Investigating Zero-Knowledge Protocols for Preserving Privacy in Coordination of Multi-Domain Applications

Adam Babs
adam.babs@student.uva.nl

September 15, 2022, 56 pages

Academic supervisor: Dr. Reginald Cushing, r.s.cushing@uva.nl
Host organisation/Research group: Informatics Institute/MNS Research Group

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Abstract

The growth of programmable infrastructures combined with the the increasing value of data facilitates the development of a new type of applications, namely, multi-domain applications. As Yuosre F. Badir et al. discuss, with globalization and the networked world of today, there is a dire need for a reliable and efficient model to manage, monitor, and control global large-scale projects, which contain thousands of workflows and hundreds of organizations located at different sites [1]. One of many examples of such multi-domain infrastructure is collaborative cyberattack defense in software-defined networking, where multiple parties share cyber threat intelligence to improve their security defense mechanisms against, for instance, DDOS attacks [2]. In this particular example, autonomous systems share blacklists containing, for example, malicious IP addresses. This process requires trust and mutual cooperation since the adversaries may gain awareness of the detection of their attack if the exchanged information is disclosed outside the trusted zone. Other use cases requiring multi-domain applications cooperation can be found here: `https://amdex.eu/usecases/` [3]. AMdEX is a field lab working on neutral exchange infrastructure that provides and executes reliable data sharing archetypes.

Distributed multi-domain applications give rise to multiple challenges such as lack of trust and single point of failure of the 3rd party coordinator, centralization of authority, and scalability limitations. On the contrary, the decentralization of cooperation between parties is associated with many challenges. How privacy can be enhanced in an environment heavily relying on transparency and universal access to all the information on the network? What are the viable mechanisms facilitating improving the privacy level?

The main objective of this work is to explore zero-knowledge protocol based approach to enhance privacy mechanisms on blockchain to prevent non-trusted parties from obtaining access to view and potentially exploit transactions published on the ledger.

Contents

1 Introduction **3**

1.1 Problem Statement 4

1.2 Research Questions 5

2 Theoretical Background **7**

2.1 Blockchain 7

2.1.1 Components & Charateristics 8

2.2 Hyperledger Blockchain 12

2.2.1 Purpose of Hyperledger Blockchain 12

2.2.2 Operational Characteristics 12

2.2.3 Confidentiality 13

2.2.4 Consensus Mechanism 13

2.2.5 Main Characteristics of Hyperledger Fabric 13

2.2.6 Components of Hyperledger 14

2.2.7 Idemix - Identity Mixer 16

2.3 Related Work 18

2.4 Petri Nets 20

2.5 Zero-Knowledge Asset Transfer (ZKAT) 21

2.5.1 zkSNARK: Zero-Knowledge Succinct Non-Interactive Argument of Knowledge 22

2.5.2 Ligero 23

2.5.3 Bulletproofs 23

2.5.4 Hyrax 24

2.5.5 Homomorphic Encryption 24

3 Proof of Concept **25**

3.1 Initial Proof of Concept: Petri Nets Workflow on Hyperleder Blockchain . . 27
3.2 Why do we need privacy? 28
3.3 Main Types of Privacy Increasing Mechanisms in Hyperledger Fabric 29
3.4 Zero-Knowledge Asset Transfer (ZKAT) 30
3.4.1 Value Commitment in Zero-Knowledge Proofs (ZKP) 31
3.5 Possible Approaches to Achieve Zero-Knowledge Asset Transfer 32
3.6 Identity Mixer Based vs. X.509 Certificate Based Identities 34
3.6.1 Identity Mixer-Based Identities 34
3.6.2 X.509 Certificate-Based Identities 35
3.7 Proposed Setup for Achieving a Zero-Knowledge Transfer with Node.js implementation 36
3.7.1 Main Components Needed to Achieve Zero-Knowledge Asset Transfer Using Identity Mixer Utilizing Node.js 38
3.7.2 Description of Idemix Identities Usage 41
3.8 Limitations of ZKAT in Hyperledger Fabric Framework 43
3.9 Problems Encountered During the Implementation Phase of the Proof-of-Concept 44

4 Project Summary & Evaluation **45**
4.1 Research Questions Answers 46
4.2 Future Work 47

Chapter 1

Introduction

Workflows are usually handled by Workflow Management Systems (WfMS), which are software components that create, direct, and monitor workflow execution. A WfMS provides help in two categories: build-time and run-time functions [4]. These functions always involve a large number of physically dispersed participants. In a conventional client–server-based workflow system, these dispersed participants interact with a centralized data repository (DR) and a centralized workflow engine with the assistance of client applications, requesting data and commands, respectively. Almost all the workflow functions are, thus, carried out on the server side in a centralized manner. Unfortunately, such an approach has encountered many problems as follows described in [5, 6, 7].

From the point of view of this work, the three most important drawbacks are the following:

1. Client-server architecture-based systems are typically vulnerable to server failures. The centralized server is typically regarded as the system's single point of failure. Furthermore, server failure could bring the entire system down. Again, this weak point is more noticeable in application domains where the workflow server is required to manage many workflow instances. [4].

2. The client-server architecture's limited scalability prevents WfMSs based on it from dealing with the ever-changing workflow environment. It also complicates system configuration because any change to the system, such as adding new participants, necessitates modifying and updating the centralized workflow server, which is both difficult and unreliable. [4].

3. Last major drawback is the single domain authority in a centralized approach which

means multi-domain applications need to trust a domain to coordinate. The utilization of blockchain will allow for independent coordination and distribution of trust among domains.

Moreover, as Yun Yan et al., the traditional client/server architecture for workflow has exhibited many weaknesses, as such an architecture seriously degrades system performance, greatly lacks scalability in a dynamic environment, and rigidly restricts human beings [5]. Petri Nets are an efficient formal model for examining concurrent and distributed systems, and they have been extensively used in various computer science areas and other disciplines. [8].

Distributed Ledger Technology (DLT) is a technological protocol that allows data to be exchanged freely by individual network members without intermediaries or third parties being used [9]. Furthermore, this technology provides a solution that enables a group of people who do not know or do not trust each other to organize themselves around certain goals, not necessarily money, without the use of institutions and external coordinators.

1.1 Problem Statement

Due to a possible lack of trust among the consortium participants, it is crucial to guarantee that the workflow is carried out correctly. This can be accomplished by handing over the coordination layer to a third-party (broker) with complete control over the workflow execution (e.g., tasks, resources, etc.). However, this could result in several security vulnerabilities. For instance, a dishonest broker could exploit their position to benefit a certain organization. Additionally, a centralized broker introduces a single point of failure problem, which jeopardizes the whole process [10]. In the single point of failure problem [11], if the coordinator fails, the whole workflow and cooperation between applications might be jeopardized. Due to the lack of central control and the decentralized nature of blockchain, the single point of failure problem is overcome [11]. This is majorly important in systems like, for instance: the aircraft maintenance data market [3] or in situations where the failure of a centralized coordinator might put the cooperation of thousands of parties in danger, like in the previously mentioned cyber attack defense [2].

As a result of utilizing the distributed consensus mechanisms, the application of blockchain to support safe, collaborative environments and deal with the lack of trust issues has proven to be an accurate choice [12, 13], and many solutions utilizing blockchain to cope with the

lack of trust have been developed. Nonetheless, the aspect of coordinating the workflow of events and concurrent processes happening on the network is rather sparsely covered.

To obtain a reliable workflow management environment, which will prevent the single point of failure issue and ensure the safety of the data exchange process, it is essential to design a decentralized architecture for workflow execution that does not contain:

(1) A centralized data repository, which will be replaced by the distributed ledger

(2) A centralized workflow engine for coordination, which will be replaced by a set of Petri Nets orchestrating the whole process.

Furthermore, the lack of robust privacy-preserving blockchain technology creates a barrier for businesses that do not want transaction data to be publicly available to outside parties. The specifics of these transactions may violate both the privacy of customers and the privacy of businesses. The problem of blockchain privacy issues has been addressed by many authors [14, 15] and remains underdeveloped. The fundamental idea is that within a blockchain transaction, the sender and receiver's addresses are concealed so that no identifying data can be extracted. To achieve this, we are going to propose a proof of concept utilizing Hyperledger Fabric [16] technologies enabling zero-knowledge asset transfer. Nevertheless, because there are still numerous ways to link identities to these pseudonyms, this method does not completely solve the privacy issue. This issue also arises on other public blockchain networks, such as the Ethereum platform, where transactions can be used to run written computations, allowing the development of decentralized applications.

1.2 Research Questions

RQ1. Can we use the zero-knowledge protocol to maintain privacy in coordinating multi-domain applications?

Sub-RQ. *How can setup zero-knowledge asset transfer to add privacy in coordinating multi-domain applications?*

RQ2. Is the current advancement of Hyperledger Fabric developed enough to provide an efficient zero-knowledge asset transfer?

RQ3. What degree of anonymity and unlinkability is possible to achieve with zero-knowledge technologies in Hyperledger?

Chapter 2

Theoretical Background

2.1 Blockchain

A blockchain is an incorruptible digital ledger of financial transactions that can be configured to record not just financial transactions but virtually everything of value [17]. This technology was first introduced by Satoshi Nakamoto in 2008 in his Bitcoin whitepaper "Bitcoin: A Peer-to-Peer Electronic Cash System" [18]. From that time onwards, blockchain revolutionized the distributed systems field. The technology introduced by Nakamoto facilitated executing "electronic transactions without relying on trust". This enables conducting transactions without putting trust into external intermediaries and provides a high level of security due to the utilization of various consensus protocols such as Proof-of-Work or Proof-of-Stake. Blockchain security is facilitated by a distributed ledger where each participant node autonomously and independently verifies that a new transaction over the network is valid. Nonetheless, there is a number of trade-offs that differ on various blockchains. However, the most frequent ones are rising latency and decreasing throughput [19], high transactional costs, or high energy consumption [17].

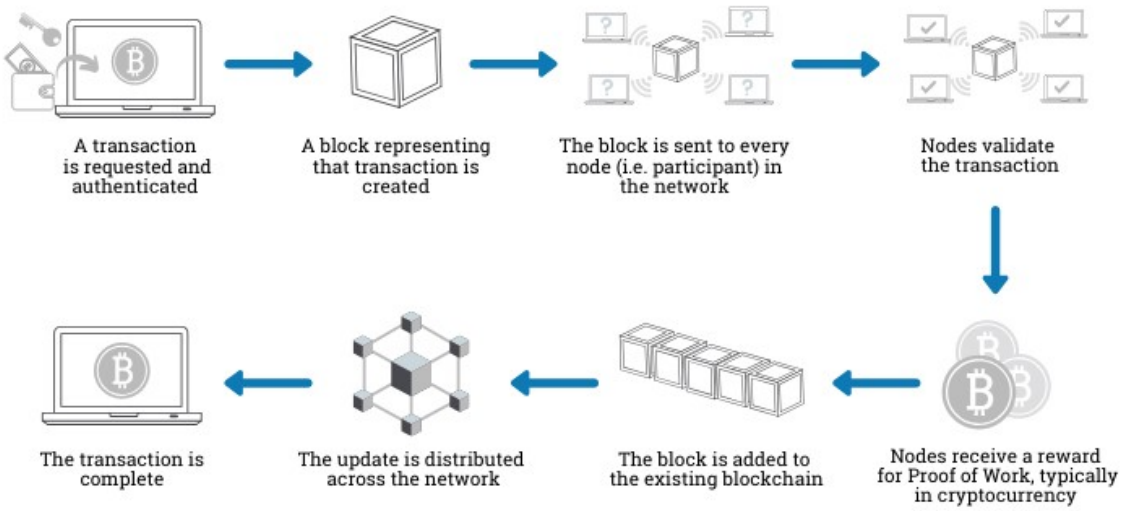

Figure 2.1: Blockchain Transaction Workflow Scheme [20].

