Secure smart contract data sharing for IoT devices

Author: Julio Vega Sanchez, j.r.vegasanchez@student.tudelft.nl
Supervisor: Kaitai Liang, kaitai.liang@tudelft.nl

1 INTRODUCTION

IoT is gaining increasing popularity and data needs to be shared more efficiently between data owners and users. There is a need for sharing this data in a decentralized manner using blockchain.

This research proposes a solution for this problem that is ***secure***, ***efficient*** and ***scalable*** by combining HyperLedger Fabric smart contracts with proxy re-encryption.

2 BACKGROUND

HyperLedger Fabric (HF) is an open-source blockchain platform that runs smart contracts. Multiple organisations can join a network and internact through the contract.

Proxy re-encryption (PRE) is an encryption method in which a third-party (proxy) transforms the ciphertext under one key into a ciphertext under another key. The proxy does this by receiving the keys of the delegator and the delegatee and generating a new key (other variants exist).

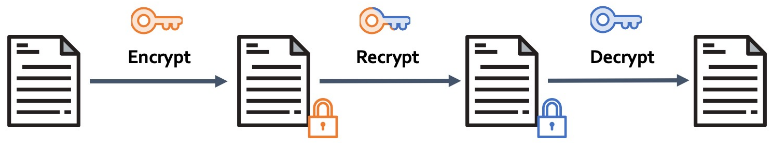

Figure 1: Schematic representation of PRE

3 METHODOLOGY

1. Research theoretical concepts and implementation techniques
2. Design scheme suitable for IoT data sharing using HF and PRE
3. Create smart contract and demo application
4. Perform tests on efficiency and scalibility

