# Towards achieving confidentiality in Hyperledger Fabric

1st Benedikt Hofmann
Cybersecurity Technology
Siemens AG
Munich, Germany
hofmann.benedikt@siemens.com

2nd Prabhakaran Kasinathan
Cybersecurity Technology
Siemens AG
Munich, Germany
prabhakaran.kasinathan@siemens.com

3rd Martin Wimmer
Cybersecurity Technology
Siemens AG
Munich, Germany
martin.wimmer@siemens.com

Abstract—Enterprise blockchain use cases have confidentiality requirements when executing cross-organizational business processes or workflows across their corporate boundaries. For instance, only a certain privileged subset of organizations may know of the workflow's existence and is allowed to access sensitive workflow's data assets such as price information. A private or permissioned blockchain such as Hyperledger Fabric (HLF) provides many features to support those enterprise use cases that require confidentiality. Nevertheless, the blockchain architects need to decide when it comes to how to use them, when to use them, and what are the performance and security implications of those design decisions. This work presents a generic methodology to design a secure Hyperledger Fabric (HLF) blockchain architecture with a given security requirements. To achieve that, we developed and used a novel formal description to specify a generic multilateral workflow in the context of an enterprise use case. Furthermore, the security of the resulting architecture is evaluated against an attacker model and relevant countermeasures are presented. In addition, the performance impact of proposed changes is evaluated using the Hyperledger Caliper benchmarking framework.

## I. Introduction

Manufacturing of specialized products most often involves a complex supply chain spanning across multiple organizations and borders. Every organization has its own business processes, but to work with each other they agree on certain conditions or processes, which we refer to as workflows. These cross-organizational workflows are not orchestrated by a trusted authority. Hence, the individual steps of the workflow are typically executed on side of organizations and the resulting data or assets need to be exchanged out-of-band. Thus, these cross-organizational business processes or workflows are orchestrated in a decentralized fashion.

Let us take an example enterprise use case presented in [1], that involves cooperation of four organizations to fabricate a product. 1. The Owner/Customer orders a cabinet to operate it in a power plant. 2. The notification body (NOBO) certifies the correct production and conformity of the produced cabinet. 3. The engineering, procurement, and construction contractor (EPC) manages the construction of the cabinet. 4. The Supplier builds a secondary component for the cabinet. The trust relationships between the organizations are not always mutual and may change over time. The Owner wants the EPC to produce a cabinet that satisfies local regulations and therefore requires NOBO to certify the correct production. No mutually accepted source of truth exists between the organizations and each organization hosts its own data. This may give rise to an inconsistent view of the business process by each organization, and it does not facilitate the audit through notary organizations such as governmental or certification authorities.

A distributed solution is required to implement a mutually accepted source of truth with an immutable ledger while maintaining access control over confidential data assets and workflows. It must be hosted by multiple organizations - multilateral - with shared responsibility and control over workflows and data assets - distributed - between the responsible organizations. The immutable audit log supported by the decentralized consensus algorithm increases the credibility and efficiency of audits by supervisory bodies. This means some organizations observe the workflow while others participate in workflows. Each organization requires the same view of the state of the workflow, which is why a transparent single source of truth is required.

Consequently, blockchain seems to be an optimal fit for such use cases as it eliminates the need for a trusted third party for orchestrating and enforcing workflows between organizations. It enables trusted transactions among untrusted participants in a network. Thus, blockchain became one the emerging technology to implement and enforce distributed workflow enforcement use cases [2], [3]. Smart contracts are distributed programs used to orchestrate the distributed multilateral workflow which modify the blockchain state. In a traditional public blockchain the decentralized trust is established when most of the parties involved in the blockchain network validate a smart contract's logic which we refer to as computational integrity. This means, most of the parties re-execute or should be able to reproduce the results i.e., they should have access to the input data as well as the program implementing the smart contract logic. Thus, the computational integrity of smart contracts is enforced without a central authority through a consensus protocol which ensures that only valid updates are made to the blockchain.

This goes against the requirements of many enterprises use cases which demand some level of confidentiality e.g., the Owner does not want to reveal the sensitive input information such as pricing data, or with which organizations it cooperates with, or even the smart contract program logic itself. In conclusion, multiple organizations need to collaborate and therefore need to share data assets and confidential data assets on a need-to-know basis with each other. Additionally, they must ensure that neither organization can tamper with audit logs successfully without others detecting misuse. These problems can be solved by a permissioned or private blockchain (like HLF). The participants of HLF network can select: a) the organizations they share the smart contract logic with (HLF chaincode lifecycle), b) with whom they share metadata (HLF channels), c) within a channel with whom they share sensitive data (HLF private data collection (PDC)), d) finally, which organizations must validate the smart contracts (HLF endorsement policies).

For more detail, we recommend referring to the respective blockchain documentation like HLF [4]. Even though, a concrete blockchain product may come with these features to solve the confidentiality concerns, how to use them, when to use them

and what security implications they have are a concern for blockchain architects. In this paper, we explore these features in detail and apply them to solve the enterprise fabrication use case. On this aspect, this paper contributes:

1) A general, formal approach is presented to describe a distributed multilateral workflow.
2) A methodology that uses the formal approach and produces a secure permissioned blockchain (in our example HLF) architecture as a solution.
3) Attacker models are defined, and the solution is evaluated against potential attacks.
4) Countermeasures against the defined attacks and their performance impact is measured (using the Hyperledger Caliper tool).
5) By means of a real-life use case this approach is demonstrated and evaluated.

The rest of the paper is organized as follows: Section II is the main contribution. It includes the definition of the requirements as a novel formal description, and the conversion of that input into a solution. This is done for general workflows and a concrete use case - the fabrication stage. Section III introduces the attacker model, applies it to the solution and defines countermeasures. Section IV evaluates the performance impact of the countermeasures. Section V highlights related research and Section VI summarizes all findings and opens questions to be addressed in future works.