2.1.1 Components & Charateristics

Distributed Ledger

One major component of a blockchain is a distributed database; therefore, every node that interacts with it has access to the entire database, including its history. The data and information in the blockchain are controlled by many users rather than just a single party which eliminates the need for a third party to verify the records of the parties involved in a transaction.

Immutability of Records

The majority of publications describe blockchain technology as immutable. Nonetheless, this is not entirely true as there exist conditions, although extremely difficult to achieve but not impossible, in which blockchain blocks can be altered. These are, for instance, a 51% attack, Sybil's attack, or simply cracking the cryptography [17]. Blockchain ledgers are better described as tamper evident and tamper resistant, this is achieved by unique hash values that identify each block and reference every previous block as well. This ensures that blocks are retroactively coupled together.

Blocks

Blocks are data structures within the blockchain ledger that permanently record transaction data within the blockchain. A block contains some or all of the most recent transactions that have not yet been confirmed by the network. The block is closed once the data has been validated by nodes on the network. Then, a new block is established to accept and validate new transactions. Resultantly, a block is a permanent repository of records that, once written, cannot be changed or removed.

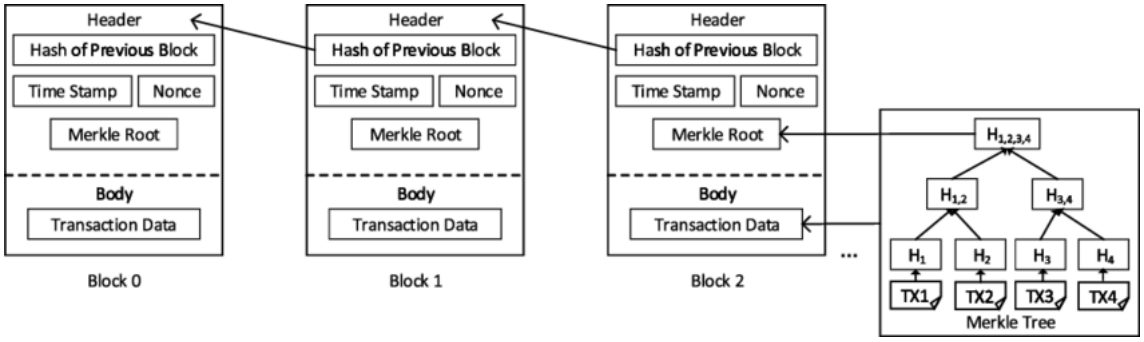

Figure 2.2: Structure of Blockchain and Underlying Blocks [21].

Peer-to-Peer Transactions

Blockchain is a decentralized peer-to-peer network of peer nodes where each node stores and tracks transactions that happen on the network. Additionally, every network participant can communicate with any other node without going through a centralized controller [22]. This attribute eliminates the single point of authority and control and makes the entire system more secure. Various consensus algorithms can be utilized to validate and record transactions virtually in a non-corruptible way.

Smart Contracts

At present, a smart contract [23] is an essential part of blockchain-based systems implemented on top of blockchains. It is a self-executing contract in which the conditions of the buyer-seller agreement are directly encoded into lines of code [24]. The code and the agreements contained within it are spread across a decentralized blockchain network. Transactions are trackable and irreversible, and the code controls the execution. As a result, smart contracts can be used to represent business rules in the form of executable code. Smart contracts enable conducting trustworthy transactions and agreements to be between disparate,

anonymous participants without the use of a central authority, legal system, or external enforcement mechanism.

Consensus Mechanisms

Consensus is a mechanism in distributed computing wherein nodes within the system must reach an agreement given the presence of faulty processes or deceptive nodes, and [25] when the participating nodes reach a decision on a value, that decision is final. Presumably, there exist dozens of different consensus protocols that can be utilized in distributed networks, and each of those algorithms can have multiple modifications to accommodate various network needs. The most widely used are the Proof-of-Work and Proof-of-Stake protocols.

Proof-of-Work (PoW)

According to Nakamoto's Bitcoin whitepaper [18], the Proof-of-Work algorithm works by scanning for a value that, when hashed, has a hash starting with a number of zero bits. This is accomplished by adding a nonce to the original value until the resultant hash starts with the requisite number of zero bits. Once this nonce has been found and the proof of work requirement has been satisfied, the block cannot be changed without redoing the work for that specific block and all blocks that come after it [25].

Proof-of-Stake (PoS)

Proof-of-Stake consensus was developed to reduce the high computational requirements of Proof-of-Work mechanisms. Participants with higher coin age, i.e., a product of network tokens and their holding time, have higher chances of being selected. Specifically, each node in the network solves a Proof-of-Work puzzle with its own difficulty, which can be reduced by consuming coin age. In the more recent PoS networks, the solution searching is completely removed, and the block leaders are no longer selected by computational power [26]. Instead, they are selected based on the stakes they are holding with the stake-based leader selection process, a node's likelihood of being chosen as a leader is no longer determined by its processing power, and hence PoS techniques consume significantly less energy compared to the PoW processes.

Proof-of-Stake vs. Proof-of-Work

In the PoS consensus, network control can be bought due to the staked-based leader selection, whereas in PoW, it is mostly based on the computational resources of the network

participant, which ensures more security. In PoW, the process is executed by miners that consume energy to work on finding a nonce. On the contrary, in PoS, one must possess tokens or stakes to participate, which is much more energy efficient. Due to the price of mining equipment and energy, it is more difficult to scale the network, whereas in PoS, obtaining a fraction of the stake facilitates joining the network, providing better scalability.

Raft

Raft is the most frequently utilized consensus mechanism in Hyperledger, it addresses the issue of making multiple servers agree on a shared state even in the face of failures. A replicated log typically supports the shared status as a data structure. Raft operates by electing a cluster leader. The leader is in charge of accepting client requests and managing log replication to other servers. Data only flows in one direction: from the leader to the other servers.
Raft decomposes consensus into three sub-problems [27]:

- **Leader Election:** If an existing leader fails, a new one must be selected [27].
- **Log replication:** Through replication, the leader must keep all servers' logs in sync with its own [27].
- **Safety:** If one of the servers has committed a log entry for a specific index, no other server can use a different log entry for that index [27].

Blockchain Types

There are several different blockchain types. The most generic division can be made by differentiating between private & public as well as permissioned & permissionless networks. Each has the following properties:

- Permissioned - only pre-authorized users can add nodes to a permissioned blockchain network, they are usually faster and more scalable than permissionless blockchains. Additionally, they tend to be more centralized and restrictive. For instance, the Hyperledger framework. We can further divide permissioned networks into two categories:
  - Private networks are permissioned blockchains that are managed by a single entity. The central authority determines who can be a node in a private blockchain [28].

  - Consortium blockchains are permissioned blockchains that are managed by a group of organizations rather than a single entity, as is the case with private blockchains [28].

- Permissionless - this blockchain network allows anyone to become a part of the network and contribute to its state. The open nature of permissionless blockchain comes at the cost of throughput and scalability. The most known examples are Bitcoin and Ethereum.

  - Public networks are permissionless in nature, open to anyone, and entirely decentralized. Public blockchains give all blockchain nodes equal access to the blockchain, allowing them to create new blocks of data and validate existing data blocks [28].

2.2 Hyperledger Blockchain

Hyperledger Fabric is an enterprise-grade distributed ledger based on blockchain technologies that utilize smart contracts to enforce trust between parties.

2.2.1 Purpose of Hyperledger Blockchain

The Linux Foundation hosts Hyperledger, a multi-project open source collaborative effort designed to advance cross-industry blockchain technologies [29]. Hyperledger founders aim to foster the development and adoption of cross-industry platforms powered by distributed ledgers. The platform facilitates the development of personalized blockchain networks to accommodate and fulfill various business needs.

2.2.2 Operational Characteristics

Hyperledger is a private and permission based blockchain, which means that only authorized users can obtain access to the network. The network is restricted to a predetermined group of members who have been granted consent to participate. On the contrary, Ethereum, Bitcoin, or Cardano are public, this implies that anyone can access the blockchain network, and no authorization is required to participate.

2.2.3 Confidentiality

The private and permissioned nature of Hyperledger implies that transactions on the network are visible only to authorized members as opposed to public networks where all transactions are visible to the outside world. In public networks, all transactions recorded are accessible by every peer node in the network.

2.2.4 Consensus Mechanism

Due to Hyperledger's private and permissioned nature, the consensus implementation can be tailored to the trust assumption of a particular deployment or solution. For instance, if two parties agree on a given transaction, no third party can view or intervene in that transaction. This contributes to increased scalability, transaction rates, and overall network performance [30]. Hyperledger achieves consensus by utilizing a backend system called the ordering service. It allows for the usage of a pluggable consensus algorithm, facilitating the orderer to be switched based on the needs of the environment.

2.2.5 Main Characteristics of Hyperledger Fabric

- Hyperledger stands out for its high throughput of up to 500 000 transactions per minute compared to Ethereum's Network 1000 per minute.
- No single place to store data, with existing tools to limit access to some data or completely prevent sensitive data from being replicated on specific nodes.
- No enforcement of requirements about hardware, network infrastructure, additional software to be combined with, security models. It is very flexible software.
- Very unlikely to be a target of a 51% (due to permissioned-based nature), unlike public, more decentralized blockchains. Policies can be configured to require 100% approval to transact.
- Highly modular and flexible architecture, various functionalities can be plugged in.
- It offers multi-language smart contract support.
- Hyperledger offers support for EVM[1] and Solidity.
- It has an open smart contract model that imparts the flexibility to implement any desired solution model (account model, UTXO model, etc.) [31].

[1] Ethereum Virtual Machine

2.2.6 Components of Hyperledger

Certificate Authority

Certificate Authority (CA) serves three main purposes on the network:

- Issues certificates of enrollment to the participating organizational units (OUs)
- Revocates and renews certificates of enrollment
- Registers credentials of users

Furthermore, it offers a variety of identity certificate-related services to blockchain users. These services are specifically related to user enrollment, blockchain transactions, and TLS-secured interactions between organizations, organizational units, or blockchain components.

Membership Service Provider (MSP)

The membership service provider is the mechanism that allows organizations and organizational units to engage in a permissioned blockchain network. The MSP is how an organization's identity is tied to its membership. Participation in an organization is achieved by adding the member's certificate to the organizational membership service provider.

Ledger

A ledger in Hyperledger Fabric comprises two distinct but related parts: a world state and a blockchain. Each of these represents a collection of facts about a specific set of business objects [32].

- Blockchain is a log of transactions organized into blocks and ordered by a service. It is physically stored and immutable in the peer nodes. It is irreversible.
- Ledger is a key value store that contains data generated by blockchain transactions. The global state, unlike the blockchain, is mutable. It contains the most recent values for all keys modified in the transactions.

Policies

A policy, at its most basic, is a set of rules that describe the structure for how decisions are made and particular results are achieved. To that aim, rules often identify a who and

a what, such as an individual's access or rights to an asset.

Policies enable members to decide which organizations can access or update a Fabric network, as well as give way for enforcing those decisions. Policies include a list of organizations with access to a specific resource, such as a user or system chaincode. They also indicate how many organizations must agree on a proposal before a resource like a channel or smart contracts can be updated. Policies, once defined, analyze the collection of signatures connected to transactions and proposals and validate whether the signatures meet the network's agreed-upon governance.