4 PROPOSED SOLUTION

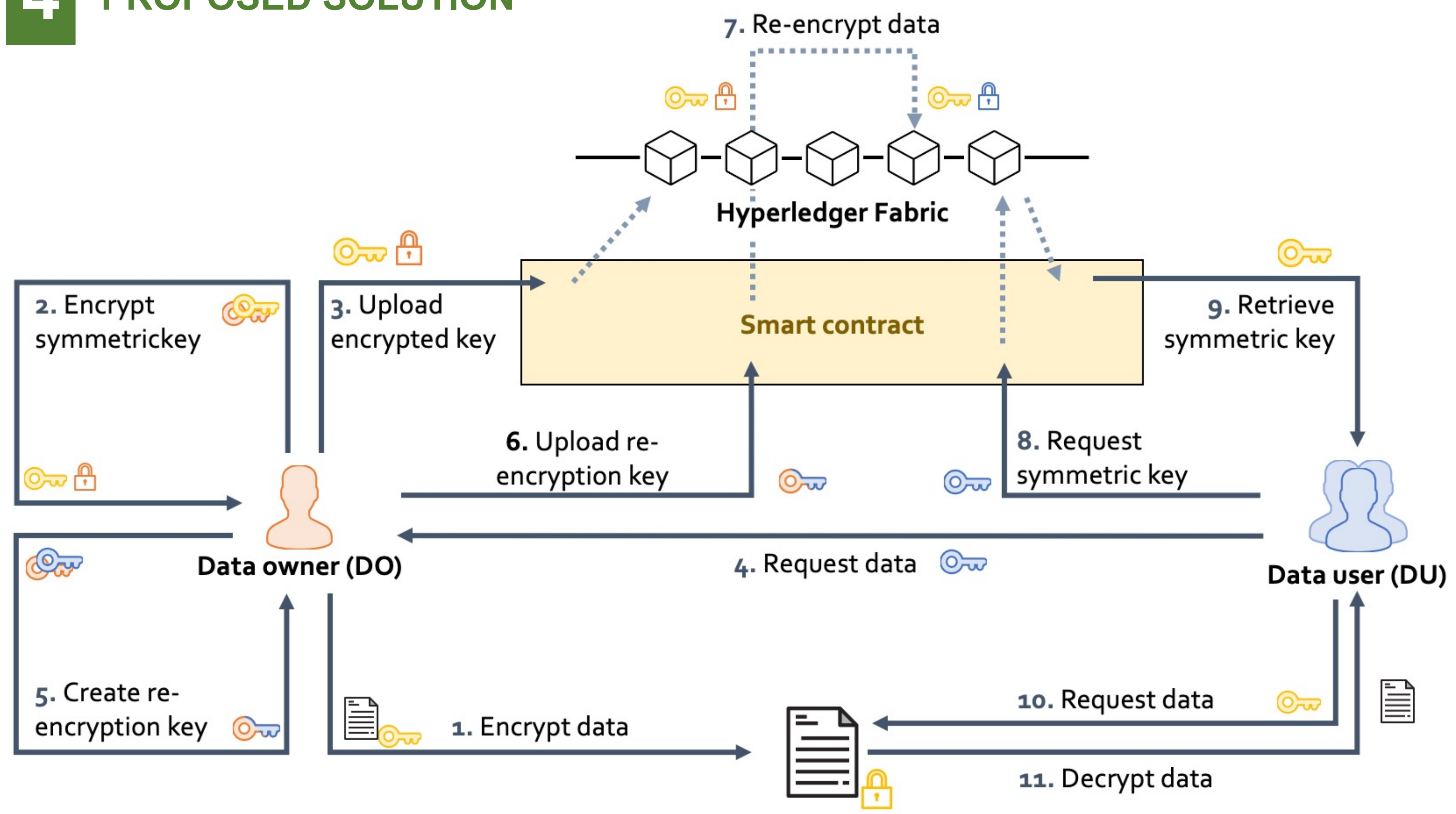

Figure 2: Schematic representation of proposed solution that incorporates symmetric encryption (step 1, 2, 10 and 11) with PRE (step 5, 7, 8 and 9) through an HF smart contract and local methods

5 IMPLEMENTATION

The proposed solution is implemented in HF and a demo application. The PRE methods hold the following properties:

- Non-interactive; the DU only shares his public key
- Multi-hop; the re-encrypted data can be accessed by the original user

6 RESULTS

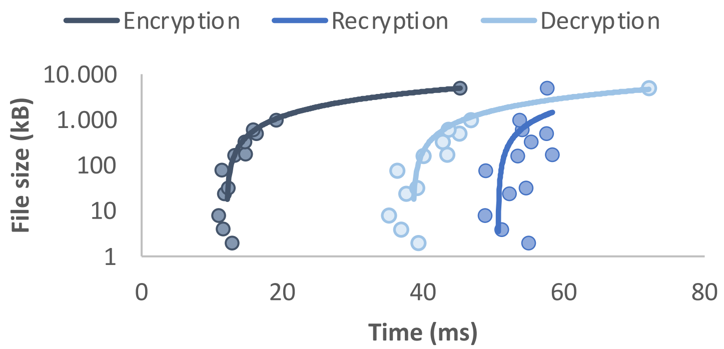

Figure 3: Speed performance of steps in implementation compared to different file sizes

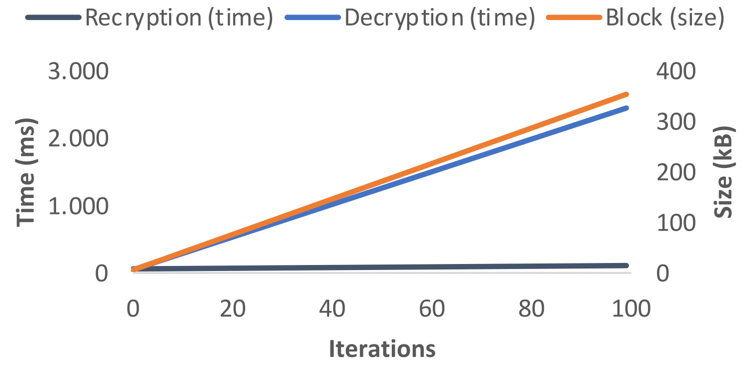

Figure 4: Speed and size performance after increasing re-encryption iterations

7 CONCLUSION

The proposed solution is ***scalable*** since it only saves encrypted symmetric keys of 256 bits. Re-encryption increases block size at minimal costs. The solution is ***efficient*** because encrypting files is comparable to industrial standards. Furthermore, re-encryption increases latency at minimal costs.

The implementation does not meet all requirements and needs further improvement.