## II. Proposed Solution

The process of constructing a solution based on a use case requires structured requirements as inputs. This input is defined in Section II-A. The Hyperledger Fabric (HLF) related components and terms used are described in Section II-B. Then, the approach is defined and shown for a general workflow in Section II-C. Next, the example use case - fabrication stage - is introduced in Section II-D and the solution for it is created in Section II-E. This demonstrates that the process can be done programmatically by a software component for most architectural decisions.

### A. Formal Description

A process to create blockchain solutions for distributed multilateral workflows needs structured requirements about confidentiality and access control. We propose a formal description to structure these requirements. Prior descriptions for multilateral workflows such as a coloured petri net (CPN) [5] or yet another workflow language (YAWL) [6] don't model our use case appropriately well. The presented description encompasses all necessary data and serves as the only input needed to create solutions. In general, a multilateral distributed workflow consists of the following: $O$ is the set of all organizations that participate in the workflow, and $|O| = n$ is the number of organizations. A multilateral distributed workflow is a collection of one or more sub workflows $w_i \in W$ where $W$ is the set of all sub workflows. Each sub workflow $w_i$ has a set of properties $w_i = (P_i, OBS_i, S_i, D_i, C_i)$. $P_i \subseteq O$ is the set of organizations that participate in the sub workflow called participants. $OBS_i \subseteq O$ is the set of observing organizations called observers. They are allowed to observe the non confidential data assets and the transactions (metadata) of the sub workflow. $P_i \subseteq OBS_i$ every participant is also an observer. $S_i$ is the set of workflow steps $s_{ij} \in S_i \in w_i$. Each step $s_{ij} \in S_i$ can be executed by one or a group of participants called the $EXECUTORS_{s_{ij}}$. $D_i$ and $C_i$ are the sets of data assets of the sub workflow $w_i$. The data assets can be classified into either confidential data assets $c_{ij} \in C_i$ or not confidential data assets $d_{ij} \in D_i$. Every data asset is created by one or a group of organizations called writers and is sent to one or a group of organizations called readers. A non-confidential data $d_{ij}$ asset has a specific set of $WRITERS_{ij} \subseteq P_i$ and $READERS_{ij} = OBS_i$. A confidential data asset $c_{ij}$ has a specific set of $WRITERS_{ij} \subseteq P_i$ and a specific set of $READERS_{ij} \subseteq P_i$. Also, every writer of a data asset must also be a reader. Importantly, reader is a role that an observer can have to read a specific data asset.

### B. HLF specific terms

Hyperledger Fabric (HLF) is a private permissioned blockchain product which we utilize. In HLF smart contracts are executed in a different transaction flow than in other blockchains. This transaction flow is called execute-order-validate. The process starts with the execute phase: Clients, third party programs interacting with the blockchain, are proposing transactions to peers. Peers are the nodes which execute smart contracts and are run by different organizations. Peers execute the proposed smart contract and return the signed result of the execution to the client called the endorsement response. The client sends the packaged responses to the ordering service, called transaction message. The ordering service may consist of multiple nodes called the ordering service nodes (OSN). The service collects multiple transaction messages, packs them into blocks, and sends these blocks to the peers. This is the ordering phase. Peers validate the transactions in the blocks and apply the changes from the transactions to their local copy of the so called world state. This is the validating phase. The world state is a versioned key value store. Multiple smart contracts can be deployed as a single chaincode. Chaincodes are basically the entity which is deployed onto channels and installed on peers. Smart contracts are essentially only the business logic. For more details we refer to [4].

### C. General Solution

1) $w_i$: Each sub workflow is implemented as a single smart contract. The endorsement policy requires all participants $P_i$ signatures.
2) $OBS_i$: Each distinct set of observers $OBS$ requires the creation of a new channel to separate the sets of observers from each other. The sub workflows' smart contracts are deployed to the channels where the set of observers of the sub workflow matches the members of the channel. Henceforth, channel observers, channel members and observers of the sub workflow and smart contracts can be used interchangeably.
3) $P_i$: Each participant $p$ of the sub workflow $w_i$ has the corresponding smart contract installed on its peers.
4) $S_i$: Each workflow step $s$ of the sub workflow $w_i$ is implemented with its own smart contract function.
5) $EXECUTORS_{s_{ij}}$: The smart contract functions which realize a step $s_{ij}$ must enforce that they are only executed by the authorized set of $EXECUTORS_{s_{ij}}$. The certificate of the transactor which is a part of the transaction proposal is used for this purpose [7].

6) Writers: The smart contract functions must authorize the transactors to write data assets. It must check if the transactor is a writer of that specific data asset.
7) $d_{ij}$: Normal data assets are stored on the world state by the smart contract of the respective sub workflow.
8) $c_{ij}$: Confidential data assets are stored in a private data collection (PDC). These PDCs can only be stored by the set of $READERS_{c_{ij}}$ of the confidential data asset. This is enforced through the collection storage policy. The write access is enforced like step 6. The resulting collection level endorsement policy is the set to $READERS_{c_{ij}}$.

The ordering service in HLF uses the RAFT [8] consensus algorithm [7] which is crash fault tolerant (CFT). Currently, HLF does not support a Byzantine-fault tolerant (BFT) [9] consensus algorithm. MirBFT [10] and FastBFT [11] are in development and might be used in the future. Raft does not provide protection against a malicious organization and the general solution will change whenever HLF implements a BFT consensus algorithm. Hence, spreading governance of the RAFT ordering service distributes the responsibility for its availability. Most nodes need to function correctly to order new blocks. An ordering service nodes (OSN) that takes part in ordering process is called a consenter. Given the number of consenters $N$. Then the ordering service is not functional only if the number of simultaneous consenter failures is greater than $F(N)$.

$$F(N) = \begin{cases} N/2 & \text{if } N \text{ is odd} \\ (N/2) - 1 & \text{if } N \text{ is even} \end{cases}$$

An odd number of consenters is better because $F(4) = 4/2 - 1 = F(3) = 3/2 = 1$. Meaning the fourth consenter results in an additional consenter which potentially might become unavailable. An odd number of consenters is preferred for all created channels. It is important to only allow organizations that are at least observers to run an OSN because it gives them read access to data assets and metadata of private data collections PDC.

