Enhancing Hyperledger Fabric smart contracts with secret sharing

Ali Kahawati*
Supervisor: Kaitai Liang†
EEMCS, Delft University of Technology, The Netherlands

Abstract

Blockchain networks have acquired ongoing notoriety among associations that need to utilise the security perspectives that blockchain gives. Hyperledger Fabric (HF) is one of the most widely utilised distributed network technologies, most ordinarily applied in situations that require private information to be maintained safely and secretly. Use case models are finance, commerce and healthcare record-keeping.

HF has been shown to have vulnerabilities that might allow hostile attackers to gain access to the data contained in the ledger or the state database, according to many studies.

Using Shamir's Secret Sharing (SSS), this paper provides a way to increase the security measurements of information stored on the ledger or in the state database. As a result of this research, design details for implementing these privacy-protection measures are given.

In conclusion, although this design improves the security of HF, it also affects the runtime and code complexity.

1 Introduction

The road towards an increased adoption in applications of new blockchain technologies, such as Hyperledger Fabric or more secure solutions to store data on open-source distributed ledgers, is still blocked by the presence of vulnerabilities. Addressing these concerns will offer one of the best and fastest solutions to many applications.

For instance, the current traditional insurance application procedure, like many other applications, may have various disadvantages. The main ones are the following:

- The data of a client is easily accessible for company departments when proceeding with his application.(Leak of private data between individuals)
- Slow verifiability of the important information of a client.

*{a.kahawati}@student.tudelft.nl
†{Kaitai Liang}@tudelft.nl

Using systems like Hyperledger Fabric (HF) offers solutions to both disadvantages. HF is defined as a distributed ledger technology blockchain platform for permissioned blockchains [1]. The technology of HF is becoming more popular among big business companies [2]. The use of it is now widely accepted, like in the insurance business field. But unfortunately, it is not perfectly secure, since the data on the ledger is stored as plain text. And since the GDPR" (General Data Protection Regulation) or "AVG" (Algemene verordening gegevensbescherming) (as called in the Netherlands) has been revealed in 2018 [3], [4], more privacy is required for it to be easily used around the EU.

All in all, it is important to research a way to solve the vulnerability of revealing private data to all peers on the same channel in decentralised systems for general use cases. By achieving that, applying the new technology of Hyperledger Fabric will become easier, in addition to providing speed and security.

This paper is proposing a system that combines a cryptographic method called secret sharing with Hyperledger Fabric smart contract. This will enable saving the data on a fast distributed ledger as encrypted data and thus securing it from any malicious attack.

Motivation

This paper will focus its research on perfecting the security of Hyperledger Fabric smart contracts by adding a secret sharing scheme to it.

Secret sharing alone has a lot of feasible applications in many scenarios in network based transactions for instance, attribute-based encryption, key transfer protocols and secure multilateral computation [5].

The main goal of this research is to solve the issue where an attacker has gained access to an organisation, the attacker will be able to see all assets stored inside the ledger of that channel. Because the data is stored as plain text in the ledger, attackers or malicious peers can access private data or maybe keys, which can be used to perform some malicious transactions. For instance, changing the data stored on the ledger or attack the network. It also will solve issues like when an attacker compromises encryption keys using secret sharing.

Next to that, It will improve traditional systems, like insurance systems, by offering:

- A distributed ledger solution supported by a standard design providing high degrees of confidentiality.
- Scalable and fast ledger while preserving privacy.
- high ability to customise the consensus and network protocols.

Contribution

This research aims to answer the following question, *How to enhance the security and privacy of Hyperledger Fabric's smart contracts using secret sharing?*.

The question can be broken down to the following subquestions, which can be answered separately:

1. *Are Hyperledger Fabric's smart contracts secure? To what degree are they secure?*
2. *What is secret sharing, how does it work?*
3. *How to combine secret sharing with a Hyperledger Fabric smart contracts?*
4. *How does such a combination affect the performance? What are the drawbacks and what are the benefits?*

The proposed system idea will provide the next properties:

- Authorization where only authorised departments will be able to write to the ledgers.
- Slightly reduce the key management complexity.

As a result, private data will be securely shared between organisations like hospitals, insurance companies or government's institutions for instance.

Structure

The rest of the paper will answer the stated question in 8 sections presented as follow:

Section 2 will provide the background of the technical expertise required to understand the technical parts of this research.

The research methodology and the implementation idea of this study will be discussed in section 3. Section 4 shall present the privacy enhancements achieved during the experimental work, which will be followed by the results and analysis in section 5.

Section 6 will discuss other possible vulnerabilities or improvements to this research, which will be followed by section 7 responsible research. Section 8 will conclude the research and propose future work areas.

2 Background

This section presents the technical expertise required to understand the technical parts of this work.

To reduce the cost of trust, as the primary purpose, the use of *Blockchain* has been objectified by Nakamoto [6]. Offering peer-to-peer networking, thus removing centralised third-party dependencies, which offers trust for high prices. By doing so, Blockchain reduces the validation time of transactions and provides consistency and immutability.

2.1 *Hyperledger Fabric*

Offering an enterprise-grade, code base and open-source distributed ledger framework, the Hyperledger project has been found by the Linux Foundation in early 2016 and not so late after that a distributed ledger platform has been implemented for running smart contracts, i.e. *Hyperledger Fabric*. The Hyperledger Fabric [7] is a Blockchain framework, a permissioned one to be specific, i.e. peers need some credentials to be able to write to the ledger. The Membership Service Provider (MSP) ensures that Fabric is a permissioned network. It identifies every member on the network, as a list of permissioned identities. MSP allows trust between members without revealing their private keys.

