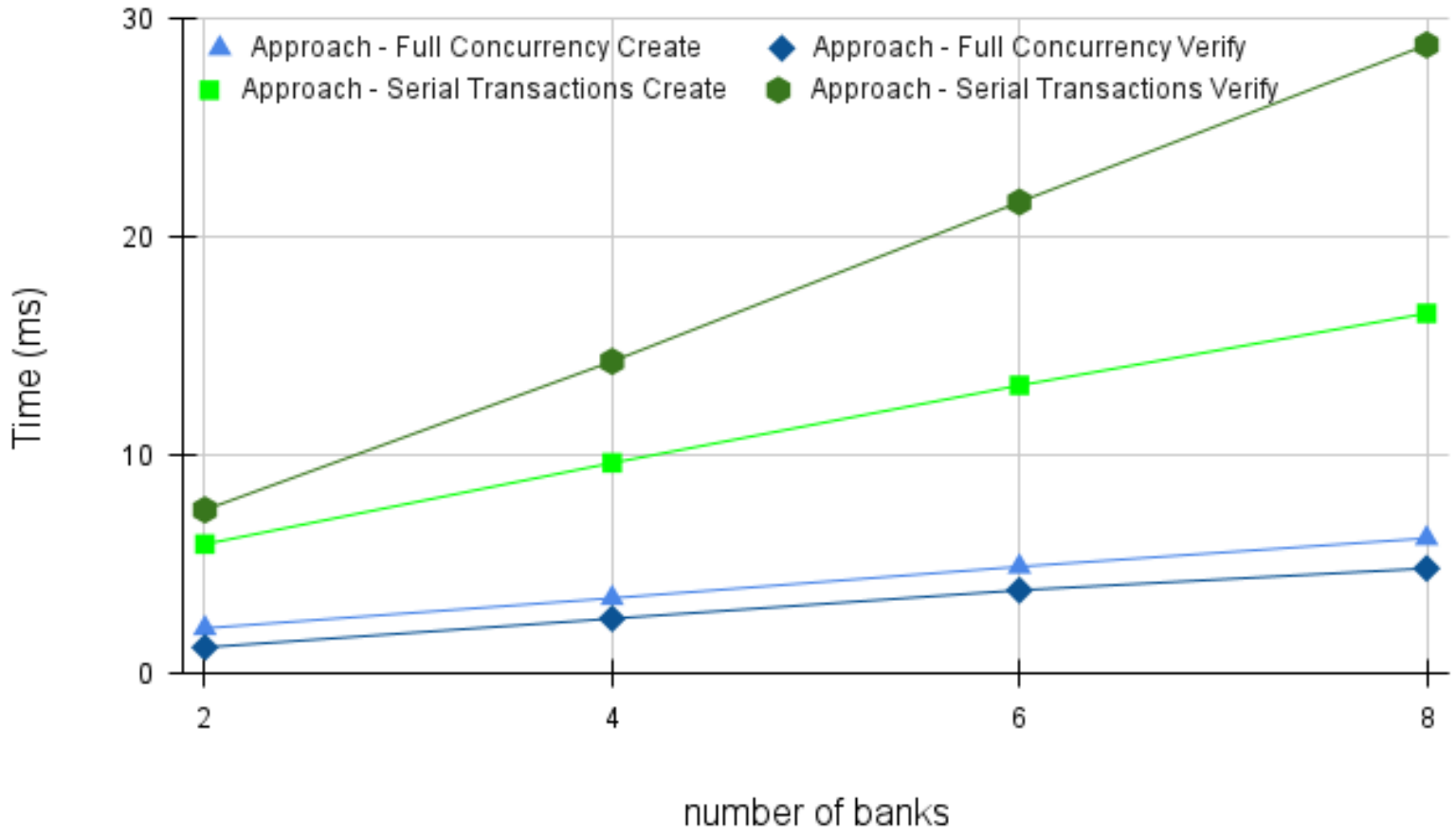
[Throughput evaluation vs number of banks]



[Latency evaluation vs number of banks]