### D. An example use case: Fabrication Stage

To illustrate we summarize a use case of a concrete workflow between 4 organizations. For further details we refer to [12]. There are 4 organizations: The Owner orders a cabinet to operate in a power plant. NOBO certifies the cabinet's correct production and conformity. EPC manages construction of the cabinet. The Supplier builds a secondary component, a pressure sensor in this case. The resulting set of organizations is $O$ ={Owner, NOBO, EPC, Supplier} and the number of organizations is $n = |O| = 4$. In short, the workflow orchestrates the ordering, construction, and certification of a cabinet. The different parts and used data assets are illustrated in Fig. 1 for each sub workflow. For instance, the audit_report contains the certification of NOBO of the cabinet is a part of $w_1$ and $w_3$. Both sub workflows work on the same data asset but they each govern over their own copy independently. The endorsement policies and the implementation of cross-chaincode invocations require this. Section II-E will discuss the reasons in detail. Each workflow step $s_{ij}$ has a set of authorized $EXECUTORS_{s_{ij}}$, the mapping of workflow steps to executors is defined in [12]. For instance, the step order_cabinet has the executor Owner. Table I defines the roles each organization holds for each sub workflow.

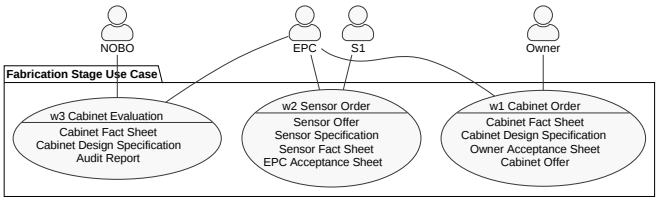


Fig. 1: Use case diagram for the fabrication stage showing the 3 separate sub workflows. The participants and data assets according to Section II-A are depicted.

| Workflows \ Org. | EPC | NOBO | Owner | Supplier |
|---|---|---|---|---|
| $w_1$ Cabinet Order | OBS\|P | OBS | OBS\|P | |
| $w_2$ Sensor Order | OBS\|P | OBS | | OBS\|P |
| $w_3$ Cabinet Evaluation | OBS\|P | OBS\|P | OBS | |

TABLE I: (Obs) = observers, (P) = participants. Two distinct sets of observers: First set: EPC, NOBO, and Owner. Second set: EPC, NOBO, and Supplier.

Write and read access are defined for each data asset and confidential data asset. For instance, an offer for the cabinet is sent by the Owner to the EPC, in the send_cabinet_offer step. The contents are confidential as they contain information regarding price and delivery date. Therefore, this data asset is considered confidential and has the readers Owner and EPC. The Owner is the writer because it makes the offer.

### E. Solution for the fabrication stage use case

The general approach is now applied to the example use case - fabrication stage. The resulting design is called default configuration.

1) The three sub workflows $w_1, w_2, w_3$ are implemented as three smart contracts each in their own chaincode. The separation into chaincodes isolates the world states of each sub workflow. These smart contracts are called Cabinet contract, Sensor contract, and Evaluation contract. The contracts have their own endorsement policies, shown in Table. II. These endorsement policies enforce that a peer each from every sub workflow participant $P_i \in w_i$ must endorse the transactions of the implementing smart contract. For instance, $w_1$ Cabinet Order is implemented by the Cabinet contract. The participants of $w_1$ are EPC and Owner. The endorsement policy requires an endorsement each from their peers.

| Contract | Sub Workflow | Endorsement Policies |
|---|---|---|
| Cabinet | $w_1$ | AND("EPC.peer", "Owner.peer") |
| Sensor | $w_2$ | AND("EPC.peer", "S1.peer") |
| Evaluation | $w_3$ | AND("EPC.peer", "NOBO.peer") |

TABLE II: Endorsement policies of the three smart contracts.

2) There are two distinct sets of observers $OBS$ as stated in Table I. $OBS_1 = OBS_3$ this set requires the creation of the Cabinet channel. $OBS_2$ requires the second channel, the Sensor channel. The channel participation is illustrated by Table III. The Cabinet and Evaluation contracts are deployed on the Cabinet channel. The Sensor contract are deployed on the Sensor channel. The smart contract deployment is shown by Table IV. The chaincode lifecycle [12] enables the independent deployment of chaincodes by multiple subsets of channel members. This enables confidentiality of sub workflows while enabling observers to audit and verify the results of these workflows. For instance, the Lifecycle Endorsement policy of the Cabinet channel allows the deployment by two subsets. The

| Channel \ Member | EPC | Owner | NOBO | Supplier |
|---|---|---|---|---|
| Cabinet channel | ✓ | ✓ | ✓ | |
| Sensor channel | ✓ | | ✓ | ✓ |

TABLE III: Mapping of channels to the channel members.

| Channel \ Contract | Cabinet | Sensor | Evaluation |
|---|---|---|---|
| Cabinet channel | ✓ | | ✓ |
| Sensor channel | | ✓ | |

TABLE IV: Mapping of channels to deployed smart contracts.

subset of EPC and Owner cooperate to deploy the Cabinet contract. The subset of EPC and NOBO cooperate to deploy the Evaluation contract. Hence, the Evaluation and Cabinet contract can be approved and deployed individually by their participants. In brief, the channel separation enables that Owner and Supplier do not know of each other's transaction with the EPC. This possibility is important, because a forced disclosure of business relations can lead to the EPC not wanting to participate.