Figure 1 represents a transaction flow instance of a Hyperledger Fabric network. A network consists of organisations, chaincode, ordering services, channels and optionally private data collections. Before joining a network, organisations must agree to its policies and permissions stored on the channel configurations. Organisations use applications located outside the network to interact. Applications can access the ledger through chaincode.

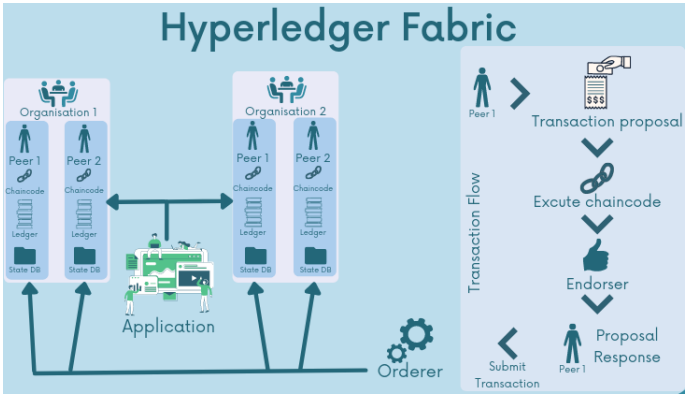

Figure 1: A transaction flow in Hyperledger Fabric architecture, shows the required steps for a transaction. It starts with Peer 1 invoking the chaincode.

HF is a distributed system, hence Peers are expected to execute as independent components rather than on the same server in the same Docker network. And as stated in the Fabric documentation, the endorsing peer process is segregated from Chaincode, which operates in a secure Docker container. This is why each peer must create, install and run their chaincode container. In addition to that, TLS private keys and certificates are stored in this container.

Chaincode (Smart contract)

Smart contracts (chaincode) [8] have made transactions between parties easier than ever using a blockchain network. Those executable programs i.e. smart contracts have opened up a diversity of new possibilities. Since smart contracts are event-driven and self-enforcing programs, by using their rules of any type of affairs can be easily implemented and controlled. In addition to that, smart contracts being customizable offers a lot of advantages. With Fabric's chaincode,

multiple smart contracts can be deployed. It handles business logic which members of the network have agreed to [9]. A chaincode on one network cannot be accessed by another chaincode or other members who didn't agree with it and are not on the same network. Chaincode has access to the ledger and all transactions happening, it even has the last word to decide if any transaction will be accepted or not thus adding the data to the ledger or not.

All parties on the same channel have access to the chain code on that channel which means also that they can read the data that is accessible by the chaincode.

Ordering service [10] is used to avoid collisions when adding new elements to the ledger. The Hyperledger Fabric ledger already implements this feature.

Private data and channels

HF offers the ability for subsets of organisations on the same channel to work with private data [11] instead of creating channels every time they need to. Using private data collections, organisations can overcome the overhead of creating many channels. They also can commit, query and endorse private data. Two elements make up private data, the actual private data and a hash of that data.

2.2 Symmetric Encryption

Symmetric encryption is an encryption type that uses only one secret key for both encryption and decryption, see figure 2. Using symmetric encryption algorithms, data is turned into a form that is incomprehensible to anyone who does not have the secret key to decrypt it.

Figure 2: the architecture of Symmetric Encryption.

2.3 *Secret sharing*

Secret sharing is a cryptographic technique for privacy-preserving data sharing between peers. Secret sharing systems are important and they are the pillars of many safe protocols. They work by distributing shares, see figure 3, between parties in such a way that the secret can only be reconstructed by permitted subsets of those parties.

Secret sharing schemes have a wide range of applications in both cryptography and distributed computing including secure multiparty computations, Byzantine agreement, access control, threshold cryptography, generalised oblivious transfer and attribute-based encryption [5]. There are two types of secret sharing schemes, schemes for secret sharing in general access structures and schemes for the threshold case introduced by Shamir [12] and Blakley [13]. The general access structures were introduced by Ito, M., Saito, A. and Nishizeki, T.[14].

Figure 3: the architecture of the Secret Sharing method.

In threshold schemes, only qualified groups are capable of reconstructing the secret, where the qualified groups are all subsets with cardinality greater than or equal to a threshold t.

The general access structure is also a set of qualified groups Γ but it must satisfy the monotone property, $X \in \Gamma$ and $Y \subseteq X$ then $Y \in \Gamma$ and where X and Y are subsets, to be capable of reconstructing the secret.

As stated by [5], the existing schemes for general access are unfeasible. The reason is that the size of shares in the number of parties in the access structure is exponential. Unfortunately, this is true for both explicit and implicit access models.

Shamir's Secret Sharing

One of the most famous threshold schemes of secret sharing is Shamir's secret sharing, which is a key cryptographic algorithm that permits private information, or "secrets," to be securely distributed over an untrustworthy network. It is a Key-less approach that is employed to keep personal data[1] secure and safe.

This simple and elegant scheme has been constructed by Shamir [12]. Sets can be divided into authorised and unauthorized sets, where every authorised set whose size is t an integer where $1 \leq t \leq n$ and where n is also an integer representing the total number of participants, see figure 4.

The domain of shares and secrets in Shamir's scheme is the elements of some finite field F_p where $p > n$ and p is some prime power. This means that shares are taken from the same finite field as secrets, which implies that Shamir's scheme is ideal. In addition to that, it has homomorphic properties since it is a linear scheme. Its linearity comes from the fact that computing the values of some linear transformation can be used to generate the shares and reconstruct the secret [15].

