Blockchain-based Identity Provisioning and Verification System

Keshava Srinivas, Nithin Ram, Noora Alfayez, Pramod Illuri, Prerit Pathak, Vaanya Gupta

November 29, 2022

1 Introduction

Currently, the method used to store identities of devices is imprinting a secret/identity information in a silicon chip such as an EEPROM or SRAM. These identities are verified using hardware cryptographic operations like digital signatures and encryptions. The disadvantage in this method is that the manufacturing cost of imprinting secrets in these types of silicon is high. Moreover, The hardware required to implement Secure Hash Algorithms for digital signatures is expensive. This method also requires an additional protection mechanism, to prevent invasive attacks, which would consume a lot of power.

The motivation behind a software-based decentralized verification system is that the identity of devices can be verified while preserving privacy, meaning it does not need an additional protection mechanism. In doing so, the cost and power consumption of the devices are also reduced. An additional advantage of implementing this approach as a set of smart contracts on a blockchain-based network is, the code is shared and run on every single device, making it easier to scale reliably.

2 Literature Review

2.1 Hardware based verification

The common reliable authentication approach for device authentication is to have device identities stored in silicon chips. Verification is done using hardware signatures and encryption techniques where cost and power consumption of imprinting secrets is relatively high in certain scenarios [1]. Furthermore, additional mechanisms for privacy protection and governance are required. Recent research suggests that utilizing blockchain as means to address components failure, monitoring, integrity checks, and security for hardware systems [2]. An industry-wide support for blockchain utilization in hardware management shows the value of shifting to a blockchain-based identity provisioning and verification.

2.2 Applications

There are numerous applications for the blockchain-based identity Provisioning and Verification System. RFC 8520 explains specification of what is known as Manufacturer Usage Description (MUD). MUDs are devices assumed to serve a limited purpose and not intended to perform general purpose computing tasks, such as smart light bulbs or industrial system components [3]. The socially verifiable identity verification system described in this document is most relevant to such devices typically present in heterogeneous environment. The presence of multiple devices produced by different manufacturers makes identity verification through software-based approach more enabling and efficient.

3 Implementation

This section explains how to implement a blockchain-based identity provisioning and verification system. In section 4, we provide more details on our POC development and explain our implementation choices. Code repository is available on GitHub.

3.1 High level design

There are 3 main components in the proposed distributed verification system, which are the following:

- The KeyMaker protocol: The identity provisioning component which uses Shamir Secret Sharing [4] to distribute shards of a random key, then appends each shard to the encrypted identity of the device. It is worth noting that Shamir Secret Sharing approach state that any shard Sx cannot be the subset of the key, and all shards S1, S2...Sn, when combined, must not reveal the secret key. Each device has visibility of only one shard. This is an important element to consider when developing the protocol.

- KeyChecker function: The identity verification component of the system. The smart contract performs the verification either through Lagrange interpolation or curve fitting (graph matching) using K shards from subset of the N peer nodes on the network along with the newly added unverified node.

- Blockchain Network: The decentralized network maintains the device identities through the transactions log (worldstate) and manages function calls for social verification through smart contracts (chaincode).

Figure 1 illustrates the reference architecture of the proposed distributed verification system and communication between each component.