The Ordering Service

Unlike the most well-known, permissionless blockchains, where any node can participate in the consensus process, which orders and packs transactions into blocks based on a consensus algorithm, the Hyperledger network contains an ordering node responsible for transaction ordering. Fabric's design is based on deterministic consensus techniques. Therefore any block certified by a peer is guaranteed to be final and correct. In many other distributed and permissionless blockchain networks, ledgers can fork. Separating the endorsement of chaincode execution (which occurs at peers) from ordering affords Fabric advantages in efficiency and scalability, reducing bottlenecks that can occur when execution and orders are handled by the same nodes [33].

Channels

A channel is a private communication pathway between two or more members of a Hyperledger Fabric network. A channel functions similarly to a virtual blockchain network that sits on top of a physical blockchain network and has its own set of access rules. Channels use their own transaction ordering method to achieve scalability, allowing for the effective ordering and partitioning of massive volumes of data [34].

Peer Nodes

A blockchain network is mainly made up of peer nodes (or, simply, peers). Peers are critical components of the network because they house ledgers and smart contracts. A peer is associated with a single organization, can be a member of several channels, and can hold multiple ledgers and smart contracts. Peers pass smart contract calls to network-specific chaincode containers and adjust network state based on smart contract results.

Chaincode

Chaincode is a program written in executable code that implements a prescribed interface. Chaincode runs in a secured Docker container isolated from the endorsing peer process it defines the business logic and initializes and manages the ledger state through transactions submitted by applications. It needs to be installed and then instantiated on each peer that is a part of the channel on which we want to deploy the chaincode.

Smart Contracts

A smart contract is an executable piece of code that specifies the regulations between multiple organizations. To generate transactions that are recorded on the ledger, applications invoke a smart contract. In Hyperledger, smart contracts are typically defined within a chaincode as a subcomponent of a chaincode. When chaincode is deployed, smart contracts within this chain are available to the applications. Below is a high-level visual representation of chaincode and smart contract dependency.

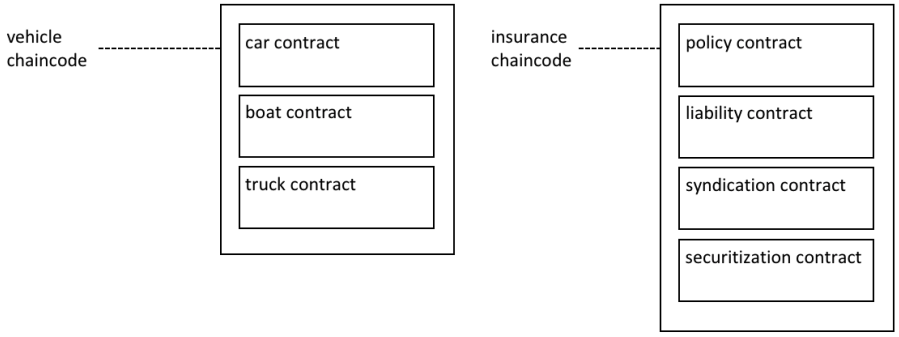

Figure 2.3: High-level smart contract and chaincode dependency. [35].

A chaincode contains a definition of a smart contract. The same chaincode can define several smart contracts. All of the smart contracts contained in a chaincode are made accessible to applications when it is deployed.

2.2.7 Idemix - Identity Mixer

Identity Mixer (Idemix) is an anonymous cryptographic protocol suite, created by IBM, which provides anonymous authentication and privacy. A credential is a mean of establishing a claimed identity, roles, or traits about oneself with an entity, generally as part of an

access control request. Additionally, all of this is executed without revealing the identity of the transactor. For example, a driving license card can be used to prove that a person is allowed to drive a car, which may be required during the city guard's control. Using a regular driver's license card would reveal to the controlling side all of the other information on the license.

2.3 Related Work

Due to its decentralized and tamper-proof nature, blockchain technology has the potential to solve a wide range of issues in a variety of fields, including data mismanagement and misalignment in health-related organizations, secure cross-organizational data exchange, preventing data security breaches, greatly improving asset recording, tracking, and sharing in all types of supply chains, or claiming ownership of intellectual property, among others. Additionally, many more applications of this technology will likely be identified in the future. Several frameworks for blockchain-based data sharing have been proposed, with a clear goal of solving the problem of possible lack of trust among the organizations that pose relevant issues in the processing of sensitive resources.

Composition of services that take part in the cross-organizational can be coordinated according to the two main approaches [36]:

(1) orchestration

(2) choreography

The first is based on a decentralized architecture, in which services independently coordinate their invocations. Orchestration, on the other hand, necessitates the presence of a central broker responsible for initiating and monitoring BPEL service invocations (Business Process Execution Language). The latter method is more in line with the idea of deploying a business process on the blockchain and having the smart contract operate as a broker and monitor [10].

Several approaches in the literature use access control techniques to achieve secure resource sharing in inter-organizational workflows. [37] was one of the first publications utilizing blockchain technology to overcome the lack of trust issues among participating organizations. Authors propose an "Encrypter" framework ensuring data integrity. The main goal of this work is to ensure the confidentiality of data exchanged on the blockchain. To achieve this, data exchange is encrypted and can only be decrypted by authorized organizations while simultaneously allowing smart contracts to perform computations on this data.

In [2], the authors propose a secure, scalable, immutable, low-cost, and easy-to-deploy decentralized infrastructure where different parties can share cyber threat intelligence to prevent DDoS attacks. Authors encode the mutual business logic between different parties

through smart contracts that facilitate submissions and retrievals of IP data from the network. When a malicious incident is reported by one of the organizations on the blockchain, and it has been a victim of an attack, this node submits the incident to the blockchain and includes a blacklist of malicious IPs. Then according to the business logic, other nodes are informed of the blacklist and start enforcing mitigation rules such as traffic blocking or rate-limitation.

Moreover, there is work [38] that analyzes how blockchain can facilitate secure inter-organizational cooperation with its benefits and potential shortcomings. Authors enumerate several reasons why blockchain introduces an unprecedented potential for improving cross-organizational processes within organizations that have weak trust relationships. Nonetheless, several security issues are listed and analyzed. One issue directly related to our work is referred to by authors as an Oracle correct flow. It is reasoned that there exists a need for ensuring the correct execution of services due to the possibility that an Oracle could invoke the delivery service of a corrupted organization rather than the one required by the smart contract. One of the main ideas of our work is to execute the cooperation between nodes based on the predefined set of smart contracts that will be deployed as a Petri Net, therefore, ensuring the correct execution of services.

In [10], the authors propose a solution for controlled information sharing in inter-organizational workflows enforced via smart contracts. Christian Rondanini et al. encode into an "engine" smart contract the logic of the workflow coordination. This workflow engine comprises all business rules established by the involved organizations and serves as a coordination layer. Another key component is an invocation smart contract that decides which is the next task to be invoked till reaching the final state.

While the above-mentioned solutions demonstrate a growing interest in deploying corporate operations on the blockchain, none of them propose a substantial and well-established [39, 40, 41] framework for safe zero-knowledge asset transfers, which is the main goal of this work.
To the best of our knowledge, there are no papers about combining the Petri Net models with blockchain to coordinate distributed multi-domain applications. However, these two tools have often been used together before for blockchain analysis [42], verification of smart contracts on the blockchain, [43] or generation of safe smart contracts based on Petri Nets models [44].

As Wil van der Aalst describes [45], Petri Nets have a wide spectrum of possible applications in managing workflow systems, and there are at least three good reasons for developing Petri-net-based workflow management systems. These are:

- Formal semantics, regardless of their graphical nature.
- State-based instead of event-based flow.
- Wide range of available analysis techniques and tools.

In later stages, the author proves that Petri Nets can cover all requirements stated by the Workflow Management Coalition [46] that established standards to be used in workflow management. Furthermore, Petri Nets facilitate state-based descriptions of WFMS, which allows for a clear distinction between enabling a task and executing a task. This has proven to be valuable in numerous projects. Lastly, there is an abundance of analysis techniques developed for Petri Nets, enabling the user of a Petri-net-based WFMS to analyze a workflow process in various ways. There are many tools to simulate the workflow and analyze its performance and structure from different angles.

Recently, Petri Nets have been mainly considered as a modeling tool for workflow and workflow systems [39, 40, 41]. There are a number of reasons [45, 47] for using a Petri Net-based approach in modeling and analyzing workflow systems. Petri Nets allow a graphical representation to ease the understanding of the modeled system, and at the same time, they can be used in formal analysis, verification, and validation of the model [48, 49, 50]. In a brief summary, different modeling techniques can be applied to workflow modeling, but Petri Nets are the only formal techniques able to be used for both structural modeling and a wide range of qualitative and quantitative analysis [51].

2.4 Petri Nets

A Petri net is a graphical representation for describing and analyzing concurrent processes that occur in large systems with many components (distributed systems). The graphics, as well as the principles for coarsening and refining them were invented in August 1939 by the German Carl Adam Petri [52].

A Petri net is composed of points, transitions, and arcs. Arcs always run from one location to another, never between locations or transitions. The spots where an arc goes to a transition are known as the input sites of the transition, whereas the places where arcs flow away from a transition are known as the output places of the transition. Petri net execution is nondeterministic unless an execution policy is defined: when many transitions are activated simultaneously, they will fire in any sequence. Because firing is nondeterministic and several tokens may be present anywhere in the net, Petri Nets are well suited for modeling the concurrent behavior of distributed systems [53].

2.5 Zero-Knowledge Asset Transfer (ZKAT)

The fundamental idea behind zero-knowledge proofs can be described as an allegory of a ring-shaped Ali Baba cave. This cave has two paths (A and B), one entrance, and a magic door at the end that separates path A from B. Suppose Alice has discovered a secret word that opens the magic door and wants to prove to Bob that she possessed the knowledge without revealing the word. To successfully achieve this, Alice enters the cave without showing Bob which entrance she chose to enter. Then Bob comes to the entrance and tells Alice, which path she should take to return to the exit. Assuming that Alice truly knows the secret, she will always be able to return to the exit. Given that Bob chooses the return path at random, probability of Alice taking the right path is always 50%, however, if they repeat the same action 10 or 15 times and each time she successfully returns using the correct path, the probability of Alice not knowing the secret is around 0.09% and 0.003% accordingly.

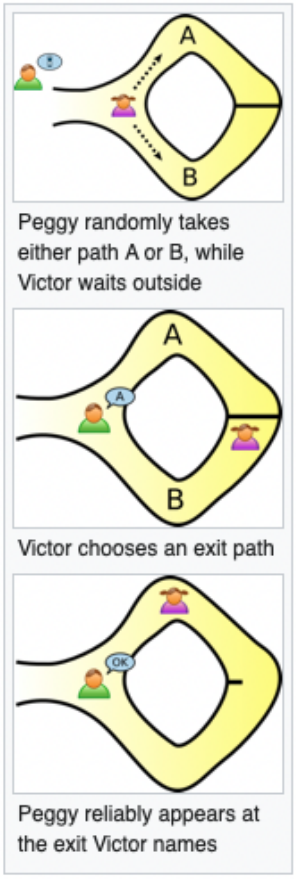

Figure 2.4: Visual Representation of the Ali Baba Cave [54]

The application of zero-knowledge proofs in blockchain transactions has been a focus of study in recent years. Some effective strategies and methodologies for using these protocols in blockchain-based networks have been developed. This chapter will examine pertinent research and solutions contributing to implementing zero-knowledge proof arguments on the blockchain.

2.5.1 zkSNARK: Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

There already exist many concepts that introduce zero-knowledge to blockchain technology that are based on different mechanisms. The most researched one seems to be the zkSNARK protocol. Bitansky et al. [55] introduced the acronym zkSNARK for a zero-knowledge succinct non-interactive argument of knowledge that became a foundation for many protocols,

for instance, the ZeroCash (Zcash) protocol.