It can also be used to create safe multiparty computation protocols because of its multiplicative features. This is only in the case when the ratio between the number n of participants and the threshold t is large enough. In that case, Shamir's scheme exhibits homomorphic features concerning multiplication in the finite field.

There are a couple of reasons why this research chooses to use Shamir's scheme:

- The abstract underpinning of Shamir's approach allows for great proofs and applications.
- It is simple to switch and change shares while maintaining the same secret.

[1] Biometric data, private keys, or any other type of personal information that shouldn't be shared with the public.

- It is simple to add or withdraw shares without affecting other shares.
- The possibility for each person to have more than one share.

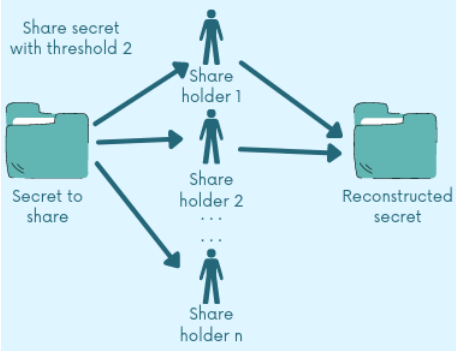

Figure 4: Shamir mechanism. Share secret shares between n users and using only t users to reconstruct the secret

3 Methodology

This section will contain a description of the methodology used to answer the research question and its sub-questions mentioned in 1.

3.1 *Literature research*

Toward the start of the research project time frame, a broad review was performed on the current status of Hyperledger Fabric security.

First of all, the terms used in the search on google scholar are as follows: (”secret sharing”) AND ((”Hyperledger Fabric” AND ”smart contract”) OR chaincode). Using those terms, most of the sources mentioned at the reference section have been found. Thanks to TU Delft library chrome extension, most of the results were made accessible.

Number of the results was high, especially that Fabric is now very popular in the research field around the world. Furthermore, another two methods have been used to make sure that enough resources are found to strengthen the results of this research. Using wildcard searching method and Citation searching to locate related literature that was not caught by the first Boolean logic search query.

Related research

To find the vulnerabilities of Fabric technology, multiple studies have been done to test its security. In [16] Hyperledger fabric’s security architecture has been analysed. Some vulnerabilities have been shown, like Wormhole attacks, DOS attacks and Majority’s decision. Some of them are related to the security of the network and other to the security of chaincode like Random key generation.

According to Dabholkar, Vishal, Ahaan and Saraswat [17], a compromised MSP, a noxious Ordering Service and malicious validators, all contribute to a significant number of possible security issues. Paulsen [18] shows other potential risks like ”updates using rich queries” or ”range query risk” which can exploit Illegal value propagation and changing the expected behaviour of smart contracts using global variables.

Although a lot of studies have been done to issue Fabric’s vulnerabilities, not as many countermeasures has been done for general use cases. Hasanova [19] and Putz [20] did not only detect risks on blockchain but also stated that data can be encrypted before storing it in the ledger. Likewise, Fabric’s documentation [21] expresses that, in addition to channels and private data, data encryption can be utilised as a type of privacy protection via file system encryption on the peer for data at rest. Whereas data in-transit can be encrypted via Transport Layer Security(TLS).

The method of secret sharing [5] has been used to improve the privacy of Blockchain generally in a couple of different ways like in [22] where Mao and Abdihakim enhanced authorized access to medical data. And in [23], where they fairly reconstruct secrets in a trust-less network in one round. Tso, Liu and Hsiao [24] have implemented the first decentralised e-voting system and bidding system, where they use secret sharing, in addition to other cryptographic techniques.

But because Hyperledger Fabric smart contracts have different applications compared to other blockchain platforms, this implies, it also has other security needs.

There are some papers which worked on increasing the security and the privacy of the Hyperledger Fabric in different ways and for different systems, like e-voting and bidding systems. For instance, [25] implements an E-voting system which is decentralised and secure using blind signature and Hyperledger Fabric platform.

Kyazhin and Popov [26], improved the system implemented by [25] where they have resolved a couple of performance issues like constructing a two-stage anonymization using link-able ring signature and Idemix and without the need to change Fabric’s standard signature scheme.

Another anonymous credential system has also been implemented, it is called Idemix [27], which increases the security and privacy of transactions on Blockchain.

Other studies, like Benhamouda and Halevi in [28], implemented a demo using Secure Multiparty Computation to Support Private Data on Hyperledger Fabric.

[29] also used secret sharing but with zero-knowledge proofs and homomorphic encryption to protect the privacy on Fabric using secure multiparty computation (MPC) .

Some papers also provide other encryption methods for more general use cases like [30]. Rado [31] provides a very good way to save encrypted data on the ledger using symmetric encryption and Paillier encryption.

3.2 *Implementation*

Before the implementation of the prototype took place, the Fabric test network has been set and used to become acquainted with the workflow of smart contracts on Hyperledger Fabric. Fabric provides a very clear tutorial to get started with it [32].

The implementation by itself is meant as an extension of the smart contract example provided by the Hyperledger Fabric test network tutorial.

In general, smart contracts contain assets and functions, the examples provided by the tutorial store the data on the ledger

as plain text. Rado [31] has extended this implementation using symmetric encryption to be able to store the data encrypted on the ledger. This implementation will also use the same idea but then add secret sharing to it to increase the security.

4 Experimental work

To achieve the wanted result, the next sections will explain how the data is encrypted, how private keys are managed and what is the role that secret sharing is playing to improve the security.