3) Chaincode installations are determined by the organizations that are participants $p_{ij} \in P_i$ of the sub workflow $w_i$. For instance, the Cabinet contract is deployed on peers of the EPC and Owner because both are participants of $w_1$.

4) The set of workflow steps $s_{ij} \in S_i$ of each sub workflows are implemented as functions of the smart contracts. For instance, the Cabinet contract has a function send_cabinet_offer. It is used by the EPC to send the cabinet_offer to the Owner. The complete list of steps can be found in [12].

5) Every workflow step $s_{ij}$ has a set of executors $EXECUTORS_{s_{ij}}$ which define who can execute which transaction. For instance, the send_cabinet_offer function has the executor EPC. Hence, upon invoking the function the signature of the transaction proposal is checked against the root certificate of the EPC organization. The root certificate is stored in the membership service provider definition that is contained in the channel configuration. Only upon correct validation does the peer endorse the transaction.

6) The set of writers/readers are implemented like the executors. Workflow steps writing data assets check in the authorization whether the transactor is also a writer/reader of the data assets. For instance, get_cabinet_offer function checks whether the transactor is the EPC or the Owner, because both are readers.

7) Non confidential data assets $D_i$ of a sub workflow $w_i$ are stored on the world state. For instance, the audit_report which represents the certification of NOBO is such a data asset. The audit report is used in two sub workflows $w_1$ and $w_3$ which leads to redundancies. Both contracts Cabinet and Evaluation need to store the asset separately. HLF presents opportunities to remedy this redundancy and let chaincodes call each other, called cross-chaincode invocations [7]. Using cross-chaincode invocation assets of chaincode A can be accessed by chaincode B. But both chaincodes (A and B) must be installed on the same peer. But we want to make sub workflows available on a need-to-know basis. And we cannot deploy them independent from each other if the need to call each other using cross-chaincode invocation. It is also possible to combine multiple smart contracts into a single chaincode to enable each contract to directly access the data assets of the other contract. This means we need to deploy both smart contracts at the same

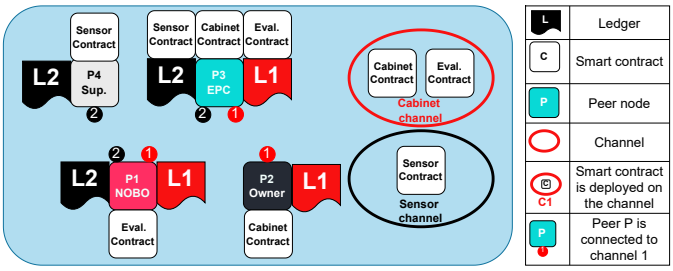


Fig. 2: HLF deployment diagram. Sup=Supplier, Eval=Evaluation.

time. Similarly, deploying multiple smart contracts into one chaincode also removes our ability to deploy them separately. Redundancies are utilized to decouple the smart contracts.

8) Two sub workflows contain confidential data assets $w_1$ and $w_2$. These assets are cabinet_offer and pressure_sensor_offer. The cabinet_offer, must only be shared with its $READERS_{c_{\text{cabinet\_offer}}}$ EPC and Owner. Hence, a PDC is defined with the name PDC1 between both organizations and cabinet_offer is stored there. Now, NOBO cannot read the contents of cabinet_offer even though it is part of the Cabinet channel. But NOBO can verify the integrity of the cabinet_offer with the hash on the channel's ledger if the Owner/EPC provide the asset. Likewise, pressure_sensor_offer is shared between the EPC and the Supplier and is stored in another PDC with the name PDC2.

| Private Data Collection | Endorsement Policies |
|---|---|
| PDC1 | AND("EPC.peer", "Owner.peer") |
| PDC2 | AND("EPC.peer", "S1.peer") |

TABLE V: The collection level endorsement policies define organizations that endorse transactions with private data. Whenever the PDC1 is written in a transaction then the EPC's and Owner's peer need to endorse.

In summary, Fig. 2 shows the condensed overview of all channels, smart contracts, peers, organizations, and chaincode deployment. The configuration of the ordering service of the Cabinet channel depends on the channel members Owner, EPC, and NOBO. Each organization contributes one consenter, which sums up to a total number of 3 consenters, a set which is an uneven amount. The modification of the channel configuration parameters is governed by the so-called modification policies. These values which concern all channel members are governed such that an admin of each member must approve changes. The resulting policy is AND(EPC.admin, NOBO.admin, Owner.admin). Fig. 3 illustrates the ordering service and the governing organizations of the Cabinet channel. The Sensor channel can be set up accordingly. The consenters are run by the EPC, the Supplier, and NOBO. The modification policy of the channel configuration is AND(EPC.admin, Supplier.admin).

## III. Attacker Model

Multiple different attacks are possible on the presented solution in Section II. Yet, we propose a novel attacker model that takes the blockchain network architecture into account. We assume that the attack controls components within the blockchain network. The model shows the impact of the architectural decisions. HLF the components that are valuable for the attacker are client, orderer, certificate authority, and peer. A brief description of the role of these components is

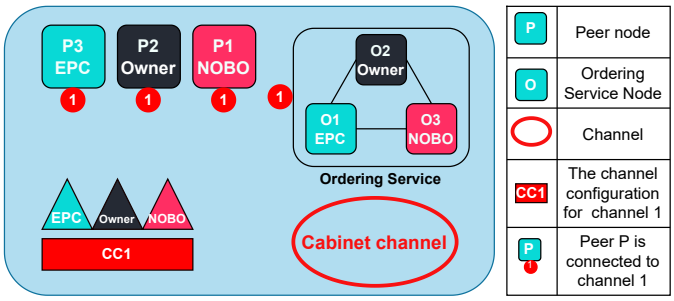


Fig. 3: Configuration for the ordering service and governing organizations for the Cabinet channel.

given in Section I, more details in [12]. An attacker controlling a client can query and submit transactions which is limited and therefore not discussed. A controlled consenter of the ordering service can access the ledger and take an active part in the consensus algorithm. However, the consensus protocol is RAFT which is crash fault tolerant (CFT) and can therefore not withstand malicious actors. This kind of attack requires a Byzantine-fault tolerant (BFT) consensus algorithm which is so far not supported by HLF [13]. Compromises of certificate authorities are called Identity Provider Compromise and are already investigated in [14].

