zkLedger Implementation on a Public Network using Hedera Consensus Service

Jaime Plata
jp@aochain.net

Abstract – Privacy-Preserving ledgers are Distributed Ledgers designed to preserve confidentiality about the nature of the transactions conducted, including the value, the actors involved, the asset balance on the ledger, as well as the sourcing graph of the assets recorded. zkLedger (1) provided the first system that enables auditing directly from the ledger without the need to reveal keys, use any third-party validator, or generate any trusted setup. FabZK (2) builds on this model as an extension to Hyperledger Fabric, keeping the privacy-preserving capabilities of zkLedger, but increasing the throughput capacity. This paper implements the baseline model of zkLedger in an open network of Hedera Hashgraph but incorporates BulletProofs (3) to calculate Range Proofs on the ledger, and the Disjunctive Zero-Knowledge Proof used for consistency in FabZK.
zkLedger provides complete and confidential auditing, where the participants cannot hide transactions. Still, the auditor only needs the value of individual operations to ascertain the integrity of the ledger. By implementing zkLedger on top of Hedera Consensus Service (4), our implementation assures a globally ordered ledger for valid transactions, including asynchronous Byzantine Fault Tolerance for the system.

1. Introduction

Incorporating Distributed Ledgers in the scope of Enterprise Applications architecture returned scenarios that demand full auditability of the transactions that circulate among a fluid number of participants, that do not want to reveal the counterparty of the transaction, or the value or history graph of the acquisition of the asset. Such scenarios are managed by using private blockchains, such as Hyperledger Fabric and R3 Corda, that involve notaries or validators to certify transactions before they are committed to the ledger. Another common alternative is to obscure transactions using cryptographic mechanisms, as implemented in ZCash or Ethereum zkSNARKs. But none of the existing public ledgers offer full data privacy and complete and efficient auditability.

zkLedger (1) presented the first architecture that supports secure transaction privacy, public verifiability, and practical auditing. The model proposed hides the volume transacted, the participants involved in the transaction, and the *transaction graph*[1]. FabZK revised the model and added some modifications to the implementation, including the replacement of Range Proofs constructed using Borromean ring signatures, by BulletProofs (2) (3), plus parallelization of operations on validation clustered by dependence on historical data. The FabZK paper reports significant improvements in throughput compared to zkLedger when the auditing feature is enabled.

FabZK's implementation on Hyperledger Fabric inherits some known limitations of a permissioned network: first, all the direct costs related to setting up the private deployment, including running Kafka and Zookeeper as Ordering service and the recommended redundancy for organization peers in a production environment. Second, Fabric requires an elaborate setup for the organizations and endorsers, that start for setting up the Certificate Authorities, to the deployment and definition of the network. The complexity in the organization setup is related to the third point: fluidity in adding or removing participants. zkLedger permits to "dynamically add and remove banks if this is done publicly" (1), but then again, whereas adding a new participant in Hyperledger requires updating the certificates, the endorsing agreements, and upgrading the Chaincode to update the new endorsing policies.

The objective of this paper is to propose an alternative architecture to implement zkLedger, incorporating the improvements conceptualized by

[1] Transaction graph is the links existing between transactions, that facilitate mapping the source of assets involved.

FabZK, but in a network that is public and permissionless[2].

2. System components

The architecture proposed builds on two key components: Hedera Consensus Service (HCS) and zkLedger's data model. HCS covers the total ordering of transactions and insertion in the append-only ledger, the assurances provided by the network about availability, finality, and data persistence. zkLedger includes the design of a register that shows transactions only to the organizations involved in a transaction, but in a model that is publicly verifiable by all participants.

2.1. Hedera Consensus Service.

The Hedera Consensus Service is one of four services provided by Hedera Hashgraph, and it is designed to work as a distributed notary service for any arbitrary message. As described in the paper: "The service receives messages, assigns them a consensus timestamp and a consensus order." Figure 1 explains the process.

This architecture provides the following capabilities to the system:

- Finality. Any message broadcasted will eventually receive a timestamp that, once assigned, cannot be changed.
- Total ordering. HCS assigns a unique identifier for all messages within a topic.
- Public availability. Any client that has access to the network can broadcast messages. Any client can subscribe to receive notifications from a particular topic.
- Governance. The topic object can control the keys that can publish messages, as well as those that have administrative rights to change the topic attributes.

