Enhancing the privacy and security of Hyperledger Fabric smart contracts using different encryption methods

1. INTRODUCTION

Hyperledger Fabric is an open source immutable distributed ledger technology (DLT) platform.

- Uses **blockchain technology**
- The main purpose is creating **private** and **permissioned** blockchain networks
- Currently used by many **enterprise companies**
- scenarios where **data must be kept private**, such as finance, trading, dispute resolution, healthcare record-keeping

2. RESEARCH QUESTION

How to enhance the level of protection of data stored in the ledger of Hyperledger Fabric networks using strong methods of encryption?

Author: Rado Stefanov
tudelft@rstefanov.eu

Responsible professor
Prof. Dr. Kaitai Liang

3. MOTIVATION

- **Vulnerabilities** in
  - CouchDB, State Databases, peers, ledgers
  - Might result in leaks of private data
- **Increase the protection of Fabric** against other currently unknown vulnerabilities
- Support scenarios where
  - Information needs to be **preserved private**,
  - **BUT** members of the network **still need to validate business logic** using a smart contract

4. METHODOLOGY

- Study Fabric documentation
- **Analyse** research on
  - Fabric security
  - encryption methods for blockchain networks
- Study **source code** of Hyperledger Fabric
- Implement and test possible **encryption methods**
- Execute **experiments for time and speed performance** of chosen encryption algorithms

5. CONTRIBUTION

Symmetric encryption

- All peers have a symmetric key
- Each peer encrypts before storing to ledger
- Each peer decrypts after reading from ledger / state
- Key stored in secure docker container
- Drawbacks
  - Higher memory and computational time required
  - CouchDB range queries for numbers are not possible

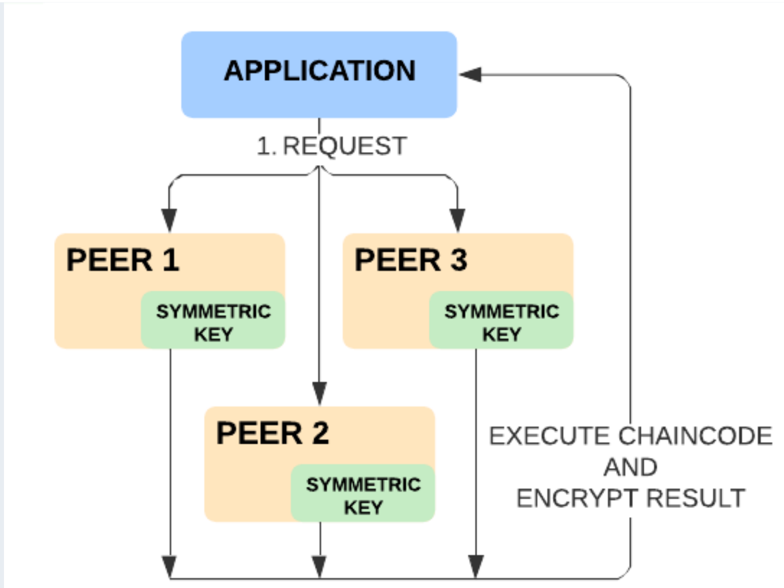

Fig 1: Overview of symmetric encryption principle with 3 peers

	asset1	blue	300	Tomoko
	asset1	KaWGxAe8ulNyKjBmz...	i90/4RGyvO9CEASKZ...	OlxU/FdjaYiSihFCsZxK...

Fig 2: CouchDB asset, first without encryption, second with encryption

[ZKP] Paillier encryption for voting use cases

- Use ZKP Range proof to validate vote validity without revealing vote value
- Use additive homomorphism to obtain final voting result
- Ensured determinism by ZKP for proving that encrypted number is zero

If 2 numbers are encrypted

- 23 -> b0ae84611cfa33aa12afe4546509
- 12 -> c7289957011b1bf315d245659eb1
- b0ae84611cfa33aa12afe4546509 + c7289957011b1bf315d245659eb1 = 319a0a9e5daaa1029b5b7
- 319a0a9e5daaa1029b5b7 = (23 + 12)

Fig 3: Demonstration of Paillier encryption with additive homomorphism

6. CONCLUSION

- Symmetric encryption
  - Enhances level of security
  - Developers should follow general security rules, such as strong CouchDB passwords, even when using encryption
- ZKPs
  - Useful in specific scenarios
  - More complicated and hard to implement
- Future Work
  - Implement plugin for symmetric encryption
  - Extend ZKP research for Fabric