Zcash is the first widely used example of zk-SNARKs, a novel way of zero-knowledge encryption. Zcash's strong privacy guarantee is because all of its shielded transactions can be entirely encrypted on the blockchain and validated with zk-SNARK proofs.

ZK-SNARK mainly consists of setup, prover, and verifier, where setup is a procedure that generates the proving key PK and the verification key VK by using a predefined security parameter λ and an F-arithmetic circuit C, which is a circuit that all inputs are elements in a field F, and its gates output elements in F. PK is used for generating verifiable proof. VK is used for verifying the generated proof. Based on the generated PK, the input $x \in Fn$ and the witness $W \in Fh$, the prover generates a proof π. Finally, with the usage of VK, x, and π, the verifier verifies π. According to the verification result, π is accepted or rejected. Furthermore, zkSNARK has additional characteristics. First, the verification can be executed in a short running time. In addition, the proof size is just several bytes. Second, the prover and verifier are not required to communicate with each other synchronously [56].

2.5.2 Ligero

The Ligero Protocol [57] redesigned the technique of building a zero-knowledge proof and applied it to a more "complex" secure computing protocol called "Ligero." The Ligero protocol is non-interactive and does not necessitate the usage of a trusted configuration. The protocol focuses on both big and relatively large verification circuits. One of the aims Ligero set out to achieve was to eliminate the necessity for a trusted setup and the extensive usage of public-key primitives and sophisticated combinatorial objects that existed in current solutions at the time. For circuits with 3 million or more gates, Ligero's communication complexity is independent of circuit size.

2.5.3 Bulletproofs

Bulletproofs are short zero-knowledge proofs that do not require any trusted setup. A bulletproof can persuade a verifier that encrypted plaintext is well formed. Prove that an encrypted number is in a given range, for example, without revealing anything else about the number. Bulletproofs, unlike SNARKs, do not require a trusted setup. Nevertheless, verifying a bulletproof takes longer than verifying a SNARK proof [58]. The amount transferred in a confidential transaction is concealed and every confidential transaction includes

cryptographic proof that it is valid. Additionally, Bulletproofs allow proofs to be committed on committed values independently, which implies that a statement can be proven directly rather than through the implementation of a specific commitment algorithm in the proof system itself.

2.5.4 Hyrax

Wahby et al. [59] originally devised a zero-knowledge argument technique with low communication and verification costs. There is no trusted setup in this argument in this scheme. When applied to batch statements, the verification time has a sub-linear connection with the arithmetic circuit size, as well as certain appropriate constants. Along with suitable constants, there is a linear relationship between the prover's running time and the arithmetic circuit size. Based on the suggested approach, the authors have created a Hyrax zkSNARK model. In terms of proof size and processing overhead, Hyrax outperforms other techniques.

2.5.5 Homomorphic Encryption

A comparable concept to the zero-knowledge asset transfer that is extensively researched [60, 61] in the blockchain field to provide and enhance privacy is homomorphic encryption. Homomorphic encryption is a type of encryption that allows users to compute on encrypted material without first decrypting it. These resulting computations are left in encrypted form, which, when decoded, produces the same outcome as if the operations had been conducted on unencrypted data. Homomorphic encryption can be used to ensure the privacy of outsourced storage and computing. This enables data to be encrypted before being sent to commercial cloud environments for processing.

Chapter 3

Proof of Concept

Our initial proof of concept was to create a multi-domain workflow to model blockchain-based cooperation between potentially untrustful parties and execute this workflow as a Petri Net graph. After constructing a working implementation of a Petri Net-based cooperation [62], our focus switched to increasing privacy and safety by being able to hide and anonymize data published on the ledger.

The idea of having an anonymous and unlinkable identity can have various applications in different fields of the economy or state administration. The most intuitive application of such privacy models can be a financial system where multiple parties transact, and some want to hide their transactions from being disclosed to other parties, apart from the ones that are authorized.

Essentially, are three main options for data concealment:

- Full transparency, where each organization participating in the network can see all transactions and the ledger state, this configuration has no privacy mechanisms implemented on the network, apart from being a permissioned blockchain network.

Ledger State Accounts	Transaction Log
Org1: 10k$ | Org2: 440k$ | Org3: 1k$	Org2 -> 100k$ -> Org3
Org1: 10k$ | Org2: 340$ | Org3: 101k$	Org1 1k$ -> Org3 | Org3 -> 5k$ -> Org2
Org1: 9k$ | Org2: 345$ | Org3: 97k$	Org3 > 29k$ > Org1
Org1: 38k $ | Org2: 345$ | Org3: 68k $	...

Table 3.1: Full Transparency

- Restricted transparency, where chosen parties or auditors can access chosen parts of the ledger and transaction log, must be stated in the Idemix identity configuration files. This is what the proof of concept aims for, as it provides flexibility in auditing data access by particular organizations.

Ledger State Accounts	Transaction Log
Org1: 10k$ | [redacted]	[redacted]
Org1: 10k$ | [redacted]	Org1 1k$ -> Org3 | [redacted]
Org1: 9k$ | [redacted]	Org3 > 29k$ > Org1
Org1: 38k$ | [redacted]	...

Table 3.2: Restricted Transparency

- Full privacy, this is the full concealment of data. It shows the auditing capability of ZKAT protocols.

Ledger State Accounts	Transaction Log
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	[redacted]
[redacted]	...

Table 3.3: Full Privacy

Another application of such an Idemix-based identity used in a Petri Net workflow could be, for instance, a supply chain where the ordering party has to hide its identity due to various reasons (for instance, military equipment or medical supply).

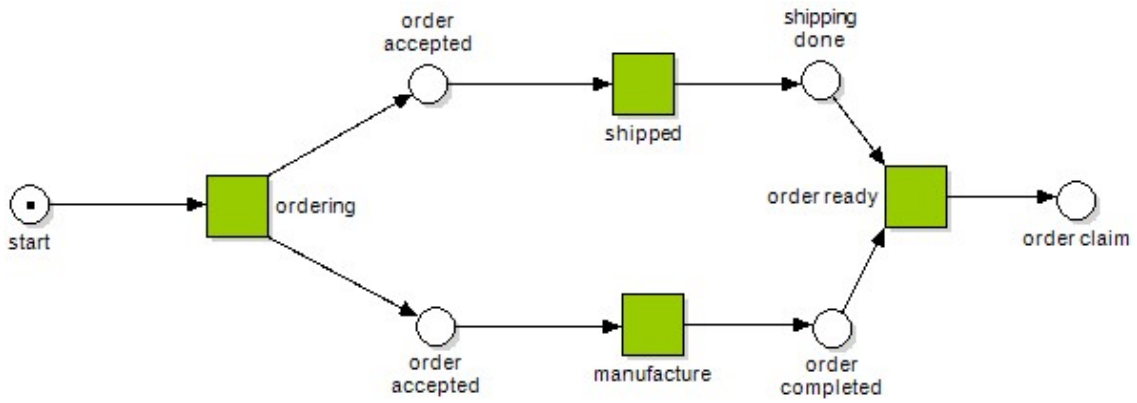

Figure 3.1: Petri Net Model of a Supply Chain

In the above configuration 3.5, if the ordering party stays anonymous, it can hide its entire identity and remain certain that confidential information will not be revealed to the manufacturing side. Furthermore, the manufacturing side does not need to know anything about the orderer except what has been ordered and in what quantity. Nonetheless, due to high flexibility, different configurations of revealing data can be specified, as illustrated in the tables 3.2, 3.3, 3.1.

3.1 Initial Proof of Concept: Petri Nets Workflow on Hyperleder Blockchain

In contrast to Petri Nets, which offers tools for modeling and coordinating concurrent processes, the blockchain network gives us access to data management, computing, and authentication. Combining these two ideas makes it easier to decentralize the coordination of concurrent processes. A basic Petri Net is mapped to the blockchain by defining it on the ledger and then adding functionality to move tokens and trigger transitions. An asset with the arcs connecting the places and transitions is created once assets are initially constructed for the three fundamental components (places, transitions, The created assets are held by the creator via the underlying public key infrastructure (PKI), and other users cannot change them until the creator transfers ownership. A Petri Net comprised of resources from various domains would be called multi-domain. When a net with many domains is created, each domain must accept the net by signing it. The internet then starts to function. Domains have the right to revoke any network at any time, making the whole net inactive.

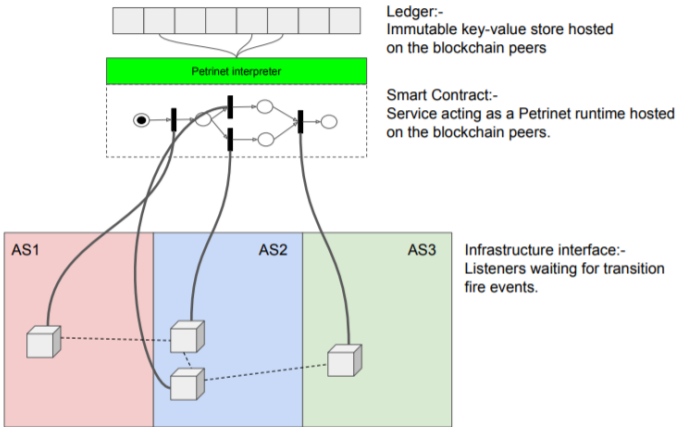

Figure 3.2: Interaction between the Petri Net layer and infrastructure layer. As the contract executes a Petri Net by moving tokens in places, transitions fire which generate events. These events are trapped by the infrastructure interface. [62].

As shown in figure 3.2, interaction with the outside world occurs through blockchain events. A kind of publish-subscribe system on the blockchain. Firing events instruct the infrastructure to do tasks and report to the blockchain. At this point, the Petri Net executor decides to move the tokens to the output places [62]. In a Petri net, the complex access, authorization, and authentication processes used by many organizations are documented. The Petri net is operated on a hyperledger layer, eliminating the control of a central coordinator. The infrastructure can monitor events involving transition firing due to the ledger event listeners on several domains.

3.2 Why do we need privacy?

In the era of ubiquitous data privacy-preserving mechanisms and the increasing role of blockchain in society in cryptocurrency-based payments or decentralized voting polls, privacy on blockchain will undoubtedly become a crucial aspect. In general, blockchain records cannot be deleted, but most modern privacy laws grant individuals the "right to be forgotten." When the information recorded on a blockchain's ledger is permanent, how can an individual or data subject exercise their right to be forgotten [63]?

In the initial proof of concept, a zero-knowledge asset transfer may provide a great advantage to parties that will want to selectively disclose their data, mainly due to its worth and the potential threats it may induce when leaked. As a result, certain privacy-enhancing

mechanisms should be implemented to improve the initial idea further.

3.3 Main Types of Privacy Increasing Mechanisms in Hyperledger Fabric

Although private and permissioned blockchains are mostly known for their security and safety of data exchange, they still need solutions for hiding transactional data from being disclosed to unauthorized parties. Currently, there exist two common privacy-preserving mechanisms of restricting access to data to particular parties on Hyperledger [34]. These are:

- Channels
- Private data collections

While private transactions prohibit unwanted parties from directly accessing private data, they do not prevent the ledger participants from recognizing when this private data is updated. Private data hashes occupy key-value entries in the ledger state, the changes to which are visible to the public. Additionally, they need to be accompanied by anonymous client authentication mechanisms to avoid exposing the link between the transaction's creator's identity and the ledger's stored (hashed) data [34].

As described previously in 2.2.6, channels enable preserving the privacy and confidentiality of information exclusively within the nodes that are in the channel, and they are useful when a subset of the blockchain network's participants share a large number of transactions, and these transactions can be processed without relying on state controlled by entities outside this group [34]. Creating separate channels for each of the possible data exchange cases, however, adds administrative overhead (maintaining chaincode versions, policies, MSPs, etc.) [64] and does not allow for use cases where you want all channel participants to see a transaction while keeping a portion of the data private, this results in very limited scalability [34].