The message is broadcasted through the network, and once it reaches consensus, it gets timestamped and sequenced inside a topic. The complete message is hashed to create an untampered proof of the sequencing of receipts and order assigned for a particular topic. (4)

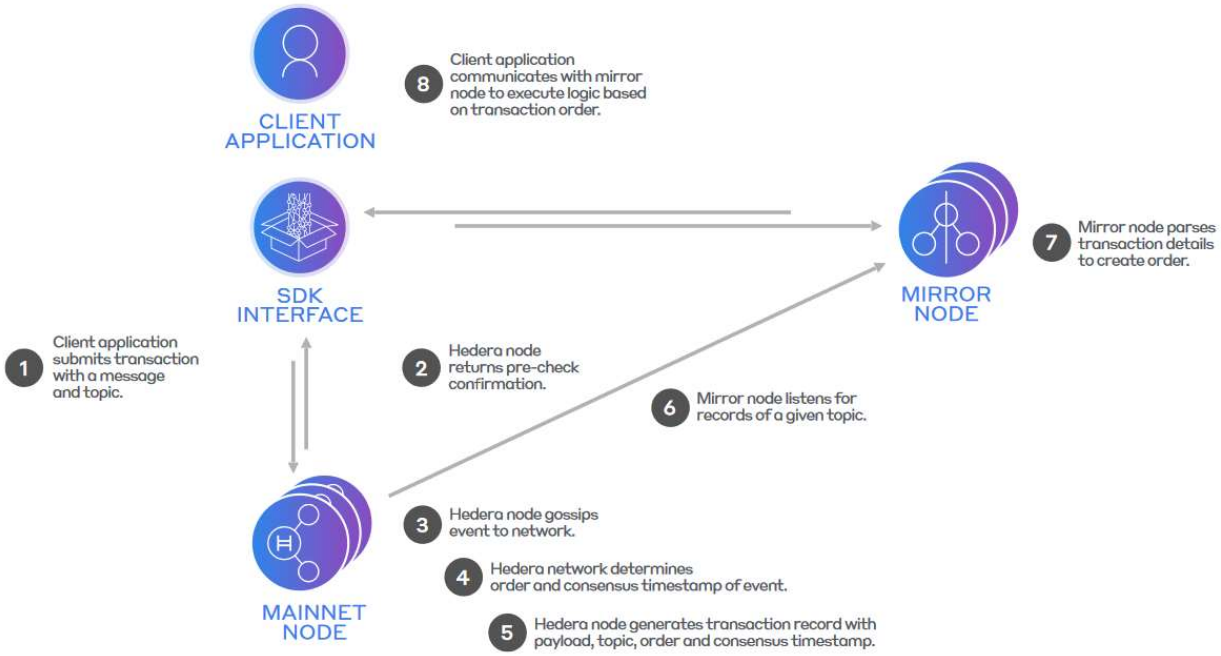

Figure 1. Transaction ordering in Hedera Consensus Service (4)

[2] Permissionless network is one where everyone can join and start transacting. Hedera does not fulfill this characteristic, since the nodes running the consensus are restricted to run by one of the council members (5).

Hedera Hashgraph separates the nodes into two categories: Mainnet nodes and Mirror nodes. Mainnet nodes are those that can take a stance in voting the validity of a message broadcasted. Mirror nodes receive the gossip messages but cannot vote in any Proof of Stake rounds[3].

Hedera Consensus Service uses three objects:

Topics: represents the unique ID for which all messages are classified together. For a particular topic, the `setTopicMemo` represents a description key without a warranty of uniqueness. The `setSubmitKey` holds the object that is authorized to submit messages to the topic. Consistent with other artifacts in Hedera, a Key can be a single key or list of keys to declare multilevel threshold access (1 of M, N of M, or M of M).

Messages: The message represents a string of bytes that is broadcasted for a particular topic. There are no restrictions on the content other than the length of the message 5kb, including the signatures. If longer messages are needed, it could be split into segments and broadcasted sequentially (5).