4.1 Encryption & Decryption

External attackers that breach the security of Hyperledger Fabric and get access to the state database or ledger will be able to read the stored data, if the data stored is raw, i.e. plain text. To secure the data, it can be encrypted, and doing so the attackers will be unable to read the stored data since the data will be encrypted.

Encryption of the data can be achieved using either of Symmetric Key Systems or Asymmetric Key Systems. In addition to that one of the multiple encryption algorithms must be used, like Triple DES, AES, RSA security, Blowfish or Twofish.

Rado [31], has already researched this area and came to the conclusion that using a symmetric key encryption is secure enough. Other encryption methods are also possible, but since Ebrahim, Khalid, Khan and Bin[33] and Atwal, Umesh and Kumar [34] showed that AES, the algorithm used in [35], when compared to other prominent symmetric encryption methods like RSA DSS, 3DES, Blowfish, and others, is the most secure symmetric encryption algorithm. And when compared to asymmetric encryption, AES is faster and more efficient encryption method.

Note that the method studied by this research can also be applied to Asymmetric Key Systems.

4.2 Privacy Enhancement

Since Rado [31] has already explained how Symmetric Encryption can be applied to encrypt the data on both the ledger and database, this paper is not going to explain that part, but only going to dig further in improvements using secret sharing.

Using secret sharing on the data before storing it on the ledger, doesn't add any improvements.

To increase the security of the Hyperledger Fabric this paper uses Shamir's Secret Sharing to secure the symmetric key used to encrypt and decrypt the data. The next section will explain how this is done and what are the effects of doing so.

Shamir's Secret sharing role

Shamir is used to divide the secret key, used for encryption and decryption, into shares. Every peer on the channel will have one share, i.e. part of the key. This means that no one peer will be able to recover the secret key alone. Doing so, even if an attacker was able to get access to one of the peers on a channel, by getting its share, he will not be able to do anything with it. The only condition for attackers to compromise the secret key and reconstruct it is to gain access to all peers on the network. The algorithm 1 shows how to use Shamir's Secret Sharing.

Algorithm 1 Divide key

Require: $n \geq 2$ ▷ Is the total number of peers to share the key with
$2 \leq Threshold \leq n$ ▷ Is the required number of shares to recover the key
$secret$ ▷ Is an encrypted key, or just a key
function DIVIDE KEY(n, threshold, secret)
$r \leftarrow ShamirSecretSharing(n, threshold, secret)$
return r ▷ Return an array containing all the shares (n share)
end function

This operation must be done in specific conditions, for example, the original key might be destroyed after making shares, and the threshold is unknown by any peer. If those two conditions are not met, then this whole process does not add any security, but only adds complexity to the process. This process can be seen in figure 5.

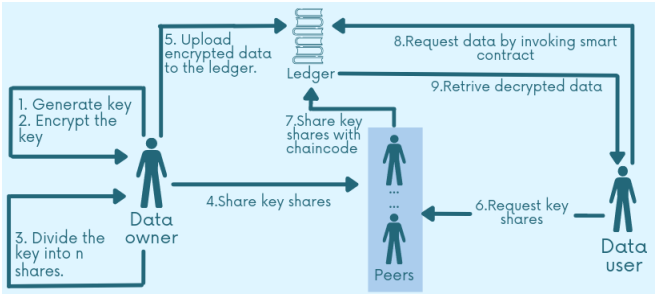

Figure 5: Schematic representation of this enhancement.

Hyperledger Fabric Endorsement policy

To be able to retrieve a secret key, at least a couple of peers must share their shares with the chaincode when a peer invokes the chaincode. Another option is to share the shares between peers. Sharing shares between peers on the Hyperledger Fabric is not something that is easily done. Because of the leak of time, this research will not be able to implement something to deal with this problem. But using the endorsement policy will ensure that at least a couple of peers will be available when some peer invokes the chaincode at that same time. Doing so a couple of shares will be shared with the chaincode (or with the peer invoking the chaincode), ensuring that the recovery of the secret key is possible again.

Endorsement policies are defined by default in the chaincode specification. But using state-based endorsement, endorsement policies can be overridden.

4.3 Key management

All data security is built on the foundation of key management. Because data is encrypted and decoded using encryption keys, the loss or compromise of any encryption key renders the data security mechanisms in place ineffectively. Ad-

ditionally, keys ensure the secure transfer of data over an Internet connection.

Key generation

The first step of key management is the generation of a key. This step forms the basis, since it assures the security of a key. If the encryption key is generated using a weak encryption method or in an insecure location, any attacker may easily figure out what it's worth or could compromise it when it is created. There are multiple schemes to generate secure keys like [36]. This paper will only refer to the importance of having a secure key and secure key generation method.

Key Distribution

Using the proposed method of secretly sharing the secret key, no distribution of the secret key is needed. This add another layer of protection, increasing the safety of the key used for encryption. Even if an attacker tends to execute a man-in-the-middle attack, he will only be able to get one share. Not knowing that this share is not enough to recover the data and not knowing the number of shares needed to reconstruct the key. Using Shamir's secret sharing not only secure the distribution of a key but also ensure that only the chaincode will be able to retrieve the key and use it. This will prevent unauthorised users from retrieving the key and prevent misusing it.

The algorithm 2 will handle the distribution of shares between the peers, using some random communication tool.

Algorithm 2 Distribute key shares

Require: $len(peers) \geq 2$ ▷ The array of peers to share the key with must contain more than two peers
function DISTRIBUTE KEY SHARES(peers, threshold, secret)
$shares \leftarrow Dividekey(len(peers), threshold, secret)$
▷ The array of shares, $len(shares) = len(peers)$
for $peer$ in $peers$ **do**
Send a $peer$ a random $share$ from $shares$
end for
end function

Key Rotation & Backup/Recovery

Secret keys tend to be unusable after a period, which is called a cryptoperiod. When this happens, the key must be retired or revoked and replaced with a new one. This process increases the security of the data encrypted since keys are always protected against compromise. It also adds more complexity to the encryption process, since every time this happens the data encrypted using that key needs to be decrypted and re-encrypted.