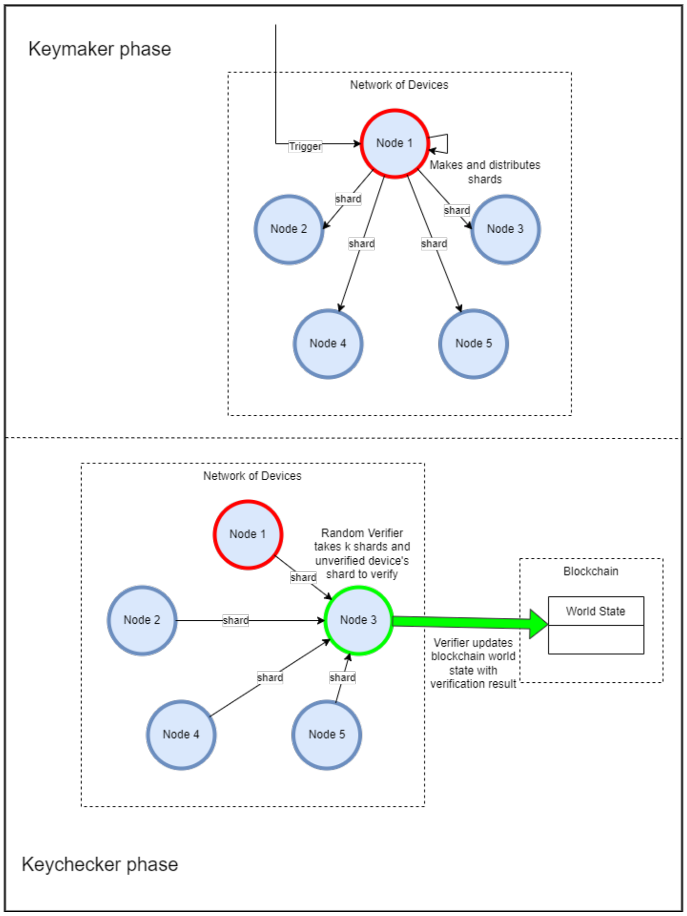


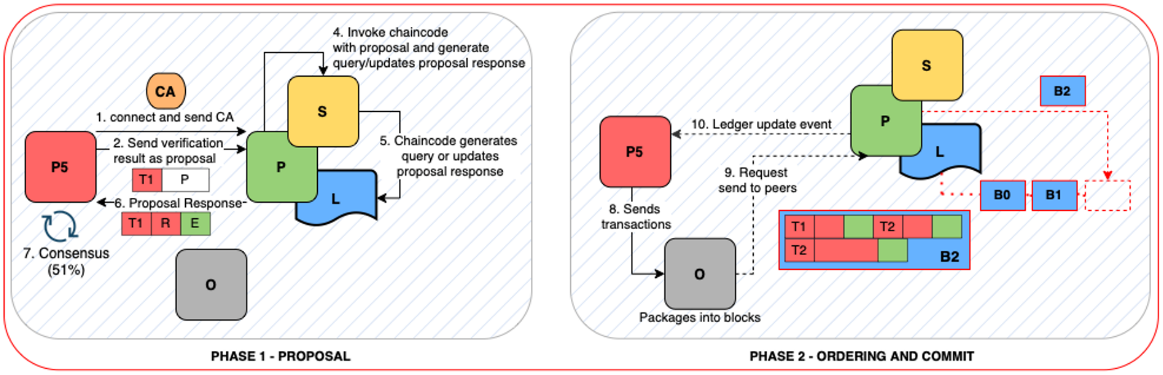
Figure 1: Reference Architecture

3.2 Blockchain design

When a new node is added to the network, the world state is checked and the blockchain is triggered to perform verification. A verification of a node can also be triggered by other events such as device maintenance, replacement, or periodic check. The verification signal on blockchain triggers the following steps for the new node to be successfully verified by already-verified peer nodes on the chain.

- ID provisioning.
- Creation of random key.
- Random key is split into shards (following Shamir's Secret Sharing approach).
- New node identity is encrypted with the generated random key and appended to each of the generated shards.
- Shards are distributed to peer nodes on the network while one shard remains with the new (unverified) node.
- verification is performed by a random valid node on the network using K shards from subset of the N peers
- Verification result is sent by new node to peer nodes as proposal along with its Certificate Authority.
- Peer nodes validate the signature and invoke smart contracts to endorse/reject the new node identity.
- If new (unverified) node receives 51% consensus, it sends the transaction to the Ordering Node on the blockchain which then request peer nodes to update the ledger.
- The blockchain Worldstate is updated, and new node is marked as verified.

Figure 2 shows the flow of steps on the blockchain in further detail.



CA
1. connect and send CA
P5
2. Send verification result as proposal
T1
P
4. Invoke chaincode with proposal and generate query/updates proposal response
S
P
5. Chaincode generates query or updates proposal response
L
6. Proposal Response
T1
R
E
7. Consensus (51%)
O
PHASE 1 - PROPOSAL
S
B2
P5
10. Ledger update event
P
L
9. Request send to peers
8. Sends transactions
B0
B1
T1
T2
T2
B2
O
Packages into blocks
PHASE 2 - ORDERING AND COMMIT