Mirror Node Subscriptions: The subscriptions to a Mirror Node are similar to an RSS feed. The Mirror node collects the messages produced by the Mainnet and forwards them to any client that is listening to that particular Topic. The original message includes three more attributes: the Topic Sequence, which is a consecutive number based on the order of timestamps, the consensus timestamp, and the hash of the message[4].

2.2. zkLedger.

zkLedger proposes a data structure that facilitates audits while keeping the transaction details hidden for any party that is not involved in the transaction. The model leverages previous developments to obscure the transaction detail using homomorphic commitments, particularly a Pedersen Commitment. “Pedersen Commitments are perfectly hiding (an attacker with infinite computing power cannot tell what amount has been committed to) and computationally binding (no efficient algorithm running in a practical amount of time can produce fake commitments, except with small probability).” (6)

ZkLedger proposes to generate the transactions and NIZK proofs for a discrete and small number of participants and to publish the commitments and proofs into a public ledger. The participants keep a local and unencrypted copy of the ledger. Figure 2 shows the main case presented in the paper for a close number of banks settlement of transactions. Note that the ledger presents (the greyed area) transaction commitments and proofs, that serves as a proxy to validate the open transaction figures. Upon request by an auditor, the bank(s) disclose precise data and row high (or the equivalent timestamp), and the inquirer can certify the response.

zkLedger’s performance is linear to the number of entities because obscuring the transactions requires a similar effort regardless if the participant is included or not in the transaction. None-the-less, the data model permits to add or remove banks dynamically, which gives flexibility in the business models that can be deployed.

Another core capability of zkLedger is the ability to generate key measures without having to reveal the granular transactions of the ledger. As reported by the authors, these include sums, means, ratios, variance, co-variance, standard

[3] As of Feb 2020, Mirror Network is in Beta, where the information available is transferred from the Mainnet using a batch feed and file system via two cloud service providers. The Mirror network is planned to receive the messages online and in synchrony to the Mainnet. Our model was tested with Kabuto (www.kabuto.sh) and Hedera Mirror nodes.

[4] The has this structure: `currentRunningHash+topicshard+topicrealm+topicnum+consensusSeconds+consensusNanos+sequencenumber+message`. The hash becomes the new running hash and will be input as `currentRunningHash` to calculate the `runningHash` of the next message on the same topic id

deviation, and range proofs. The Range Proofs are stored as Proof of Assets, but additional calculations can be produced, generating new commitments and NIZKs. (1)

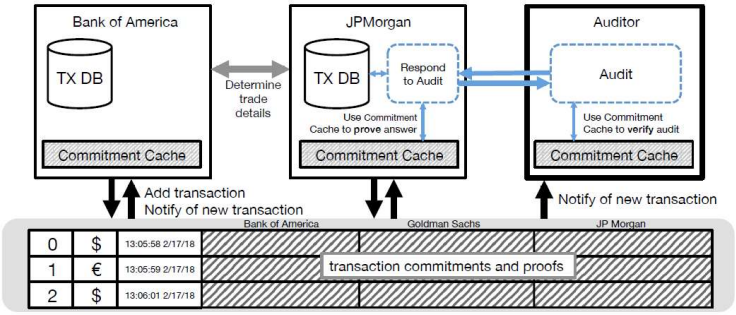

Figure 2. Overall zkLedger Design showing the interactions between the three main entities (banks, auditor, and the shared ledger). Each bank keeps a private state, consisting of the transaction database for transactions the bank originated and the bank's secret key. (1)

2.3. FabZK

FabZK implements a version of the shared ledger of encrypted data suggested by zkLedger but incorporates three significant improvements: Bullet Proofs (3) for the Range Proof of Assets; Disjunctive Zero-Knowledge Proof (DZKP) to replace the Proof of Consistency; and parallelization of processing for transaction data and historical data.

FabZK is implemented on top of Hyperledger Fabric (using Kafka as the Orderer Service) and takes advantage of the different layers and services defined: Peers, Notification, Endorsers, Ordering, and Committer. The overall system shows significant improvements in throughput as reported in Figure 3.