According to many sources [65, 56], zero-knowledge asset transfer is a seemingly suitable solution to these flaws.

3.4 Zero-Knowledge Asset Transfer (ZKAT)

Zero-Knowledge Transfer Requirements and Explanation

While implementing confidential transactions, the main aim is to hide and protect transactional data. These might be fees, the number of possessed assets, prices, and much other information used for business purposes. This can, for instance, be used in multi-party cooperation with a supply chain where the supplier might want to hide the margin added to the final price from his customers.

Suppose that organization A wants to prove to organization B that they know some amount X without disclosing X or any other information about it. This needs to hold three properties [66]:

- Completeness: if a proof is valid, organization B will be certain of its validity.
- Soundness: if organization A does not know the amount X, they cannot cheat about it
- Zero-knowledge: if organization A knows the amount X, organization B does not need to know more than that organization A knows the amount X.

In simple words, zero-knowledge transfers can be explained in the following manner. Suppose that to buy cigarettes, you have to be at least 18 years old. Zero-knowledge asset transfer will confirm to the seller whether the buyer is eligible to purchase it without disclosure of birth date or age. The seller will only know if the buyer is eligible to buy cigarettes without acknowledging any of the private data that the buyer wants to avoid disclosing. This mechanism can have multiple applications in business. For example, on the supply chain, the manufacturer wants to hide his margin from various clients in order not to disclose his business strategy to the competition. For instance, in the banking world, if financial institutions would like to be able to exchange assets and record the relevant transactions in the shared ledger without disclosing the fact that they are transacting, who they are transacting with, or the value of the assets they are exchanging in their transactions. Failure to do so would clearly violate company confidentiality policies and disclose their business methods [34].

Zero-Knowledge Protocol Types

Zero-knowledge protocols can be divided into two main categories, depending on the interaction required between the prover and the verifier. These two are interactive and non-interactive protocols.

Interactive Proofs

To gain the verifier's trust in the presence and authenticity of the prover's secret information, interactive requires the prover to complete the challenges provided by the verifier. The verification procedure entails the two parties exchanging information until the prover meets the parameters of the evidence.

Non-interactive Proofs

In non-interactive ZKPs, the prover must solve challenges based on the prover's commitments presented by a simulated verifier. Non-interactive ZKPs differ from interactive ZKPs in that an automated system, rather than another human, verifies the prover's claims. As a result, these ZKPs necessitate additional software and processing power. Non-interactive ZKPs allow users to conduct transactions without requiring direct interaction between the parties involved, and there is just a single message sent from the prover to the verifier who is involved. Hyperledger's Identity Mixer is an example of a non-interactive protocol, as there cannot be any additional interaction between the parties.

3.4.1 Value Commitment in Zero-Knowledge Proofs (ZKP)

Suppose two organizations, A and B, exchange data and one organization is a verifier and the other one is a prover. Prover holds to a value X that they do not want to disclose to the verifier, simultaneously convincing the verifier that this value cannot be modified. Therefore the prover sends a commitment to the value X to the verifier. At the same time, the verifier should not be able to decode the received value from the commitment. When the prover wants to disclose the value of X, he sends an opening to the verifier, from which the verifier can compute the value of X and check if the commitment was made honestly. This is also referred to as the binding property of the commitment schemes used in zero-knowledge Proofs [34, 67].

3.5 Possible Approaches to Achieve Zero-Knowledge Asset Transfer

Hyperledger developers are currently working on a solution for providing an ability to anonymize transactions (hiding the identity of parties) and to execute them fully confidentially (hiding the identities of senders of receivers in transactions).

The framework which is under development is Idemix, which is aimed at providing the ability to transact on the blockchain network without revealing the identity of the transactor [68]. It utilizes various cryptographic solutions to store and create anonymous credentials which distrustful parties on the blockchain can utilize to rest assured of their anonymity.

Idemix-based identity combined with an appropriate signing function based on Curve25519 based on elliptic curve cryptography or the Pedersen Commitment, based on scalar multiplication, can potentially facilitate zero-knowledge asset transferring on Hyperledger Fabric blockchain.

Firstly, we shall consider what is a standard process of submitting a transaction to a Fabric network where parties have a clear overview of what is committed to the ledger and which organization is an author. In the Hyperledger Fabric framework, there is a number of steps required to commit and validate a transaction to the network successfully.

1. **Transaction proposal**
   The application sends a transaction proposal to a chosen number of peer nodes. The smart contract and the arguments for chaincode activation are specified in the transaction proposal.

2. **Execution of the transaction proposal**
   The chaincode is executed by the peers who receive the proposal using the arguments specified. They include the execution outputs, the return value, and a read/write-set in the proposal. The read/write-set contains updates to and dependencies on the world state. It is important to note that the world state does not change during a chaincode invocation; suggested modifications are just described in the read-write set. All peers who run the chaincode sign the output and return it to the application. This is referred to as an endorsement.

3. **Gather all endorsed transactions**
   The application creates a transaction out of all endorsed transactions and delivers it to the ordering service.

4. **Order transaction**
   The ordering service gathers incoming transactions and groups them into blocks using a consensus process. When a block is finished, the ordering service distributes it to the committing peers.

5. **Transaction validation**
   Upon receiving a new block by the committing peers, they append it to the ledger and validate each transaction included within it. Validation primarily guarantees that a transaction's endorsements satisfy the endorsement rules for that chaincode and that the read/write set does not conflict with previously committed concurrent modifications. If a transaction is confirmed legitimate, the world state is updated with the transaction's read/write set.

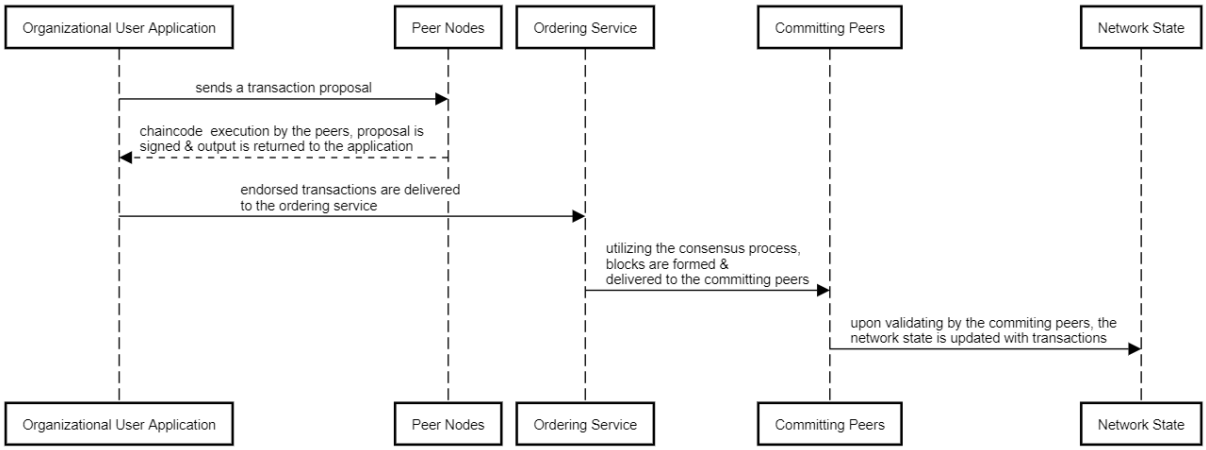

Figure 3.3: Visualization of the Process of Submitting a Transaction to the Fabric Network

Currently, according to our research, there does not exist a ready approach or library to support Idemix-based identities and fully provide an ability to perform a genuine zero-knowledge asset transfer. Furthermore, there is a long list of limitations to the current ideas and solutions being worked on.

3.6 Identity Mixer Based vs. X.509 Certificate Based Identities

In general, the creation, storage, and usage of Idemix and x.509 identities are very similar. A set of attributes is digitally signed with an immutable signature, and a credential is cryptographically coupled to a secret key. The main difference is that when an X.509 certificate is submitted, all attributes must be displayed for the certificate signature to be verified. This means that all certificate usages for transaction signing are linkable, whereas, in the Idemix-based credentials, only a subset of attributes is needed to verify a signature.

3.6.1 Identity Mixer-Based Identities

Identity Mixer (Idemix) is an anonymous cryptographic protocol suite, created by IBM that provides anonymous authentication and privacy. A credential is a mean of establishing a claimed identity, roles, or traits about oneself with an entity, generally as part of an access control request. For example, a driving license card can be used to prove that a person is allowed to drive a car, which may be required during the city guard's control. Using a regular driver's license card would reveal to the controlling side all of the other information on the license.

Anonymous credentials address this by allowing a user to selectively expose any of the attributes provided in the credential without revealing their personal information. As a result, anonymous credentials are an important component in protecting one's privacy in a digitalized world.

Users can use an identity mixer to receive a credential from an issuer that contains all of the information the issuer is willing to certify about them. When a user later has to validate a statement about herself to a service provider, they use an identity mixer to safely alter the supplied credential. The modified credential will only include the attested information they are willing to reveal. The user can apply this transformation as many times as they like, but none of the credentials will be associated together.

There is a number of components that contribute to the creation of anonymous identity.

- Issue certificate
  Is created based on four main sub-components that will be certified:

  - Organizational unit (OU)
  - Role
  - Enrollment ID
  - Revocation handle

  Using these elements, the certificate authority (CA) (see:2.2.6) randomly samples the issuer secret key and computes the issuer public key.

- Client certificate
- Signature generation
  To sign a message and simultaneously disclose a subset of attributes, the client chooses a new random element from the issuer's secret key and generates a new pseudonym.
- Signature verification
  Upon receiving a signature σ over message m the verifier checks whether a number of equations between the signing function, attributes, and the sampled secret key hold. If it is the case, then the verifier recomputes the attribute values and accepts the signature if the credential request verifies on attribute values and the user's secret key. This is the main difference between the Identity Mixer and X.509 credential that enables efficient proofs of signature and attribute possession without revealing the signature and (chosen) attribute values themselves. It allows for ensuring that the knowledge transferred is not revealed while guaranteeing that the signature over some attributes is genuine and that the user has the relevant credential secret key.
- Pseudonymous signature generation
- Pseudonymous signature verification

3.6.2 X.509 Certificate-Based Identities

The X.509 certificate is a digital certificate that verifies that a public key belongs to the person, computer, or service identity contained within the certificate using the widely established international X.509 public key infrastructure (PKI) standard [69].

To avoid such linkability, new X.509 certificates must be used each time, resulting in complex key management as well as communication and storage overhead. Furthermore, in some circumstances, it is critical that not even the CA providing the certificates can trace

all transactions to the user. Idemix aids in the avoidance of linkability with regard to both the CA and the verifiers, as even the CA is unable to link evidence to the original credential. Neither the issuer nor the verifier can identify if two proofs came from the same credential (or two separate ones) [68].

The figure below illustrates the difference between X509 certificates-based Identity and Identity Mixer identity. Identity Mixer users can construct many public keys (pseudonyms) from the the same secret key and demonstrate that the matching secret key is signed inside a certificate by the CA, whereas in the X509 certificate-based identity, an identical set of attributes is disclosed to sing every transaction.