Figure 2: Blockchain Design

4 Development Approach

To build the POC for this system, we implemented a custom developed version of the Hyperledger fabric network. We then deployed the KeyMaker protocol and validation function as smart contracts on the chain. The custom blockchain network implementation allowed us to facilitate access to nodes on the network and store/retrieve data as needed. The alternative managed Hyperledger fabric service by AWS caused access challenges to the peer nodes and limited functionality. We were able to overcome this challenge by installing the Hyperledger fabric on docker containers and running the required chaincode (smart contracts). Furthermore, one critical component that is unique to our development approach is the shards verification mechanism. We implemented Lagrange interpolation to verify the shards. We're using Lagrange interpolation because its a more computationally feasible alternative to curve fitting. That is because of how large numbers are stored in a computational device. Also, it can perform the verification in a single computation compared to the other alternative, which needs to compute 2 curves and then perform comparison (curve fitting). Lastly, our POC implementation offer flexibility of testing and opportunity for further customization. We used CouchDB on top of our blockchain layer for better visualization. Figure 3 illustrates our development components and interactions.

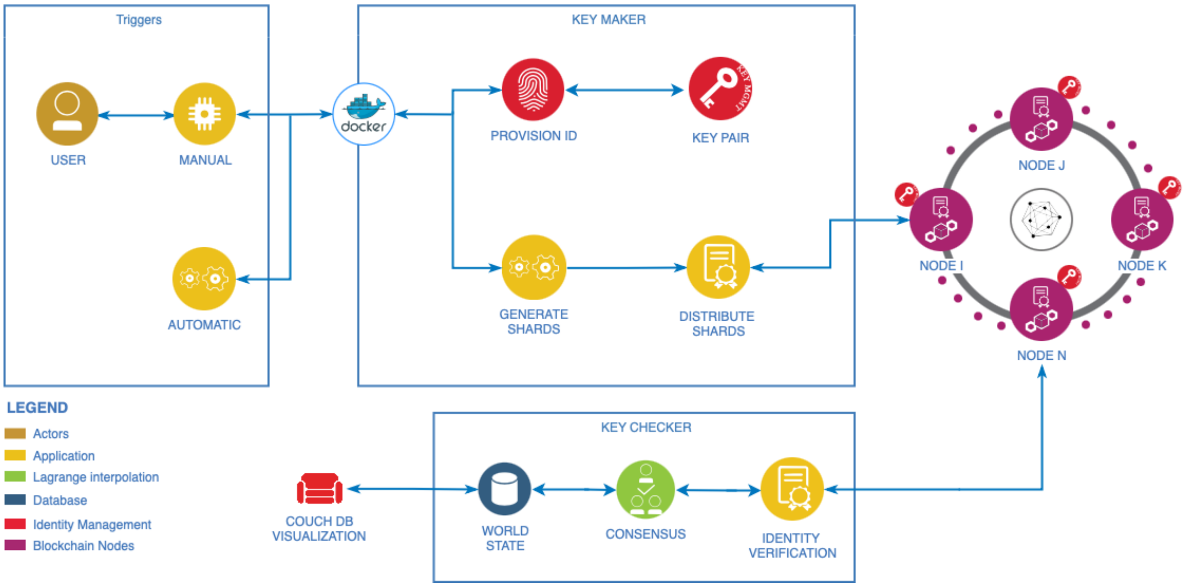


Figure 3: System Diagram

5 Conclusion and Recommendations

There are numerous advantages to shifting to a software-based identity verification including reduction in cost and power consumption. In our project, we demonstrated that it is possible to provision a device identity and authenticate it using subset of peer nodes on the blockchain network. There are still many opportunities to develop this further and have it ready for market-adoption. Our recommendations are:

1. Automation of idetity provisioning and authentication triggers. For example, programming the system to trigger an authentication of all devices on the network periodically or upon connection of a new device.

2. Deployment and testing of protocol on Raspberry Pi devices or in different network setup (open netowk).

3. Optimization of the blockchain network implementation to reduce costs of managed services.

4. Evaluation of security concerns or possible attack scenarios and deploying necassary protection measures.

References

[1] M. Malik, "Digital identity management." https://youtu.be/9XgmZtdlSBQ, Sep 2021.

[2] U. S. Ashish Kundu, Zehra Sura, "Collaborative and accountable hardware governance using blockchain." https://ieeexplore.ieee.org/document/8537824, Oct 2018.

[3] D. R. E. Lear, R. Droms, "Manufacturer usage description specification." https://www.rfc-editor.org/rfc/rfc8520.html, Mar 2019.

[4] A. Shamir, "How to share a secret." https://web.mit.edu/6.857/OldStuff/Fall03/ref/Shamir-HowToShareASecret.pdf, 1979.