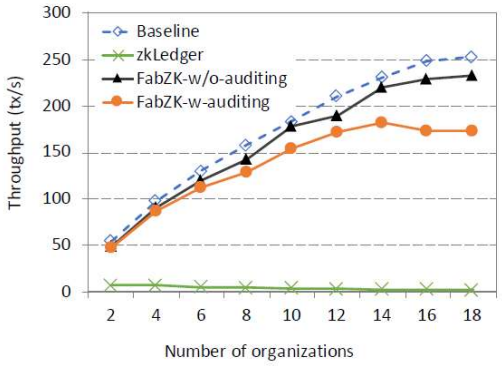

Figure 3. Performance of asset exchange transactions for the prototype using native Hyperledger Fabric APIs (baseline), zkLedger, and FabZK's APIs with and without auditing (higher is better). (2)

3. zkLedger implementation on Hedera

Our implementation takes the core model of zkLedger, including the enhancements on Range Proofs and Disjunctive Zero-Knowledge Proofs from FabZK, and deploys on Hedera Hashgraph. Figure 4 represents the different components of the architecture.

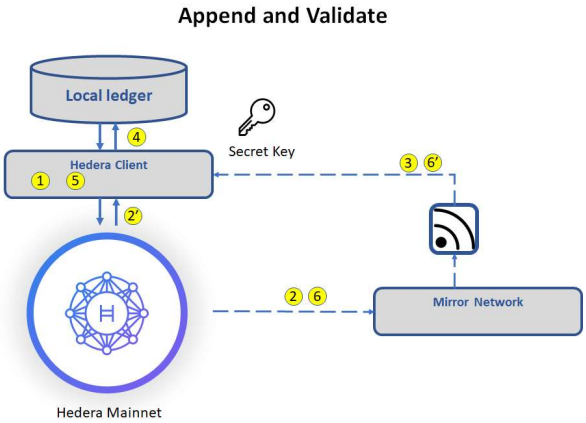

Figure 4. Reference diagram of zkLedger implemented on Hedera Consensus Service

The first model represents obscuring the transaction values, getting an absolute order placeholder, and then building the proofs of integrity (consistency and assets).

1) The client forms the transaction and produces the Pedersen Commitments:

$$Comm(v, r) = g^v h^r \qquad (1)$$

where *g* and *h* are two random generators from the cyclic group $\mathbb{G}$ taken from secp256k1 with $s = |\mathbb{G}|$ elements and prime order *p*, such as $\mathbb{Z}_p = \{0,1,\dots,s-1\}, v \in \mathbb{Z}_p$, and $r \in \mathbb{Z}_p$.

The initial message also passes the Audit Tokens:

$$Token = pk^r \quad (2)$$

for all participants *i,* where *pk* is the public key pair of a Schnorr signature of the participant and *r* are the same as in the Pedersen commitment in equation (1).

Proof of Balance. This proof applies it the ledger needs to maintain total value by proving that all commitments of a row do not create or destroy value (net transfer of value is zero). The implementation evaluates this condition by confirming the point derived by the product of the commitment corresponds to a coordinate that will force the modulo operator to be zero for the generator point of the elliptic curve.

2) The Pedersen Commitments and Audit Tokens are converted into bytecode and is broadcasted to the Hedera Network using a Hedera Consensus Message. The corresponding Topic has to exist previously. In return, the Mainnet gives a receipt and passes the message to the Mirror Network.
3) The Mirror Network broadcast the message response that includes the consensus timestamp, the consecutive order for that Topic, and the hashing of the message and the previous hash. Only those Clients that are subscribed to that Mirror Node and Topic will receive the stream of updates.
4) Once the client receives confirmation, it appends the response to the Local Ledger.
5) Once there is an entry in the Local Ledger, the Client can complete and submit to the Mainnet the missing proofs:

   Proof of Assets: represents the total sum of the committed assets in the column of the spending organization, including the transaction for which the proof is issued, is greater than or equal to zero.

$$RangeProof = ZK(v_{RP}, r_{RP}: Comm_{RP}(v_{RP}, r_{RP})) \quad (3)$$
$$where\ l_0 \le v \le l_p$$

Where $Comm_{RP}(v_{RP}, r_{RP}) = g^{vRP} h^{rRP}$, for *g* and *h* are generators from the cyclic group $\mathbb{G}$ taken from secp256k1. This definition comes as an inner-product Range Proof from Bulletproofs (3), as was implemented by FabZK ledger.