An attacker controlling peers can endorse transactions, has access to the smart contract code, the ledger, and the private data collections PDC, can refuse to execute transaction proposals, or send malicious proposal responses. We call this attacker type the Peer Attacker. The Peer Attacker can communicate between the malicious peers that they control. For instance, they can orchestrate 2 peers to send malicious proposal responses. The endorsed transaction is not necessarily the result of a smart contract execution and arbitrary changes can be made to the world state and private data collections (PDC) as a result. This paper considers 2 scenarios: Blocking of smart contracts and manipulation of the smart contract execution.

### A. Evaluation of the Peer Attacker

The Peer Attackers influence depends on the amount and kind of peers under its control. This influence is evaluated for both scenarios in two separate tables. Table VI illustrates the influence over the blocking of smart contracts. The left side of the tables shows which peers the attacker controls. The right side shows if the attacker can successfully block a smart contract. In scenarios 1, 3, and 4 the Peer Attacker can block one contract at a time and only one sub workflow is affected. In scenario 2, the EPC's peer is controlled, and all contracts can be blocked. The EPC's peer is a single point of failure regarding the blocking of smart contracts by the Peer Attacker. The influence of the Peer Attacker is the same in scenario 2 and 9 even though the total number of controlled peers is different. Only the EPC's peer is controlled in scenario 2 and all other peers are controlled in scenario 9. In brief, controlling the EPC's peer is as valuable as controlling all other peers in this scenario.

Table VII shows the influence over the manipulation of smart contracts. The right side shows if the attacker can endorse arbitrary transactions for specific contracts with its peers such that the endorsement policy is satisfied. For instance, the Cabinet contract requires two endorsements, one from

TABLE VI: Scenario: Blocking of smart contracts.

| Nr. | Controlled peers | | | | Blocked contracts | | |
|---|---|---|---|---|---|---|---|
| | NOBO | EPC | Owner | Supplier | Cabinet | Sensor | Evaluation |
| 1 | ✓ | | | | | | ✓ |
| 2 | | ✓ | | | ✓ | ✓ | ✓ |
| 3 | | | ✓ | | ✓ | | |
| 4 | | | | ✓ | | ✓ | |
| 5 | ✓ | | ✓ | | ✓ | | ✓ |
| 6 | ✓ | | | ✓ | | ✓ | ✓ |
| 7 | | | ✓ | ✓ | ✓ | ✓ | |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 9 | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |

TABLE VII: Scenario: Manipulation of smart contracts.

| Nr. | Controlled peers | | | | Contracts manipulated | | |
|---|---|---|---|---|---|---|---|
| | NOBO | EPC | Owner | Supplier | Cabinet | Sensor | Evaluation |
| 1 | ✓ | | | | | | |
| 2 | | ✓ | | | | | |
| 3 | | | ✓ | | | | |
| 4 | | | | ✓ | | | |
| 5 | ✓ | ✓ | | | | | ✓ |
| 6 | | ✓ | ✓ | | ✓ | | |
| 7 | | ✓ | | ✓ | | ✓ | |
| 8 | ✓ | | ✓ | ✓ | | | |
| 9 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

the EPC's peer and one from the Owner's peer. The Peer Attacker that controls both peers can send 2 manipulated proposal responses which can produce arbitrary results. Hence, the attacker can manipulate the Cabinet contract in scenario 6 and 9. In the scenarios 1 to 4 the Peer Attacker controls one peer each and this does not enable the attacker to manipulate any smart contracts because a minimum of two endorsements is required. In scenarios 5, 6, and 7 two peers are controlled at a time. The set of compromised peers contains the EPC's peer and another organization's peer. All these scenarios lead to exactly one smart contract which can be manipulated. Scenario 8 shows that if all peers are compromised, but the EPC remains unaffected then no contract can be manipulated.

The evaluation of the Peer Attacker shows the interaction of endorsement policies, chaincode deployment, and number of peers:

1) Blocked contracts: Table VI shows that even attacks on a single peer can lead blocked smart contracts. We recommend multiple peers for each organization to avoid a single point of failure. For instance, if the EPC has 6 peers authorized to endorse the Cabinet contract, then the attacker needs to compromise all peers to block the execution. Hence, the more peers' organizations host the harder it is for the Peer Attacker to block smart contracts.

2) Manipulated contracts: Table VII shows that if the set of peers attacked is congruent with the endorsement policy of a smart contract, then the attacker can manipulate that smart contract. Consequently, endorsement policies that require more peers to endorse lead to smart contracts which are harder to manipulate, because the Peer Attacker must compromise more peers to manipulate the smart contact. The applied endorsement policy can change when a PDC is used which must be considered. Basically, each possible endorsement policy must be analysed separately. If this endorsement policy is less restrictive than the normal smart contract endorsement policy than it may be easier for the Peer Attacker to manipulate the smart contract.

### B. Countermeasures for the Peer Attacker

The practical side of the countermeasures is investigated first. Then a model is presented in Section III-C that can estimate the chance of these attacks based on a blockchain network architecture. The results are then applied and the secured configuration for the fabrication stage's solution is created in Section III-D.

When adding more peers, the following must be considered: Adding more peers for an organization helps to protect against the blocking of smart contracts. The attack surface for attackers looking to manipulate smart contracts also increases with the addition of new peers. More peers are harder to keep in synch and a less homogeneous view of the blockchain may be the result. Hence, the chance of multi-version concurrency check (MVCC) collisions rises. These collisions occur if the endorsement response is computed on an outdated version of the world state. The resulting transaction will then fail the validation by other peers if a value it relied on has already been overwritten.