Because key shares are distributed equally between peers inside organisations, key recovery can be done safely. This is connected to whom is responsible for generating a key and creating its shares. He can easily request the shares from all other peers and reconstruct the key either outside or on the network.

Storing Keys

Peers stores their parts of the secret key on their own Docker container. To communicate each share with other peers refer to 4.3 or the chaincode.

There are multiple ways to store private keys, each has its advantages and disadvantages. As a result of dividing a key into several shares after applying secret sharing, storing cannot be done on the chaincode container or other central option. Since this will make it easy for an attacker to combine shares and retrieve the secret key. But this also decreases the complexity of storing it. Another option is to store some shares using private data collection and the rest of the shares on the peers. In this case, no communication between peers is needed, and it eliminates the need for another tool to ensure safe communication between peers. Depending on the threshold of shares, a division can be done. For instance, if the threshold is 2 and the number of shares is 10, one share should always be stored using the private data collection and other shares will be divided between peers. In this case, peers will be able to reconstruct keys without the need to communicate with other peers.

Next are some options peers can make use of to ensure the safety of their shares:

- Using HSMs: Using a hardware security module (HSM) is one of the safest ways to store cryptographic keys. HSMs are physical computing devices that can execute on-premises cryptographic operations. The only way for an attacker to steal keys from an HSM is to physically be there where the HSM is located. In addition, they need to bypass the encryption algorithm used by the HSM to keep the keys secure. Which makes that too difficult.

  HSM has some disadvantages, one of them is the lack of transparency and also the high cost of updating and fixing vulnerabilities as Prof. Yehuda Lindell has mentioned in [37].

- Software key management: A software to handle storage of encryption keys, distributing and managing them. It provides protection and prevents data loss for keys. It has a couple of advantages over the HSM, like running in the cloud, has lower cost and giving full control of keys. It also can perform all functions done by an HSM.

  Still, if something fails, replication must be done by the user. Because servers must be installed and configured manually to execute key management, this solution is only viable for IaaS. Finally, Regulatory criteria that need FIPS-certified hardware are not met, as cited by [38].

- Peers can also store their shares in couchDB. As a result, accessing the shares will become easier since peers have easy access to it. CouchDB offers rich queries which are flexible and efficient. But the main disadvantages of using CouchDB are that all peers on a network must agree on using it and that only all the data must be in JSON format.

- File System: The simplest way is to store keys in local storage. The main disadvantage of this way is concurrent

accessibility and its limitation to data sharing. This will require more communication outside the system.

Communicating secret key shares

To support secret sharing, another important question needs to be answered, i.e. how to communicate shares (secret key parts) between peers during endorsement? Two simple ways are possible:

The first one is to communicate shares using an Inner-peer communication system. The invoking peer can communicate with other peers during endorsement to ask them to share their parts with him. This can be implemented and used as pluggable addition to Fabric, as done by [28], though they didn't show the implementation details.

The second option is to use the transient data API this will allow peers to send their shares directly to the chaincode, this can be enforced by altering the endorsement policy. The chaincode will then combine shares to recover the key and use it. This is similar to the trust model used by [29].

There are a lot of other possible ways when sharing the shares outside the Fabric network. For instance, physically or using some communication platform, all those ways have one main disadvantage in common, which is not using Fabric.

5 Results

This section presents the results of the research, its subsection provides the analysis done to test the performance when secret sharing is used.

The Shamir secret sharing algorithm used[2] is simple. But the idea of using it can be divided into two ways.

One is using secret sharing to share the secret between peers. This cannot be used easily in the case of Hyperledger Fabric, since all data are stored either on the ledger or in the state database. In both cases, all shares are still easily accessible to peers since all peers have a copy of the ledger and have direct access to the state database. By having direct access to all shares (parts) of a secret, it is easy to reconstruct the secret.

The other way is dividing the key instead of using this method for encryption and decryption. In other words, combining two cryptographic techniques so that one solves the other problem. In other words, when a peer has some data to add to the ledger then he encrypts it before adding it to the ledger using some encryption method. And in the case of using symmetric encryption, for instance, one key is used for both encryption and decryption. This key is then secretly shared between peers i.e. using secret sharing, see figure 6. As a result, no single peer can decrypt the data without other peers being notified or left out. This offers three improvements:

1- Decreasing the complexity of managing a secret key.

2- Increasing the security of the encrypted data, since no single peer can decrypt it or get the key to doing so.

3- Even though it is not a good idea to reuse keys, using secret sharing, keys can be used multiple times.

[2]A simple implementation of Shamir's Secret Sharing configured to use a finite field in $GF(2^8)$ with 128-bit padding: https://www.npmjs.com/package/shamirs-secret-sharing

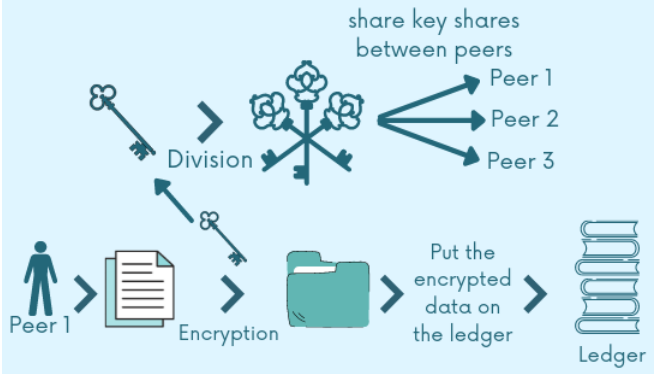

Figure 6: The end result of the system proposed.