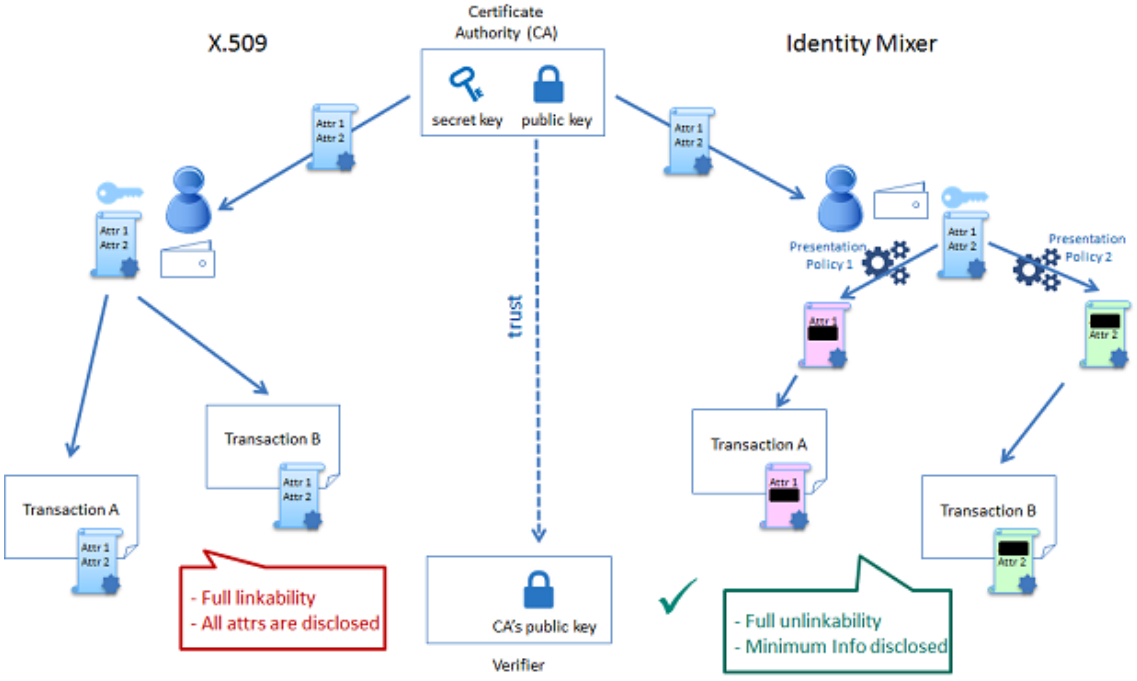

Figure 3.4: X.509 vs. Identity Mixer [70]

3.7 Proposed Setup for Achieving a Zero-Knowledge Transfer with Node.js implementation

To further enhance the idea of prevention of a DOS [62] attack based on a Petri Net directed cooperation, we conducted an analysis of a potential implementation of zero-knowledge asset transfer to ensure particular parties stay anonymous and their data is enclosed only to

chosen parties.

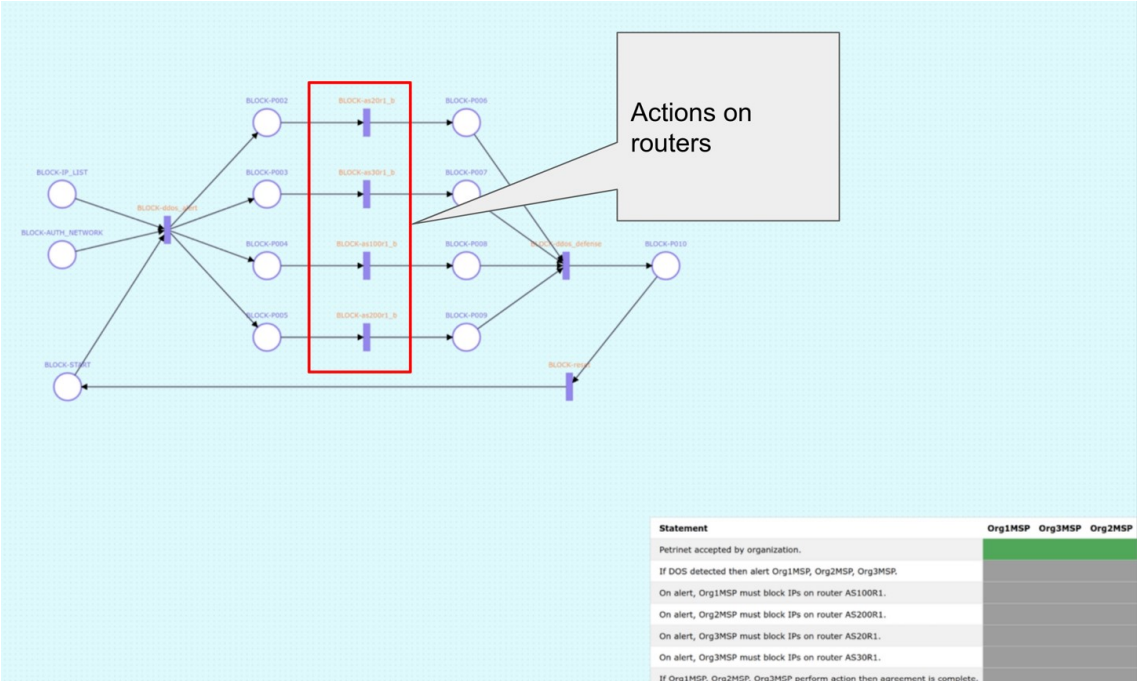

Figure 3.5: Petri Net DOS Attack Defence [62]

In the given use case [62], particular parties may have some highly private data that cannot be disclosed or may have reasons to hide that data from particular organizations. Under such circumstances, utilizing the ZKAT protocol is probably the most optimal solution. Nonetheless, potential use cases of ZKAT are numerous since data privacy is increasingly important. Other potential use cases could be blockchain-based voting systems [71], supply chains,[72] 3.5 or financial systems [73].

3.7.1 Main Components Needed to Achieve Zero-Knowledge Asset Transfer Using Identity Mixer Utilizing Node.js

The first step is a configuration of the *network.sh* file that is used to deploy and setup the whole network, set up directories for each organization, generate the cryptographic materials (keys, certificates, and configuration .yaml files).

Cryptographic materials generation is by default done by the Cryptogen tool [74]. Cryptogen creates the certificates and keys for each participating organization that is subsequently consumed by the Fabric to operate the network. The cryptogen tool processes a series of configuration files configured by the user for each organization in the "organizations/cryptogen" directory, and based on these files, it generates the crypto material for each organization specified in the "organizations" directory.

Cryptogen creates the following files and keys for each organization participating in the network:

- client files: certificates, public key, private key, and configuration files
- organization's admin certificates, public key, private key, and configuration files
- ca's certificates public key, private key, and configuration files
- MSP's public key, private key and configuration files, and configuration files

This script brings up a Hyperledger Fabric network for testing smart contracts and applications. It also creates a channel or multiple channels and deploys chaincode on them. The required modification in the *network.sh* file would utilize the $Idemixgen\ ca-keygen$ command to create cryptographic materials needed for establishing an Idemix Identity instead of *cryptogen generate*.

Five main components needed to achieve ZKAT

There are 7 main components that shall be used for utilizing Idemix on the Fabric's network. These are:

- File wallet store for adding identities to the network
- Idemix wallet implementation (contains a set of identities)

- Idemix identity implementation
- Idemix identity provider
- Idemix certificate generator
- Gateway implementation (suitable for Idemix usage)
- Network Implementation

File wallet store

First of all, after creating all cryptographic materials, a file wallet store has to be created, where users using Idemix will store their identities. These actions will happen in the wallet store, which is just a standard wallet store also used for the x509 certificate-based identities. Users add their identities to a wallet by submitting the path to credentials, their certificate, and the private key. The file wallet store is responsible for the safe storage of those identities and distributing them to appropriate calls from the network.

Idemix wallet implementation

After building a wallet store for Idemix wallets, a suitable implementation of a single Idemix wallet needs development. In the current state, a suitable implementation is non-existent. This wallet implementation should be responsible for serializing the identity and putting it into the Wallet Store object, which would be returned to the organization's certificate authority for further usage. The second important role would be the deserialization of identities that are utilized for a node that wants to endorse a transaction on the network.

Idemix certificate generator

This component would be responsible for creating Idemix credentials based on the generated cryptographic materials. The main component of the Idemix certificate generator is the signature algorithm used to create the certificate. As mentioned before, 3.6.1 the signing function used to build a certificate will sample the public key to reveal only the attributes that can be disclosed, ensuring other participants that the given information is true. The basic building block of the protocols used to issue credentials in Idemix is the Camenisch-Lysyanskaya signature scheme [75], which is used to generate identity certificates. The certificate is generated based on a number of arguments, the validity begin date,

validity end date, name of the identity, public key information, signature function, and the attributes that are to be revealed. Finally, the certificate that is returned can be utilized to sign transactions.

Camenisch-Lysyanskaya signature scheme serves as the foundation for Idemix-based identities. The scheme employed is founded on the strong RSA[1] assumption and the decisional Diffie-Hellman [77] assumption modulo, a safe prime product. It enables a user to unlinkable demonstrate credential possession as many times as needed without involving the issuing organization. Furthermore, it prevents the misuse of anonymity and provides the option to revoke anonymity for specific transactions. The scheme's authors also claim that it provides separability: all organizations can choose their cryptographic keys independently of one another.

Idemix identity provider

The previously mentioned certificate generator is a sub-part of the identity provider. The identity provider is a framework for adjusting the identity to different forms needed on the network. It is utilized to convert the cryptographic materials into Privacy Enhanced Mail (PEM) format files to build the gateway, as well as set the user context. It is a component used to create and operate on identity information.

Building a gateway implementation suitable for utilizing Idemix identites

To connect a user to the network utilizing an Idemix identity, a suitable gateway implementation needs to be developed in a way that will enable handling Idemix data. This means setting user context variables across transaction invocations within a smart contract and building a connection client based on identity data.

Building an application gateway based on the Idemix-based identity.

The final step of using Idemix identity to submit transactions to the network needs to be executed in the application part of the organization, where organizations interact with the deployed blockchain network. Each application that is deployed to interact with the network based on the chaincode has to have a gateway built before any interaction is possible. This

[1]In cryptography, the strong RSA assumption states that the RSA problem is intractable even when the solver is allowed to choose the public exponent e (for e $\geq$ 3). More specifically, given a modulus N of unknown factorization and a ciphertext C, it is infeasible to find any pair (M, e) such that $C \equiv M^e \ mod \ N$ [76].

gateway establishes the endorsing peer from the gateway peer's organization that executes the interaction and identity that will be used and specifies additional details, such as timeouts or the service discovery details, provided to understand the current view of the network.

```
function gateway("args"){
    # Load an existing wallet holding identities used to access the network
    Path walletDirectory = get_path("idemix_wallet");
    Wallet wallet = Wallets.newFileSystemWallet(walletDirectory)

    # Path to a common connection profile describing the network
    Path network_config = Paths.get("configfile.json")

    # Configure the gateway connection used to access the network
    Gateway.Builder builder = Gateway.createBuilder()
        .identity(wallet, "zkat_user")
        .networkConfig(networkConfigFile);

    # Create a gateway connection
    try (Gateway gateway = builder.connect()) {

        # Obtain a smart contract deployed on the network.
        Network network = gateway.getNetwork("channel");
        Contract zkat_contract = network.getContract("zkat_contract");

        # If the connection is successful, we can continue with the Idemix
        # identity and hide our data while submitting transactions to the
    network

        zkat_contract.createTransaction("transfer_assets").submit("...")
    } catch(...)
```

Listing 3.1: Connecting to the newtork via Gateway, using ZKAT identity [78]

According to our research, submitting a transaction using a suitable Idemix-based identity is sufficient to achieve zero-knowledge asset transfer. Hence this is the last step of this setup.

3.7.2 Description of Idemix Identities Usage

All of the components listed above should contribute to a smooth process of creating an Idemix identity and using it to transact and interact with other organizations on the network. A high-level description of the process is described below.