When an increasing number of endorsements is required, the following must be considered: Requiring more endorsements increases the effort for the Peer Attacker when trying to manipulate smart contracts. Increasing the number of required endorsements also increases the attack surface for the attacker looking to block smart contracts. Increasing the number of endorsements required for smart contracts is not trivial. The endorsements are defined according to the roles of each organization in the respective workflows. Hence, the endorsement policy must require more endorsements without involving more actual organizations. The following possibilities exist in HLF:

1) Adding another channel member: It is possible to add organization to the endorsement policies that are only observers of the workflow. This violates the need-to-know requirement about smart contracts. Hence, this cannot be employed.
2) Using the organisational unit (OU): In HLF each organization can define so called organisational unit (OU). It is possible to give each of these OU's different permissions. For instance, the default implementation knows 4 different organization units: client, admin, orderer, and peer. The membership service provider (MSP) maps certificates (default) to organizations and OUs [4]. Their name suggests also what they are intended to be used for. The peer OU is normally used to define endorsement policies, and the clients or admins cannot create endorsements. It is possible use these within an endorsement policy: AND(EPC.peer, EPC.client, EPC.admin, EPC.orderer) A better way would be to define new organization units like: peer1, peer2 etc. Yet this requires a change the MSP implementation and is therefore out of scope.
3) Using multiple MSPs for one organization: This option requires multiple MSP entries for one organization in the channel configuration. This option is supported by the current codebase and is therefore chosen. For instance, the Cabinet contract could require 4 endorsements for sufficient protection. Then the endorsement policy can be defined as follows: AND(EPC1.peer, EPC2.peer, Owner1.peer, Owner2.peer).

### C. Probability Model

We are modelling the probability for each attack scenario depending on the blockchain network architecture. The model has the following inputs:

1) N: Total number of organizations that need to endorse a smart contract.
2) m: Number of endorsing peers each organizations hosts. Every organization hosts the same number of peers in the model to retain usability and reduce complexity.
3) $p_c$: The probability that a peer is compromised by an attacker at any time. This probability is static and independent of the organization and the number of peers. The assumption is that every organization protects its peers in the same way.
4) $MAX_{p_b}$: Security requirement defining the maximum acceptable probability for a blocked smart contract.
5) $MAX_{p_m}$: Security requirement defining the maximum acceptable probability for a manipulated smart contract.

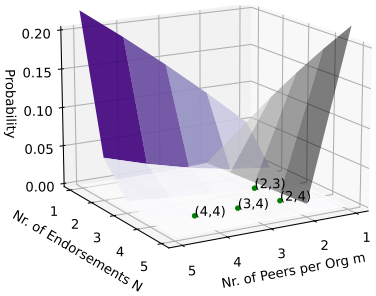


Fig. 4: The acceptable solutions are calculated for $p_c = 0.05$, and a maximum probability for blocked or manipulated smart contracts of $MAX_{p_b} = MAX_{p_m} = 0.001$. The purple surface shows the behavior of $p_m$ and the grey surface the behavior of $p_b$. The green dots with tuples illustrate the configurations which satisfy the security requirements. The tuple $(2, 3)$ defines that a configuration with 2 peers per organization and 3 endorsements satisfies the requirements.

The probabilities can be calculated as follows: The probability that a smart contract can be blocked is called $p_b = 1-(1-p_c^m)^N$. Hence, the following must hold true: $MAXp_b > 1-(1-p_c^m)^N$. The probability that a smart contract can be manipulated is called $p_m = (1-(1-p_c)^m)^N$. The following must hold true: $MAX_{p_m} > (1-(1-p_c)^m)^N$. Both requirements combined and reduced results in the following statement which must hold true for both security requirements to be satisfied: $\frac{ln(MAX_{p_m})}{ln(1-(1-p_c^m)} < N < \frac{ln(1-MAXp_b)}{ln(1-p_c^m)}$

The model can now be applied to the default configuration of the fabrication stage. The security requirements we set up are:

1) $MAX_{p_b}$: The maximum acceptable probability for a blocked contract is $MAX_{p_b} = 0.001$
2) $MAX_{p_m}$: The maximum acceptable probability for a manipulated contract is $MAX_{p_m} = 0.001$

The requirements $MAX_{p_m}$ and $MAX_{p_b}$ can be a part of a service-level agreement (SLA) like e.g., 99.999% availability. The value for $p_c$ must be set based on past experiences or comparable deployments. In this use case we assume that the probability for any one peer to be controlled by the attacker to be $p_c = 0.05$. The calculation that is visualized in Fig. 4. The figure illustrates, that increasing $m$, the number of peers per organization, has a positive effect on $p_b$ and a negative effect on $p_m$. Also, it shows that increasing the number of endorsements $N$ has a negative effect on $p_b$ and a positive on $p_m$. A balance must be struck to satisfy the security requirements. It shows that $m = 2$ peers and $N = 3$ endorsements is a secure configuration. The example in Fig. 4 shows this because the acceptable configurations are found where at a combination of an increased $m$ and increased $N$ and not at their respective extreme values.