A couple of limitations do still exist:

1- Multiple peers must work together to retrieve the decrypted data.

2- Key management is still important. Having the key divided into unusable shares (if not combined) doesn't mean that a peer can just share his key share with anyone or store it unsafely.

3- Extra runtime complexity since the secret key should be divided into shares.

4- What about the threshold, only one peer know it, i.e. the data owner. This means it is difficult for other peers to have access to the data without having all other peers involved in the transaction.

5- If a communication tool needs to be implemented. So that shares can be shared between peers instead of directly sharing them with the chaincode. Its limitations are still unknown since this need to be further studied.

5.1 Analysis

Every additional step to increase data privacy does also increase either the runtime, the space complexity or the cost of the software. Hence, adding Shamir secret sharing as explained before, the security of the proposed software hugely increases. But it also slows down the system, since every time the key reconstruction takes around 160 milliseconds on a middle-class machine which adds up to the encryption and decryption time, see figure 7.

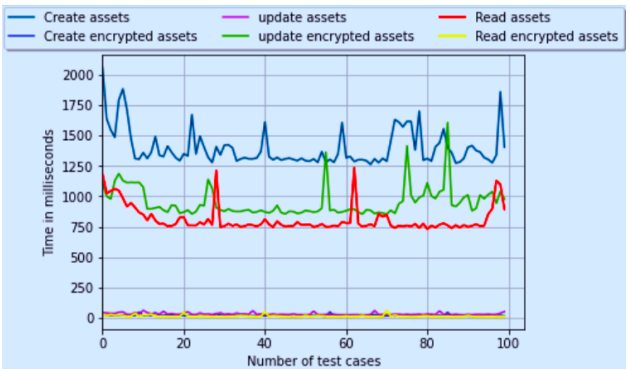

Figure 7: Performance analysis when using Shamir secret sharing.

Even though this addition increases the complexity of the code and slows it down, it is relatively cheap and easy to apply. Another thing that couldn't be tested or analysed is the communication tool to share key parts between peers. This tool couldn't be finalised because of the lack of time for this study. This will be further discussed in future work.

6 Discussion

Multiple studies have used secret sharing to implement systems such as Secure Multiparty Computation like [28] and [29]. But both studies didn't use secret sharing as a tool to protect the data by dividing it. This research also didn't implement this idea for a couple of reasons. Firstly, is to avoid the increase in needed storage, which might lead to the usage of external database services. Secondly, since every peer on the channel has access to the ledger, even if peers got different shares, they all need to store them on the ledger. This will not work without using private data collection so that every peer has access only to his share. But this will also increase the overhead.

Further in this section, some other possible scenarios, where security can be breached, will be discussed.

Storing shares

Despite the secrecy provided by private data collections, they should be utilised with caution because the metadata of secret information is much more than metadata and can be used to unlock the true private data, which is in this case the shares. Unauthorised peers on the same channel can examine the shared ledger in this attack scenario and see if private transactions happen regularly. In addition to that, if an unauthorised peer got compromised by an attacker, this attacker can easily recover the secret key using both shares, on the private data collection and the compromised peer.

Communicating shares

As early proposed in 4.3, a pluggable communication system can be implemented and used to share shares between peers. Since the system proposed is pluggable, a lot of vulnerabilities might breach and need further study. In addition to that complicated implementation can be viewed as an exaggerated drain on energy consumption when using Hyperledger Fabric. This might be further studied in future work.

As mentioned by Brotsis [16], Hyperledger Fabric's ecosystem is not post-quantum secure. This means, if information broadcasted over the network can be compromised, it can then be exposed to malicious decryption techniques to try decrypting it using a large number of quantum computers. Still, this remains to be seen in the (near) future, whether post-quantum digital signatures will be implemented by then.

7 Responsible Research

This section is divided into two parts. The first section discusses the reproducibility of the methodologies utilized in this study. The second section then goes through the research's integrity implications.

7.1 Reproducibility

The reproducibility of this study is crucial to ensure that it is robust and can be used for further studies in the future.

The research explained in this paper is done in such a way that it is easy for anyone who has a computer science background to reproduce it. In addition to that, all sources used in this paper are listed in the respective section. The steps needed to get the experiments running are all described and included with the code.

The running environment used to produce the results was Fedora 35, Intel(R) Core(TM) i7-7700HQ @ 2.80GHz and 16 GB of RAM. The latest version of Hyperledger Fabric (v2.4) and its test network has been used to experiment.

The test network uses separate docker containers for each peer and for the chaincode too. Those containers were all located on the same machine. The network had two organisations in all of the experiments, and each organisation contained two peers belonging to it.

The code for the experiments can be found on Github[3] on a public repository.

7.2 Integrity

Academic integrity at TU Delft is of high importance and is one of the six core values of TU Delft. TU Delft has created a course where it explains the essential topics which need to be considered when doing research [39].

Generally, integrity is accomplished by the application of high ethical and moral standards. In addition to that, researchers must commit to the policies of their faculty and university.

The code of this research commits to the TU Delft code of conduct and the Netherlands code of conduct for research integrity. It is developed from the code of other researchers and designed to offer more solutions for developers.

Furthermore, the experiments conducted in this study do not create any ethical or privacy problems. Because no real data is used in the trials or evaluations in this study, and no real-world data is obtained.