1. Setup

Using the Identity Mixer to generate all needed cryptographic material. This is done using the $Idemixgen\ ca{-}keygen\ {-}u\ Organization1\ {-}{-}admin{-}e\ "zkatuser"{-}r\ xxxx$ command.

2. Enrollment and Identity Issuance

Credential issuance is a mutual exchange between the user and the issuer. The issuer receives input from its secret and public keys, as well as user attribute values. As input, the user provides the issuer's public key and a user secret. The following steps contribute to the issuance protocol [79]:

1. Issuer sends a random nonce to the user.

2. User generates a credential request with the issuer's public key, user secret, and nonce as input. The request consists of a promise to keep the user secret (which can be viewed as a public key) and a zero-knowledge proof of knowledge of the user's secret key. The issuer receives the credential request from the user.

3. Zero-knowledge proof is verified by the issuer. If the request is valid, the issuer generates a credential for the user by signing the commitment to the secret key along with the attribute values and returning it to the user.

4. User checks the issuer's signature and saves the credential, which consists of the signature value, the randomness used to construct the signature, the user secret, and the attribute values.

The Identity Mixer membership service provider, Idemixgen, is utilized to build user secrets and issue credentials. Currently, only certain attributes are available, "Organization Unit", "Role", "enrollment ID", and "revocation" items.

3. Signing Transactions

Idemix signature is a signature of knowledge [80] that signs a message and demonstrates the user secret signed inside a credential. Some of the credential's properties can be selectively exposed, or alternative claims about credential features can be demonstrated without revealing them explicitly. Currently, only selective attribute disclosure is supported.

4. Transaction Signature Verification

The Identity Mixer signature is verified using the message being signed and the public key of the issuer [79].

Flow of Idemix Usage

According to our research, the flow that needs to be implemented should look as follows. A user receives an Idemix (using the $Idemixgen\ ca-keygen\ -u\ Organization1\ --admin-e$ "$zkatuser$" $-\ r\ xxxx$ command) enrollment certificate, which is a signature on a set of attributes and a commitment to the user's private key.

Following that, a commitment to the user's private key is defined as a pseudonym and can be considered as a public key. The fundamental distinction between this certificate and the X509 certificate is that the user can construct many public keys (pseudonyms) from the same secret key and demonstrate that the matching secret key is signed inside a certificate by the CA. The certificate authority blindly signs the message, by singing a pseudonym.

Using this single certificate, a user can sign messages unlinkable while selectively revealing the attributes. Instead of presenting the certificate in its entirety, the user generates a zero-knowledge proof of possession of the certificate that conceals all information while still being verifiable with the CA who issued the certificate's public key.

As a result, in the case of Idemix, a signing identity is not the original certificate, but a new unlinkable zero-knowledge proof of the certificate, which is also referred to as the pseudonymous identity.

3.8 Limitations of ZKAT in Hyperledger Fabric Framework

Currently, to the best of our knowledge, there does not exist any implementation of zero-knowledge asset transfer within the Node.js runtime. Too many major components are missing. These are gateway builder functions for Idemix and identity providers for Idemix. Idemixgen does not generate all the files and keys required by the Fabric components such as CA and MSP to fully implement a functioning framework for zero-knowledge asset transfer. According to our research, the most advanced developments of Idemix and ZKAT in Hyperledger Fabric have been made in a Golang implementation [81]. Nonetheless, this solution is

still lacking, and there does not exist an out-of-the-box signing implementation for Idemix, as well as wallet builders have not been fully developed. Furthermore, the functionalities of Idemix are very limited and in the current state, and any industrial implementations are seemingly unachievable. The main limitations are the following, firstly, Idemix organizations are not able to approve a chaincode definition to commit to the channel. This implies that if a channel has two Idemix-based MSP organizations and two organizations based on a default membership service provider, the majority to approve chaincode will not be achievable. Furthermore, signing a transaction by a peer is impossible utilizing Identity Mixer. Only signature verification is possible in the current state. This unables a complete and safe zero-knowledge asset transfer [68].

3.9 Problems Encountered During the Implementation Phase of the Proof-of-Concept

The initial plan of this project was aiming to use zero-knowledge transfer to execute Petri Nets for blockchain coordination, however, a number of impediments made it unattainable in the given timeframe. As mentioned in the previous sections, zero-knowledge asset transfer would highly likely be possible on HLF if mechanisms for handling wallets and identities were in place, and ready to use. From what is currently presented on Hyperledger's wiki page, this is a part of the future development plans [82]. Therefore this is the main problem that was encountered, a sufficient development of these components would undoubtedly take much longer than allocated for the purpose of this project with the given experience and knowledge on Hyperldeger Fabric mechanisms. A list of all the other obstacles is presented below:

- Very limited documentation of zero-knowledge transfers on Hyperledger, almost non-existing
- Little work on similar topics that would utilize Idemix identities
- Lack of implementation of components responsible for handling Idemix wallets on the Hyperledger network
- Stagnation of development of Hyperledger and Idemix
- Unfortunately, my beginner-level knowledge and experience on the topic of Hyperledger, was an additional obstacle.

Chapter 4

Project Summary & Evaluation

Currently, the aspect of unlinkable and anonymous identities participating in the blockchain networks is rather in the early stages of research and wider usage. Undoubtedly, the problem of zero-knowledge asset transfer will be one of the areas in the blockchain field with a lot of attention, due to countless potential use cases and the rising importance of protecting data privacy in the current world. Together with the wider adoption of blockchain-based solutions in various industries, privacy-enhancing mechanisms will act as crucial.

In this thesis, I addressed the problem of zero-knowledge asset transfer by utilizing the Idemix tool to enhance data privacy of the data published to the ledger by participating organizations, as well as turning the publishing identities anonymous and unlinkable for the potentially untrusted network participants. In the analyzed used cases of a DOS attack or a supply chain, utilizing this solution can have a significant impact on increasing the safety of participants, and the participating organizations can rest assured that their data is concealed and safe.

Due to the current state of the Idemix framework, successful storage of Idemix-based identities and utilizing them is not ready for use and still requires a lot of effort from the Fabric developer. In a three months time frame, it turned out that a fully working implementation of the suggested setup is not viable due to limited Hyperledger Fabric experience as well as the complexity behind, for instance, the Camenisch-Lysyanskaya signature scheme. Furthermore, in the allocated time, five weeks were given for learning Fabric, JavaScript, and the associated cryptography-related topics. Nonetheless, a number of missing parts of the Idemix environment have been proposed and listed to achieve zero-knowledge asset transfer

successfully.

4.1 Research Questions Answers

RQ1. Can we use the zero-knowledge protocol to maintain privacy in coordinating multi-domain applications

In principle, upon resumption of work on Idemix and its related components, successful utilization of the zero-knowledge asset transfer on Hyperdledger Fabric will be possible. Nonetheless, currently, the implementation is lacking, and not only it is not feasible, but the whole Hyperledger Fabric framework also is not prepared for Idemix-based identities. The list of limitations that was described in 3.8 is not only related to Idemix but to the framework as a whole. For instance, the approval of chaincode definitions by the Idemix identities which is not possible due to the lack of hierarchies of Idemix-based Certificate Authorities.

Lastly, upon successful implementation of components needed to use Idemix identities on the Fabric network, zero-knowledge asset transfer in coordinating multi-domain applications would be possible because the current implementation is lacking. Nonetheless, efficient and flexible zero-knowledge-based coordination is far away from only implementing the Idemix components needed to operate identities.

Sub-RQ1.1. How can we set up zero-knowledge asset transfer to add privacy in coordinating multi-domain applications?

The detailed description of missing components is presented in 3.7. Nonetheless, the given setup will only facilitate a very limited usage and possibilities of zero-knowledge asset transfer, this is due to the current stage of development of Hyperledger fabric and associated mechanisms, for instance, lack of certificate authorities hierarchy for Idemix identities.

RQ2. Is the current advancement of Hyperledger Fabric developed enough to provide an efficient zero-knowledge asset transfer?

Conclusions that can be drawn from what has been extensively researched, indicate that due to numerous limitations that unable an efficient zero-knowledge asset transfer 3.8. Ad-

ditionally, Hyperledger developers were contacted for the purpose of the project and they claim that Idemix in the current state is not usable.

Furthermore, Idemix is a part of Hyperledger that was put on hold and there is no ongoing work happening. Additionally, there are many issues on the HLF main repository that remain unsolved until today, despite the fact that Idemix was first introduced in 2017 with the 1.1 version of Hyperledger.

RQ3. What degree of anonymity and unlinkability is possible to achieve with zero-knowledge technologies in Hyperledger Fabric?

With the current state of Hyperledger Fabric, anonymity and unlinkability are unachievable and there do not exist any alternative techniques. Data can on the network be hidden from other organizations by the usage of private data collections and usage of multiple channels between the organizations that trust each other, but a general network where organizations can prove to be owners of data is not achievable.

Currently, only a high-level overview of the whole concept has been introduced by IBM and the community developing the Hyperledger Fabric Infrastructure.

4.2 Future Work

Future work coming from this research would be an implementation of the setup described 3.7and its particular components to achieve zero-knowledge asset transfer with the Fabric's Identity Mixer. The main important and the most complex component is the Idemix certificate generator 3.7.1 that shall be used to generate an identity revealing only the parts of data that can be disclosed.

Provided that enough time, of around two months, is added to the project, a successful implementation seems highly likely, and a certain degree of data privacy and anonymity is viable to achieve. The main focus should be put on the development of an efficient Idemix certificate generator, as it is the most complex component of the whole setup.

Additionally, an interesting concept that would provide a very high degree of data concealment for privacy and safety purposes would be a combination of zero-knowledge asset transfer and homomorphic encryption 2.5.5. In this potential use case scenario, organizations

could, for instance, hide some parts of the data published to the ledger while encrypting the remaining parts to allow network participants to perform computations on the encrypted data. For instance, two users can secretly agree on a price for an item if their secret account balances are encrypted with additively homomorphic promises. The first user can then use homomorphism to subtract this sum from their balance and add it to the balance of the second user, demonstrating to everyone that the sum added to the second balance equals the sum deducted from the first [83].

Lastly, the additional focus could be put on improving the possible ratio of Idemix-based and default Membership Service Providers. One can imagine situations in which most, or possibly all organizations want to remain anonymous and publish data in an unlinkable manner, currently, it is not possible and it is claimed to be a challenging issue. This can possibly be achieved by implementing a hierarchy of Idemix Certificate Authorities.

Bibliography

[1] Yuosre F Badir, Remi Founou, Claude Stricker, and Vincent Bourquin. Management of global large-scale projects through a federation of multiple web-based workflow management systems. *Project Management Journal*, 34(3):40–47, 2003.

[2] Mehrdad Hajizadeh, Nima Afraz, Marco Ruffini, and Thomas Bauschert. Collaborative cyber attack defense in sdn networks using blockchain technology. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 487–492. IEEE, 2020.

[3] Usecases — amdex. `https://amdex.eu/usecases/`. (Accessed on 01/15/2022).

[4] Jun Yan, Yun Yang, and Gitesh K Raikundalia. Swindew-a p2p-based decentralized workflow management system. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 36(5):922–935, 2006.

[5] Jun Yan, Yun Yang, and Gitesh K Raikundalia. A decentralised architecture for workflow support. In *Proc. 7th Int. Symposium on Future Soft. Technology (ISFST02), CD ISBN*, pages 4–916227, 2002.

[6] Giacomo Piccinelli, Anthony Finkelstein, and Scott Lane Williams. Service-oriented workflow: The dysco framework. In *Euromicro Conference*, pages 291–291. IEEE Computer Society, 2003.

[7] Gustavo Alonso, Divyakant Agrawal, Amr El Abbadi, and Carl Mohan. Functionality and limitations of current workflow management systems. *IEEE expert*, 12(5):105–111, 1997.