System's Performance Evaluation

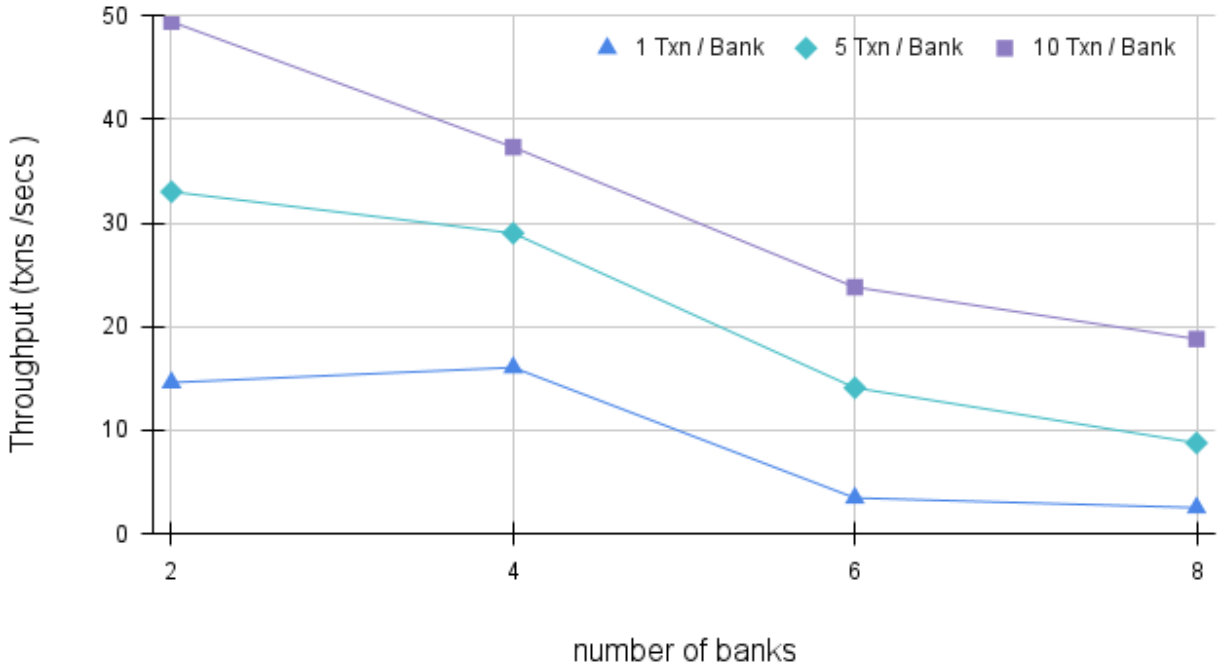
The below graph shows the time it takes for the banks to create the transactions, and the time it takes for the Hyperledger Fabric to validate the transactions in both the Serial Transactions and Full Concurrency approaches.