Proof of Consistency: represents the binding between the range proof in equation (3) is consistent with the remaining assets as the network recorded them, and also assess that non-spending organizations have balance commitments that are consistent with the transaction amount. This artifact is another contribution from FabZK ledger.

For the Spending Organization:

$$Token' = pk^{rRP} \quad (4a)$$

where r_{RP} is the random number used for the Range Proof in (3).

$$Token'' = Token \cdot \left(\frac{Comm_{RP}}{s}\right)^{sk} \quad (4b)$$

Where $s = \prod_{i=0}^{m} Comm_i$ is the product of the Participant's commitments from row 0 to row m. sk is the Secret Key of the Spending Organization.

For a non-spending organization:

$$Token' = t \cdot \left(\frac{Comm_{RP}}{s}\right)^{sk} \quad (5a)$$

where r_{RP} is the random number used for the Range Proof in (3)

$$Token'' = pk^{rRP} \quad (5b)$$

Where $t = \prod_{i=0}^{m} Token_i$ is the product of the Participant's Audit Tokens from row 0 to row *m*.

A Hedera message is formed and broadcasted with the serialized format of the Range Proof plus the Disjunctive Zero-Knowledge Proof for all the participants.

6) The Mirror Network receives the message with proofs and distributes to all the nodes that are subscribed to that Topic.

4. Implementation

The implementation is built on Java 1.8, using Maven to add the dependencies. The GitHub repository also includes a copy of Range Bullet Proofs originally developed by authors, eliminating the log outputs. The model consists of the deposit of assets, transfer, and withdrawal for a N-column ledger.

https://github.com/JaimeSilver/zkLedger-hcs

5. Alternative use case

The Internet of Things (IoT) needs to provide data availability and privacy to the users. Data protection yields to the natural discussion about trust, custody, and ownership. The decoupling of authentication of users and devices, as well as anonymization of data broadcasts from the IoT devices (7), integrates naturally with Distributed Ledgers. The opportunity to obscure transactions and make them fully auditable is of exceptional value for any party involved in the supply chain management.

For example, zkLedger and Hedera can provide additional protection to the data when the Devices are broadcasting reads such as temperature, humidity, and geolocation. The final user does not have direct access to the data broadcasted, but in case of a claim, he or she can request the IoT aggregator to reveal the value (timestamp, column, and value) produced by the device that breached a given threshold, without showing any other attribute.

That example can extend to any scenario where the participants need to reveal information on a need-to-know basis.

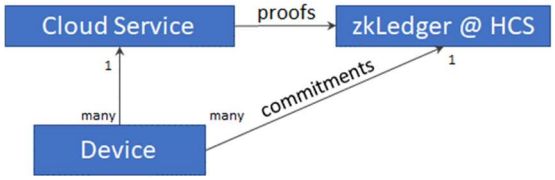

Figure 5. The device broadcasts the state to the Cloud Service using traditional encryption. The IoT aggregator feeds the ledger and broadcasts to Hedera. The auditor needs to query HCS with the values provided.

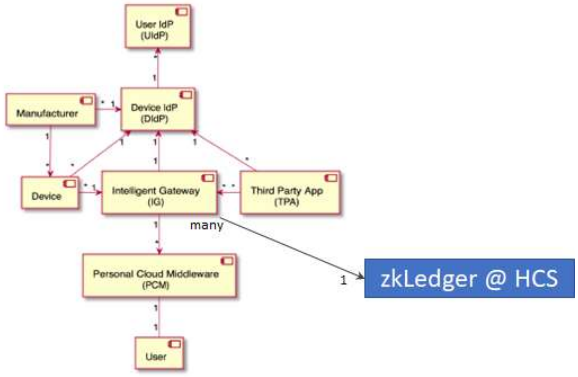

Figure 6. The federated model of OAuthing (7) broadcasts to the gateway, and from there, form the entries for zkLedger@HCS. Signatures represent the different participants without revealing their roles. The auditor needs to query HCS without the need to grant access to the PCM.