[8] Xudong He and Tadao Murata. High-level petri nets—extensions, analysis, and applications. In *The Electrical Engineering Handbook*, pages 459–475. Elsevier, 2005.

[9] Alaa Hamid Mohammed, Alaa Amjed Abdulateef, and Ihsan Amjad Abdulateef. Hyperledger, ethereum and blockchain technology: A short overview. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–6. IEEE, 2021.

[10] Christian Rondanini, Barbara Carminati, Federico Daidone, and Elena Ferrari. Blockchain-based controlled information sharing in inter-organizational workflows. In *2020 IEEE International Conference on Services Computing (SCC)*, pages 378–385. IEEE, 2020.

[11] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Blockchain in internet of things: challenges and solutions. *arXiv preprint arXiv:1608.05187*, 2016.

[12] Junjun Lou, Qichao Zhang, Zhuyun Qi, and Kai Lei. A blockchain-based key management scheme for named data networking. In *2018 1st IEEE international conference on hot information-centric networking (HotICN)*, pages 141–146. IEEE, 2018.

[13] Roman Beck, Jacob Stenum Czepluch, Nikolaj Lollike, and Simon Malone. Blockchain–the gateway to trust-free cryptographic transactions. 2016.

[14] Ryan Henry, Amir Herzberg, and Aniket Kate. Blockchain access privacy: Challenges and directions. *IEEE Security & Privacy*, 16(4):38–45, 2018.

[15] Dan Wang, Jindong Zhao, and Yingjie Wang. A survey on privacy protection of blockchain: the technology and application. *IEEE Access*, 8:108766–108781, 2020.

[16] Ledger — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/`. (Accessed on 09/04/2022).

[17] Julija Golosova and Andrejs Romanovs. The advantages and disadvantages of the blockchain technology. In *2018 IEEE 6th workshop on advances in information, electronic and electrical engineering (AIEEE)*, pages 1–6. IEEE, 2018.

[18] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[19] Dodo Khan, Low Tang Jung, Manzoor Ahmed Hashmani, and Moke Kwai Cheong. Empirical performance analysis of hyperledger lts for small and medium enterprises. *Sensors*, 22(3):915, 2022.

[20] Blockchain explained: How does a transaction get into the blockchain? — euromoney learning. https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain. (Accessed on 04/14/2022).

[21] Ying-Chang Liang. Blockchain for dynamic spectrum management. In *Dynamic Spectrum Management*, pages 121–146. Springer, 2020.

[22] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[23] Stefano Bistarelli, Gianmarco Mazzante, Matteo Micheletti, Leonardo Mostarda, Davide Sestili, and Francesco Tiezzi. Ethereum smart contracts: Analysis and statistics of their source code and opcodes. *Internet of Things*, 11:100198, 2020.

[24] How to write smart contracts — 10clouds. https://10clouds.com/blog/defi/how-to-write-smart-contracts/. (Accessed on 09/04/2022).

[25] Leo Maxim Bach, Branko Mihaljevic, and Mario Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. Ieee, 2018.

[26] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[27] Understanding the raft consensus algorithm: an academic article summary. https://www.freecodecamp.org/news/in-search-of-an-understandable-consensus-algorithm-a-summary-4bc294c97e0d/. (Accessed on 08/23/2022).

[28] Kathleen E. Wegrzyn Eugenia Wang. Types of blockchain: Public, private, or something in between — blogs — manufacturing industry advisor — foley & lardner llp. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between, August 2021. (Accessed on 04/15/2022).

[29] Hyperledger – open source blockchain technologies. https://www.hyperledger.org/.

[30] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[31] Hyperledger fabric: Most essential features & applications you need to know — upgrad blog. `https://www.upgrad.com/blog/hyperledger-fabric-features-applications/`. (Accessed on 06/29/2022).

[32] Ledger — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.4/ledger/ledger.html`. (Accessed on 09/04/2022).

[33] The ordering service — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html`. (Accessed on 06/29/2022).

[34] Private and confidential transactions with hyperledger fabric - ibm developer. `https://developer.ibm.com/tutorials/cl-blockchain-private-confidential-transactions-hyperledger-fabric-zero-knowledge-pro` (Accessed on 06/13/2022).

[35] Smart contracts and chaincode — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/smartcontract/smartcontract.html`. (Accessed on 06/29/2022).

[36] Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[37] Barbara Carminati, Christian Rondanini, and Elena Ferrari. Confidential business process execution on blockchain. In *2018 ieee international conference on web services (icws)*, pages 58–65. IEEE, 2018.

[38] Barbara Carminati, Elena Ferrari, and Christian Rondanini. Blockchain as a platform for secure inter-organizational business processes. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 122–129. IEEE, 2018.

[39] van der Aalst Wil, Jörg Desel, and Andreas Oberweis. *Business process management: Models, techniques, and empirical studies.* Springer, 2003.

[40] WMP van der Aalst. Workflow management: net-based concepts, models, techniques, and tools (wfm'98): proceedings of the workshop, june 22, 1998, lisbon, portugal. 1998.

[41] Amit Sheth. Nsf workshop on workflow and process automation in information systems: state-of-the-art and future directions. *ACM SIGGROUP Bulletin*, 18(1):23–24, 1997.

[42] Andrea Pinna, Roberto Tonelli, Matteo Orrú, and Michele Marchesi. A petri nets model for blockchain analysis. *The Computer Journal*, 61(9):1374–1388, 2018.

[43] Zhentian Liu and Jing Liu. Formal verification of blockchain smart contract based on colored petri net models. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 555–560. IEEE, 2019.

[44] Nejc Zupan, Prabhakaran Kasinathan, Jorge Cuellar, and Markus Sauer. Secure smart contract generation based on petri nets. In *Blockchain Technology for Industry 4.0*, pages 73–98. Springer, 2020.

[45] Wil MP Van Der Aalst. Three good reasons for using a petri-net-based workflow management system. In *Information and Process Integration in Enterprises*, pages 161–182. Springer, 1998.

[46] David Hollingsworth and UK Hampshire. Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19(16):224, 1995.

[47] Clarence A Ellis and Gary J Nutt. Modeling and enactment of workflow systems. In *International Conference on Application and Theory of Petri Nets*, pages 1–16. Springer, 1993.

[48] Wil MP Van der Aalst. Verification of workflow nets. In *International Conference on Application and Theory of Petri Nets*, pages 407–426. Springer, 1997.

[49] Wil MP Van Der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *Business Process Management*, pages 161–183. Springer, 2000.

[50] Jörg Desel. Validation of process models by construction of process nets. In *Business Process Management*, pages 110–128. Springer, 2000.

[51] Khodakaram Salimifard and Mike Wright. Petri net-based modelling of workflow systems: An overview. *European journal of operational research*, 134(3):664–676, 2001.

[52] C. Adam Petri and W. Reisig. Petri net. *Scholarpedia*, 3(4):6477, 2008. revision #91647.

[53] Wikipedia contributors. Petri net — Wikipedia, the free encyclopedia, 2022. [Online; accessed 7-July-2022].

[54] Zero knowledge proofs — ali baba cave example. `https://en.wikipedia.org/wiki/Zero-knowledge_proof#The_Ali_Baba_cave`. (Accessed on 09/01/2022).

[55] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[56] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4):198–205, 2021.

[57] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 acm sigsac conference on computer and communications security*, pages 2087–2104, 2017.

[58] Bulletproofs — stanford applied crypto group. `https://crypto.stanford.edu/bulletproofs/`. (Accessed on 08/23/2022).

[59] Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943. IEEE, 2018.

[60] Sharath Yaji, Kajal Bangera, and B Neelima. Privacy preserving in blockchain based on partial homomorphic encryption system for ai applications. In *2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW)*, pages 81–85. IEEE, 2018.

[61] Yuxuan Wang, Fengji Luo, Zhaoyang Dong, Ziyuan Tong, and Yichen Qiao. Distributed meter data aggregation framework based on blockchain and homomorphic encryption. *IET Cyber-Physical Systems: Theory & Applications*, 4(1):30–37, 2019.

[62] Cees de Laat Reginald Cushing. Paola Grosso. Petrinet smart contracts for multi-domain network collaboration, University of Amsterdam 2021.

[63] Dentons - the privacy paradox in blockchain: best practices for data management in crypto. https://www.dentons.com/en/insights/articles/2022/june/9/the-privacy-paradox-in-blockchain-best-practices-for-data-management-in-crypto. (Accessed on 08/31/2022).

[64] Private data — hyperledger-fabricdocs main documentation. https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html. (Accessed on 06/13/2022).

[65] M Harikrishnan and KV Lakshmy. Secure digital service payments using zero knowledge proof in distributed network. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 307–312. IEEE, 2019.

[66] Zeba Mahmood and Jusas Vacius. Privacy-preserving block-chain framework based on ring signatures (rss) and zero-knowledge proofs (zkps). In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6. IEEE, 2020.

[67] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[68] Msp implementation with identity mixer — hyperledger-fabricdocs main documentation. https://hyperledger-fabric.readthedocs.io/en/release-2.2/idemix.html. (Accessed on 08/11/2022).

[69] What is an x.509 certificate? https://www.techtarget.com/searchsecurity/definition/X509-certificate. (Accessed on 08/11/2022).

[70] Msp implementation with identity mixer — hyperledger-fabricdocs master documentation. https://hyperledger-fabric.readthedocs.io/en/release-1.2/idemix.html. (Accessed on 08/17/2022).

[71] Ceyhun Onur and Arda Yurdakul. Electanon: A blockchain-based, anonymous, robust and scalable ranked-choice voting protocol. *arXiv preprint arXiv:2204.00057*, 2022.

[72] Jiang-ping Liu and Ri-geng Wu. A petri net-based supply chain system. *International Journal of Online Engineering*, 14(11), 2018.

[73] George Kappos and Ania M Piotrowska. Extending the anonymity of zcash. *arXiv preprint arXiv:1902.07337*, 2019.

[74] cryptogen — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/commands/cryptogen.html`. (Accessed on 08/12/2022).

[75] Masayuki Tezuka and Keisuke Tanaka. Improved security proof for the camenisch-lysyanskaya signature-based synchronized aggregate signature scheme. In *Australasian Conference on Information Security and Privacy*, pages 225–243. Springer, 2020.

[76] Strong rsa assumption - wikipedia. `https://en.wikipedia.org/wiki/Strong_RSA_assumption`. (Accessed on 08/29/2022).

[77] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, pages 93–118. Springer, 2001.

[78] Overview (fabric-gateway-java 2.2.5 api). `https://hyperledger.github.io/fabric-gateway-java/release-2.2/index.html`. (Accessed on 09/04/2022).

[79] Msp implementation with identity mixer — hyperledger-fabricdocs main documentation. `https://hyperledger-fabric.readthedocs.io/en/release-2.2/idemix.html`. (Accessed on 08/17/2022).

[80] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference*, pages 78–96. Springer, 2006.

[81] Idemix signer for go client · issue #242 · hyperledger/fabric-gateway. `https://github.com/hyperledger/fabric-gateway/issues/242`. (Accessed on 08/16/2022).

[82] Project plan - identity mixer support for both fabric gateway sdk for java and fabric gateway client api for java - hyperledger mentorship program - hyperledger foundation. `https://wiki.hyperledger.org/display/INTERN/Project+Plan+-+Identity+Mixer+Support+for+both+Fabric+Gateway+SDK+for+Java+and+Fabric+Gateway+Client+API+for+Java`. (Accessed on 09/11/2022).

[83] Fabrice Benhamouda, Shai Halevi, and Tzipora Halevi. Supporting private data on hyperledger fabric with secure multiparty computation. *IBM Journal of Research and Development*, 63(2/3):3–1, 2019.