### D. Secured configuration

To be compliant with the security recommendations calculated in the previous section the blockchain network architecture must change. The default solution uses 2 endorsements

| Contract/PDC | Endorsement Policies |
|---|---|
| Cabinet | AND("EPC1.peer", "Owner.peer", "EPC2.peer") |
| PDC1 | AND("EPC1.peer", "Owner.peer", "EPC2.peer") |
| Sensor | AND("EPC1.peer", "S1.peer", "EPC2.peer") |
| PDC2 | AND("EPC1.peer", "S1.peer", "EPC2.peer") |
| Evaluation | AND("EPC1.peer", "NOBO.peer", "EPC2.peer") |

TABLE VIII: Endorsement policies complying with security recommendations.

and 1 peer per organization. The recommendation requires 3 endorsements and 2 peers for all smart contracts. Hence, each organization must add 1 peer to their existing deployment. Yet, adding another endorsement is less trivial because only one organization needs to add another MSP. For instance, the Evaluation contract is currently endorsed by the EPC's and NOBO's peer. To reach a total amount of 3 endorsements, either NOBO or EPC must add another MSP to the channel. Because the EPC is part of every contract its efficient to choose the EPC add another MSP for the third endorsement to all other contracts. This keeps the needed number of nodes small. The resulting endorsement policies are shown in Table VIII. The newly added MSP EPC2 must also host 2 peers and have the same contracts deployed to at so it matches the configuration of EPC1's peers. It may happen that all contracts themselves are compliant with the recommendations however the private data collection (PDC) may not be. Then organizations in the collection level endorsement policy of the PDC must add further MSPs to comply with the security recommendation.

## IV. Performance Evaluation

The performance evaluation goals are twofold. First, assert that the performance is sufficient for the use case. A study from [15] concluded the maximum wait times users are willing to tolerate for web applications to be about two seconds. Hence, the requirement to transaction latency is 2 seconds. For our use case a transaction throughput of more than 20 TPS is sufficient. Second, the performance impact of the applied security measures in Section III-B will be estimated. The default configuration (Section II) and the secured configuration (Section III-D) were compared regarding transaction throughput and transaction latency. A complete impact modelling of the measures is out of scope as there is ongoing work on modelling the latency of HLF depending on the block-size and transaction throughput [16] and on performance modelling using stochastic reward networks [17].

Setup: The experiment used HLFs long-term stable (LTS) version 2.2 using Hyperledger Caliper, a blockchain benchmarking tool [18]. Two nodes in 2 regions of the world were used to create realistic latency. One in Ohio (us-east AWS) and one in Frankfurt (eu-central AWS). The number of nodes stayed the same across the experiments to produce comparable results. 2 nodes were chosen over 3 nodes because the default configuration can only use 2 peers at a time. Hence, using 3 nodes would positively impact the results for the secured configuration and positively skew the results in that direction. The nodes had 4 GiB of RAM, 2 vCPUs with max 3.3 GHz, 25 GiB of storage and ran Ubuntu 20.04. The configuration of the ordering service did not change between the setups, one ordering service node (OSN) with the default RAFT implementation was used. The ordering service created new blocks after 2 seconds and after 10 transactions. The minimal ordering service deployment reduces its performance impact which is out of scope of the analysis. Once pBFT algorithms can be used then the ordering services impact on performance must also be considered. Each function was invoked 1000 times at a fixed rate of 300 transactions per second (TPS) and the benchmark was repeated 3 times.

| readonly | PDC | Avg TPS (Min/Max) | Avg LAT (Min/Max) in s |
|---|---|---|---|
| | | 45.0 (36.8/52.9) | 0.60 (0.15/2.5) |
| | X | 40.6 (33.5/47.7) | 0.69 (0.26/2.8) |
| X | | 99.2 (91.7/109) | 0.21 (0.01/0.46) |
| X | X | 99.1 (99.5/107) | 0.21 (0.01/0.47) |

TABLE IX: Benchmark result for the default configuration of the fabrication stage.

| readonly | PDC | Avg TPS (Min/Max) | Avg LAT (Min/Max) in s |
|---|---|---|---|
| | | 31.2 (21.3/42.7) | 1.0 (0.17/3.4) |
| | X | 27.9 (20.2/35.5) | 1.2 (0.26/3.44) |
| X | | 115 (109/124) | 0.11 (0.01/0.56) |
| X | X | 115 (108/121) | 0.12 (0.01/0.66) |

TABLE X: Benchmark result for the secured configuration of the fabrication stage.

Results: Each function is classified into using a private data collection (PDC) or being readonly. Table IX shows the 4 cases and lists throughput and latency for the default configuration and Table X for the secured configuration.

The results of the secured configuration differ slightly from the default configuration. The write transactions are significantly better for the default configuration for latency and throughput. The additional endorsement requires additional resources for the computation and communication of the endorsement response and proposal. The transaction messages size and block-size also increase due to the additional endorsement required. These influences probably lead to the decreased performance values of the secured configuration regarding write transactions. The values for read transactions improved slightly for both latency and throughput. There exists an increased pool of possible peers to query data from in the secured configuration. This might positively affect the read transactions.

Conclusion: The application of countermeasures for the identified attack scenarios has a measurable impact on performance. Performance of read only transactions increased and performance of write transactions decreased irrespective of whether a PDC was used or not. On average both configurations produced a shorter wait time than 2 seconds which satisfies our requirement. The throughput for any write transaction is 27.9 TPS on the lower end of the spectrum. Hence, the resulting configurations both satisfies the requirements regarding performance for the use case. Further, the performance can be improved by adjusting the ordering service or by using more performant nodes and networks.

## V. Related Work

A framework to guide organizations which blockchain framework to use was introduced in [19]. Business processes are translated into smart contracts and used on blockchain to orchestrate the processes in [20]. These approaches do not consider confidentiality and the security implications of the blockchain network architectures. The business process management systems reviewed in [21] showed that most systems focus on intra-organizational processes. Inter-organizational processes are implemented, but each individual system remained independent. Kasinathan et al. show that remote maintenance in industrial use cases can be securely orchestrated using their Workflow-Driven Security Framework (WDSF) which utilizes petri nets and blockchain [3].

The security and recent advances of smart contracts was surveyed in [22]. Attacks on the blockchain structure or peer

to peer communication are described in [23]. Potential risks of Hyperledger Fabric (HLF) smart contracts are listed in [24] and general vulnerabilities of HLF are surveyed in [14]. Yet, our approach of analysing the blockchain network architecture remains unaddressed.