8 Conclusions

In conclusion, this research has studied the possibilities of improving the security and privacy of Hyperledger Fabric using the cryptographic method called Secret sharing. A brief comparison between secret sharing methods has been done to conclude that using Shamir's secret sharing is the most suitable for this enhancement. Two different ways of using secret sharing have been considered and the better option this research believes has been applied. This research (like others) proposed combining secret sharing with other encryption methods, like homomorphic encryption or in the case of this study experiment symmetric encryption. This strategy increases the security and safety if malevolent attackers obtain access to the encrypted data kept on the ledger, as well as their secret key in a Fabric network.

This paper result was to secretly share keys used to encrypt data stored on the ledger or state database, to decrease the

[3] https://github.com/alikahawa/fabric-samples

complexity of key management. Some possible ways to share the key parts have also been discussed and a proposal for a communication tool is set as future work.

The main advantages and disadvantages of this research are clearly stated in the results sections. For more information, please contact us and we hope to help you as much as we can.

8.1 Future Work

This research is only a small study in an enormous study field. Many features are currently being developed to improve the security of Hyperledger Fabric and other decentralised systems. That being said, a lot of issues are left to be further studied and verified in the future.

The main research work for the future will be finishing the communication tool mentioned earlier and investigating potential limitations and performance issues related to it. Also testing the use of Transient data API to communicate shares directly with the chaincode.

Other research is also needed to assess the performance implications in real-world circumstances, and investigate the limitations when using this approach in production-level smart contracts. As for data storage, IPFS [4] can be used in future experiments.

Most importantly, for this system to be more attractive performance must be improved. A deeper dive into improvements that increase the performance of this system and decrease both runtime and space complexity shall be further studied. Also the use of some advanced encryption like attribute-based encryption [40], [41] and [42].

Acknowledgment

The author would like to thank his supervisor and his peer group for their continuous feedback and suggestions.

References

[1] E. Androulaki, A. Barger, V. Bortnikov, *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[2] H. Insights., "Companies currently using hyperledger fabric. 2021," 2021. [Online]. Available: https://discovery.hgdata.com/product/hyperledger-fabric.

[3] M. Goddard, "The eu general data protection regulation (gdpr): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017.

[4] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, p. 3 152 676, 2017.

[5] A. Beimel, "Secret-sharing schemes: A survey," in *International conference on coding and cryptology*, Springer, 2011, pp. 11–46. DOI: https://doi-org.tudelft.idm.oclc.org/10.1007/978-3-642-20901-7_2. [Online]. Available: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-642-20901-7_2.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[7] C. Ferris, "Hyperledger achieves huge milestone: Introducing hyperledger fabric 2.0," *IET Cyber-Physical Systems: Theory & Applications*, 2020. [Online]. Available: https://www.ibm.com/blogs/blockchain/2020/01/hyperledger-achieves-huge-milestone-introducing-hyperledger-fabric-2-0/.

[8] H. Fabric, "Smart contracts and chaincode," 2021. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html.

[9] M. Krstić and L. Krstić, "Hyperledger frameworks with a special focus on hyperledger fabric," *Vojnotehnicki glasnik*, vol. 68, pp. 639–663, Jul. 2020. DOI: 10.5937/vojtehg68-26206.

[10] H. Fabric, "Ordering service," 2021. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html.

[11] ——, "Private data," 2021. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/private-data/private-data.html.

[12] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[13] G. R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*, IEEE Computer Society, 1979, pp. 313–313.

[14] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 72, no. 9, pp. 56–64, 1989.

[15] O. Farràs and C. Padró, "Ideal secret sharing schemes for useful multipartite access structures," in *International Conference on Coding and Cryptology*, Springer, 2011, pp. 99–108.

[16] S. Brotsis, N. Kolokotronis, K. Limniotis, G. Bendiab, and S. Shiaeles, "On the security and privacy of hyperledger fabric: Challenges and open issues," in *2020 IEEE World Congress on Services (SERVICES)*, IEEE, 2020, pp. 197–204.

[17] A. Dabholkar and V. Saraswat, "Ripping the fabric: Attacks and mitigations on hyperledger fabric," in *International Conference on Applications and Techniques in Information Security*, Springer, 2019, pp. 300–311.

[18] C. Paulsen, "Revisiting smart contract vulnerabilities in hyperledger fabric," 2021.

[19] H. Hasanova, U.-j. Baek, M.-g. Shin, K. Cho, and M.-S. Kim, "A survey on blockchain cybersecurity vulnerabilities and possible countermeasures," *International Journal of Network Management*, vol. 29, no. 2, e2060, 2019.

[20] B. Putz and G. Pernul, "Detecting blockchain security threats," in *2020 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2020, pp. 313–320.

[4] The InterPlanetary File System (IPFS) is a peer-to-peer distributed file system

[21] H. Fabric, "Frequently asked questions, security & access control," 2021. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.4/Fabric-FAQ.html.

[22] A. M. Mao, "Using smart and secret sharing for enhanced authorized access to medical data in blockchain," Ph.D. dissertation, Carleton University, 2020.

[23] E. Zhang, M. Li, S.-M. Yiu, J. Du, J.-Z. Zhu, and G.-G. Jin, "Fair hierarchical secret sharing scheme based on smart contract," *Information Sciences*, vol. 546, pp. 166–176, 2021.

[24] R. Tso, Z.-Y. Liu, and J.-H. Hsiao, "Distributed e-voting and e-bidding systems based on smart contract," *Electronics*, vol. 8, no. 4, p. 422, 2019.