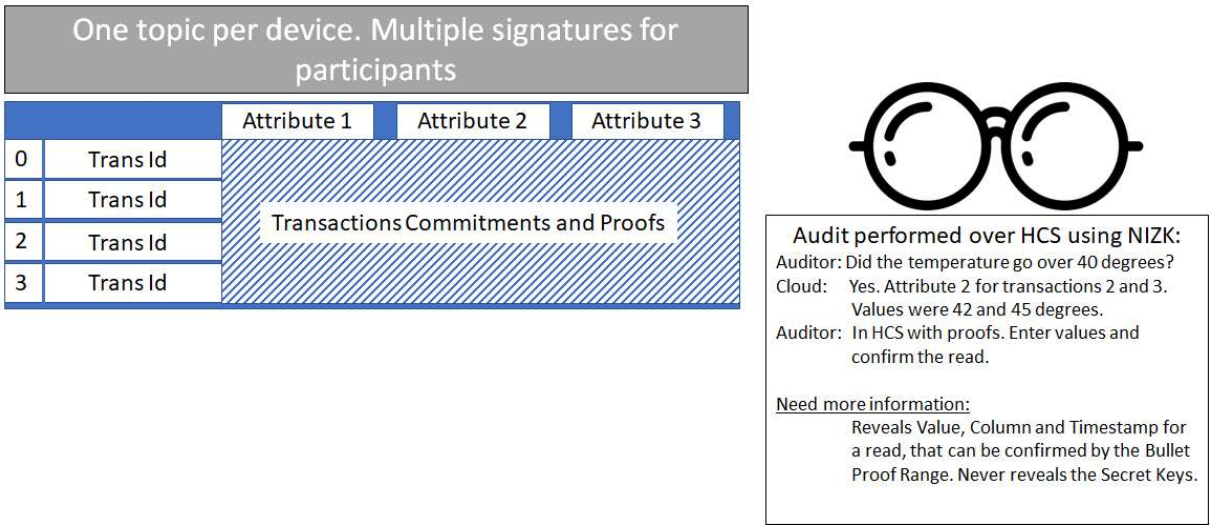

Figure 7. An interested party requests to reveal the information of an attribute, and can be verified based on range proofs issued for that column and timestamp.

The model proposed hides information in a public ledger but can be validated by providing the full range of r and v for all m transactions, range proofs, or alternative calculations as proposed by the zkLedger paper. Note that some proofs are not required since each transaction posted is independent of the previous reads.

6. References

1. **Narula N, Vasquez W, and Virza M.** *zkLedger: Privacy-Preserving Auditing for Distributed Ledgers.* s.l. : Syposium on Networked Systems Design and Implementation, 2018.

2. **Hui K, Ting D, Nerla JL, Shu T, Xiaohui G.** *FabZK: Supporting Privacy-Preserving Auditable Smart Contracts in Hyperledger Fabric.* s.l. : IBM Research.

3. **Bnz B, Bootle J, Boneth D, Poelstra A, Wuille P, and Maxwell G.** *Bulletproofs: Short proofs for confidential transactions and more.* s.l. : IEEE Syposium on Security and Privacy, 2018.

4. **Baird L, Gross B, Thibeau D.** *Hedera Consensus Service (https://www.hedera.com/hh-consensus-service-whitepaper.pdf).* s.l. : Hedera Hashgraph LLC, 2019.

5. **LLC, Hedera Hashgraph.** https://github.com/hashgraph/hedera-hcs-sxc-java/blob/master/hcs-sxc-java-core/src/main/java/com/hedera/hcs/sxc/consensus/OutboundHCSMessage.java. *Java implementation for Message Consensus.* [Online]

6. **Labs, Turi.** https://tlu.tarilabs.com/digital-assets/confidential-assets/MainReport.html. *Confidential Assets.* [Online]

7. **Fremantle P, Aziz B.** *OAuthing: Privacy-enhancing Federation for the Internet of Things.* 2017.

8. **Baird L, Harmon M, Madsen P.** *Hedera: A Public Hashgraph Network & Governing Council v1.5.* 2019 .