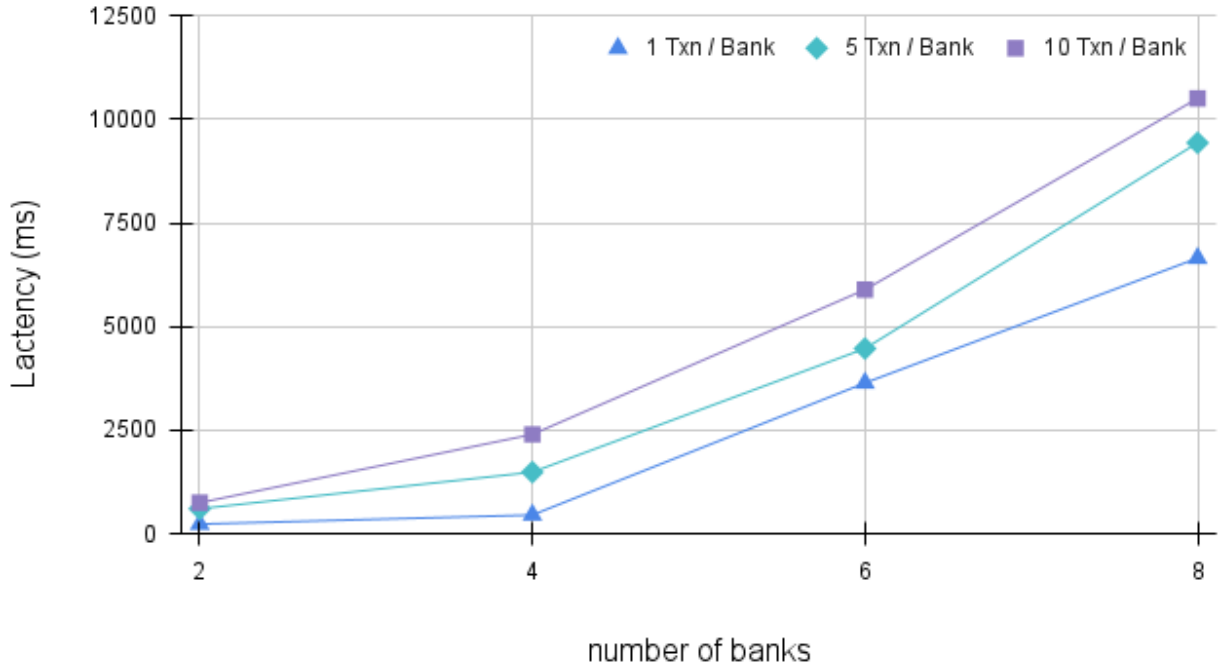
[Transaction create and verify time evaluation vs number of banks]

System's Performance Evaluation

Since banks have the ability to submit as many transactions as they wish in the Full Concurrency approach, it would be interesting to see how the system would react depending on the number of transactions each bank is transacting simultaneously.



[Throughput evaluation vs number of banks]



[Latency evaluation vs number of banks]

Conclusion

Conclusion

This thesis has presented a comprehensive study on the design and implementation of an integrated system for privacy-preserving, and auditable decentralized transaction system on Hyperledger Fabric with zkLedger.

Our result

- We have implemented zkLedger on top of Hyperledger Fabric on a set of AWS EC2 machines to evaluate the system performance as a distributed inter-bank transaction system.

- The limitation of concurrency was identified, we have proposed of two distinct approaches to address the design practicality and the identified limitations.

Core finding

- The two approaches are proposed as trade-offs between security and efficiency: the first provides complete security properties, while the second offers increased system performance.

- Although the second approach has increased system performance, in general, the system scalability has to be further improved to optimize the performance in a larger group of participants.

Future Direction

- Explore different cryptographic approaches of privacy and auditability suitable for inter-bank transaction system.