[25] J. Lyu, Z. L. Jiang, X. Wang, Z. Nong, M. H. Au, and J. Fang, "A secure decentralized trustless e-voting system based on smart contract," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2019, pp. 570–577.

[26] S. Kyazhin and V. Popov, "Yet another e-voting scheme implemented using hyperledger fabric blockchain," in *International Conference on Computational Science and Its Applications*, Springer, 2020, pp. 37–47.

[27] J. Camenisch and E. Van Herreweghen, "Design and implementation of the idemix anonymous credential system," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 21–30.

[28] F. Benhamouda, S. Halevi, and T. Halevi, "Supporting private data on hyperledger fabric with secure multiparty computation," *IBM Journal of Research and Development*, vol. 63, no. 2/3, pp. 3–1, 2019.

[29] J. Zhou, Y. Feng, Z. Wang, and D. Guo, "Using secure multi-party computation to protect privacy on a permissioned blockchain," *Sensors*, vol. 21, no. 4, 2021, ISSN: 1424-8220. DOI: 10.3390/s21041540. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1540.

[30] J. Cha, S. K. Singh, T. W. Kim, and J. H. Park, "Blockchain-empowered cloud architecture based on secret sharing for smart city," *Journal of Information Security and Applications*, vol. 57, p. 102 686, 2021.

[31] R. Stefanov, "Enhancing the privacy and security of hyperledger fabric smart contracts using different encryption methods," 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:dbf548c7-849f-4aad-b4b7-455ba4a1835d.

[32] H. Fabric, "Using the fabric test network," 2021. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html.

[33] M. Ebrahim, S. Khan, and U. B. Khalid, "Symmetric algorithm survey: A comparative analysis," *arXiv preprint arXiv:1405.0398*, 2014.

[34] E. S. Atwal and U. Kumar, "A comparative analysis of different encryption algorithms: Rsa, aes, dss for data security," 2021.

[35] P. Mahajan and A. Sachdeva, "A study of encryption algorithms aes, des and rsa for security," *Global Journal of Computer Science and Technology*, 2013.

[36] L. Harn and H.-Y. Lin, "A cryptographic key generation scheme for multilevel data security," *Computers & Security*, vol. 9, no. 6, pp. 539–546, 1990.

[37] L. Yehuda, "Uncovering hardware security modules vulnerabilities," 2019. [Online]. Available: https://www.unboundsecurity.com/blog/major-vulnerabilities-in-hardware-security-modules/.

[38] E. consulting, "What is software key management?," [Online]. Available: https://www.encryptionconsulting.com/education-center/what-is-software-key-management/.

[39] T. Delft, "Scientific integrity," [Online]. Available: https://www.tudelft.nl/ethics/ethics/teaching-activities/courses-for-phd-students/scientific-integrity.

[40] Y. Chen, W. Li, F. Gao, *et al.*, "Efficient attribute-based data sharing scheme with hidden access structures," *The Computer Journal*, vol. 62, no. 12, pp. 1748–1760, 2019.

[41] J. Han, L. Chen, W. Susilo, X. Huang, A. Castiglione, and K. Liang, "Fine-grained information flow control using attributes," *Information Sciences*, vol. 484, pp. 167–182, 2019.

[42] P. Zhang, Z. Chen, J. K. Liu, K. Liang, and H. Liu, "An efficient access control scheme with outsourcing capability and attribute update for fog computing," *Future Generation Computer Systems*, vol. 78, pp. 753–762, 2018.

A Appendix A

To simplify the vision and purpose of this paper, the following example can be used as an application example. This example shall show the benefit of the Hyperledger Fabric if supported by a preserving privacy method securing the data on the ledger:

The National Road Traffic Agency in the Netherlands (RDW) wants to have their data on a joint ledger in a privacy-preserving manner because of the privacy law in the Netherlands. For instance, they cannot reveal information about a car owner while they need to reveal all information about that car itself. They also want smart contracts that can be used to change owners of cars, but that also need to be done in a privacy persevering manner so that no one can see the private information of owners. Using Hyperledger Fabric, When the RDW receives a transfer application, the data of both the old owner and the new owner of a car will be revealed to the departments of the RDW. This means all employees who have access to the ledger through their departments will be able to see the private data of both old and new owners. Fabric offers two ways to protect data, channels and private data, but they both have disadvantages [5]. Using both Hyperledger Fabric and cryptographic technologies will result in making the procedure of transferring car's data between RDW's departments faster and at the same time ensuring the privacy of the clients by encrypting the data before storing it on the ledger.

B Appendix B

This appendix will add some elements that might be needed to gain a better understanding of Hyperledger Fabric.

Execute-order-validate transaction

Execute-order-validate, figure 8, transaction workflow includes the following three phases: Execute, a transaction is simulated and endorsed. Read and write sets are created and endorsements are collected. Order will order read and write sets in addition to atomic broadcasting (consensus). This step also contains a stateless ordering service. During the validation step, endorsements, read and write sets are validated and invalid and conflicting transactions are eliminated. When all those steps finish, the state is persisted on all peers.

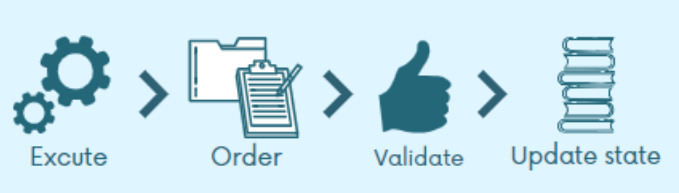

Figure 8: the architecture of Execute-order-validate of Hyperledger Fabric.