## VI. Conclusion and Future Work

The paper presented a formal approach towards achieving a secure Hyperledger Fabric (HLF) deployment architecture from orchestrating a cross-organizational use case with confidentiality requirements. The paper demonstrated that trust in the access control of data assets, confidentiality and integrity of the workflow execution can be established between different organizations using the solution presented in the paper. This paper investigated an attacker model and presented a guideline for applying countermeasures tailored to attack scenarios and the requirements of the presented enterprise use case. The performance evaluation showed that the secure architecture solution provides the performance necessary for the enterprise use case. Thus, this paper provides a generic methodology for securing HLF blockchain architectures for cross-organizational use cases which use PDCs, endorsement policies, and channels. As future work, we plan to extend our work in the following ways: a) deployment: a fully automatic deployment of multilateral business processes with confidential data and workflows. To achieve that goal, we must have the capabilities of automatically creating smart contracts directly from a given formal description in HLF that can be extended to utilize features such as the PDC, access control, and channels and endorsement policies; b) security: extend the attacker models and mitigations by detecting peer attacks such as blocking and manipulating of smart contracts. In addition, the influence of the peer attacker on the state of other smart contracts or channels needs to be investigated further; c) performance: a comprehensive performance analysis including the impact of multiple peers, multiple organizations in endorsement policies and their effect on the latency and throughput must be conducted.

## Acknowledgements

This research has been funded by the European Union's Horizon 2020 Research and Innovation program under grant agreements No. 830929 and No. 871518. We wish to extend special thanks to Ricarda Weber for her great reviews.

## References

[1] S. Stahnke, K. Shumaiev, J. Cuéllar, and P. Kasinathan, "Enforcing a cross-organizational workflow: An experience report," in Enterprise, Business-Process and Information Systems Modeling. Springer International Publishing, 2020, pp. 85–98.

[2] K. R. Choo, A. Dehghantanha, and R. M. Parizi, Eds., Blockchain Cybersecurity, Trust and Privacy, ser. Advances in Information Security. Springer, 2020, vol. 79. [Online]. Available: https://doi.org/10.1007/978-3-030-38181-3

[3] P. Kasinathan, D. Martintoni, B. Hofmann, V. Senni, and M. Wimmer, "Secure remote maintenance via workflow-driven security framework," in 2021 IEEE International Conference on Blockchain, 2021, pp. 29–37.

[4] Hyperledger, "Fabric," https://hyperledger-fabric.readthedocs.io/en/release-2.2/ - last accessed on March 2022.

[5] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and CPN tools for modelling and validation of concurrent systems," Int. J. Softw. Tools Technol. Transf., vol. 9, no. 3-4, pp. 213–254, 2007.

[6] W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: yet another workflow language," Inf. Syst., vol. 30, no. 4, pp. 245–275, 2005.

[7] Hyperledger, "Fabric," https://github.com/hyperledger/fabric- last accessed on March 2022, 2022.

[8] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm," in 2014 USENIX Annual Technical Conference, USENIX ATC '14, Philadelphia, PA, USA, June 19-20, 2014, G. Gibson and N. Zeldovich, Eds. USENIX Association, 2014, pp. 305–319.

[9] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in Concurrency: the Works of Leslie Lamport, 2019, pp. 203–226.

[10] Hyperledger, "Mir-bft," https://github.com/hyperledger-labs/mirbft- last accessed on March 2022, 2022.

[11] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Scalable byzantine consensus via hardware-assisted secret sharing," IEEE Transactions on Computers, vol. 68, no. 01, pp. 139–151, jan 2019.

[12] B. Hofmann, "Privacy enhancing audit trail in hyperledger blockchain," masterthesis, Hochschule für angewandte Wissenschaften Kempten, 2021.

[13] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," 2018-10-03 2018.

[14] B. Putz and G. Pernul, "Detecting blockchain security threats," in 2020 IEEE International Conference on Blockchain, Blockchain 2020, Rhodes Island, Greece, November 2-6, 2020. IEEE, 2020, pp. 313–320.

[15] F. F. Nah, "A study on tolerable waiting time: how long are web users willing to wait?" Behav. Inf. Technol., vol. 23, no. 3, pp. 153–163, 2004.

[16] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for hyperledger fabric blockchain network," Inf. Process. Manag., vol. 58, no. 1, p. 102436, 2021.

[17] H. Sukhwani, "Performance modeling & analysis of hyperledger fabric (permissioned blockchain network)," Ph.D. dissertation, Duke University, Durham, NC, USA, 2019.

[18] Hyperledger, "Caliper," https://github.com/hyperledger/caliper- last accessed on March 2022, 2022.

[19] N. Six, "Decision process for blockchain architectures based on requirements," in Proceedings of the Doctoral Consortium Papers Presented at the 32nd International Conference on Advanced Information Systems Engineering (CAiSE 2020), Grenoble, France, June 08-12, 2020, ser. CEUR Workshop Proceedings, O. Pastor and M. C. Cornax, Eds., vol. 2613. CEUR-WS.org, 2020, pp. 53–61.

[20] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, ser. Lecture Notes in Computer Science, M. L. Rosa, P. Loos, and O. Pastor, Eds., vol. 9850. Springer, 2016, pp. 329–347.

[21] S. Pourmirza, S. Peters, R. M. Dijkman, and P. Grefen, "A systematic literature review on the architecture of business process management systems," Inf. Syst., vol. 66, pp. 43–58, 2017.

[22] Y. Huang, Y. Bian, R. Li, J. L. Zhao, and P. Shi, "Smart contract security: A software lifecycle perspective," IEEE Access, vol. 7, pp. 150 184–150 202, 2019.

[23] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and D. Mohaisen, "Exploring the attack surface of blockchain: A comprehensive survey," IEEE Communications Surveys Tutorials, vol. 22, no. 3, pp. 1977–2008, 2020.

[24] K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun, "Potential risks of hyperledger fabric smart contracts," in 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2019, pp. 1–10.