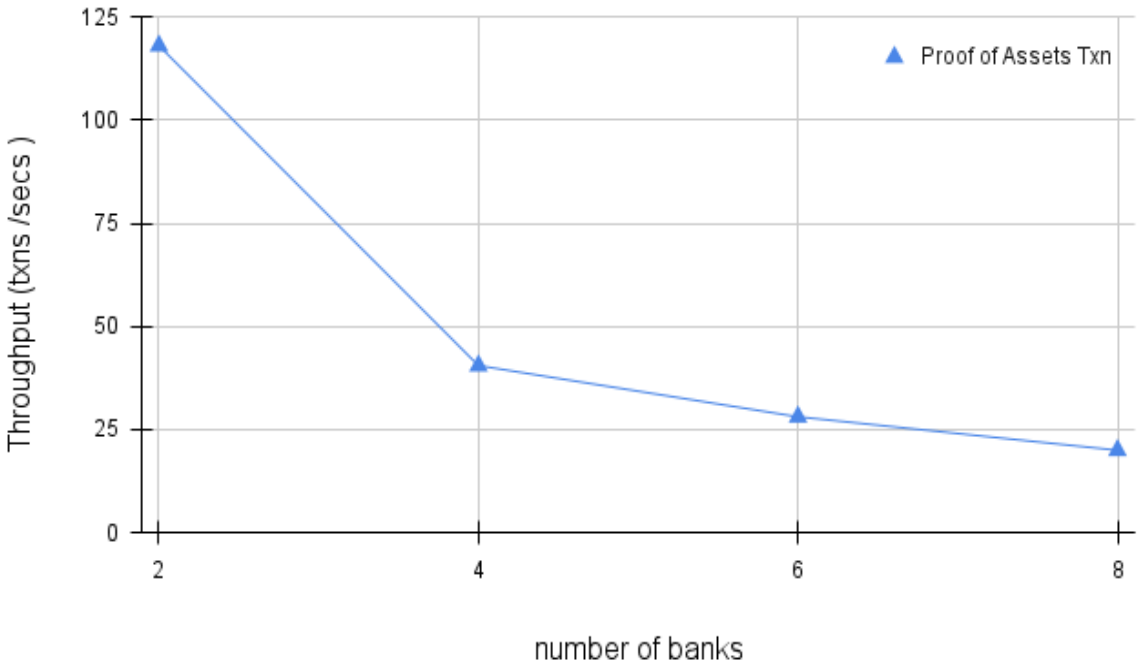
Q&A

This page intentionally left blank.

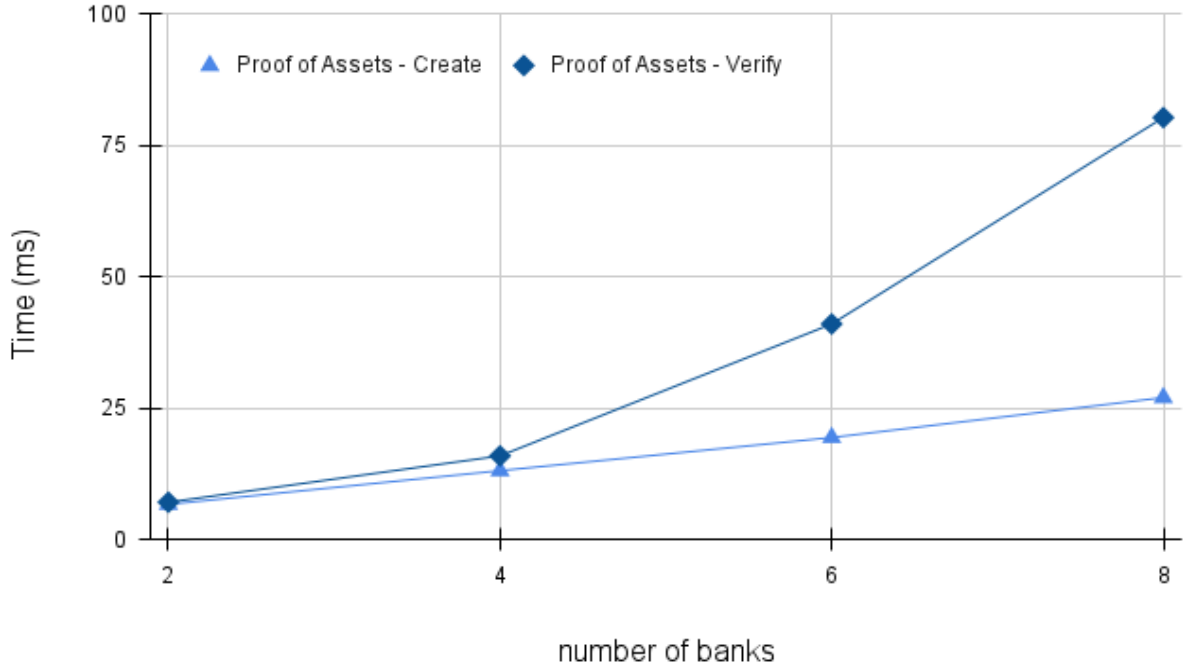
Appendix

System's Performance Evaluation

The below graphs show the throughput measured for the delayed Proof of Assets transactions of the Full Concurrency approach depending on the number of banks. Each bank is set to create and submit 10 PoA transactions as one batch.



[PoA Throughput evaluation vs number of banks]



[PoA Transaction create and verify time evaluation vs number of banks]