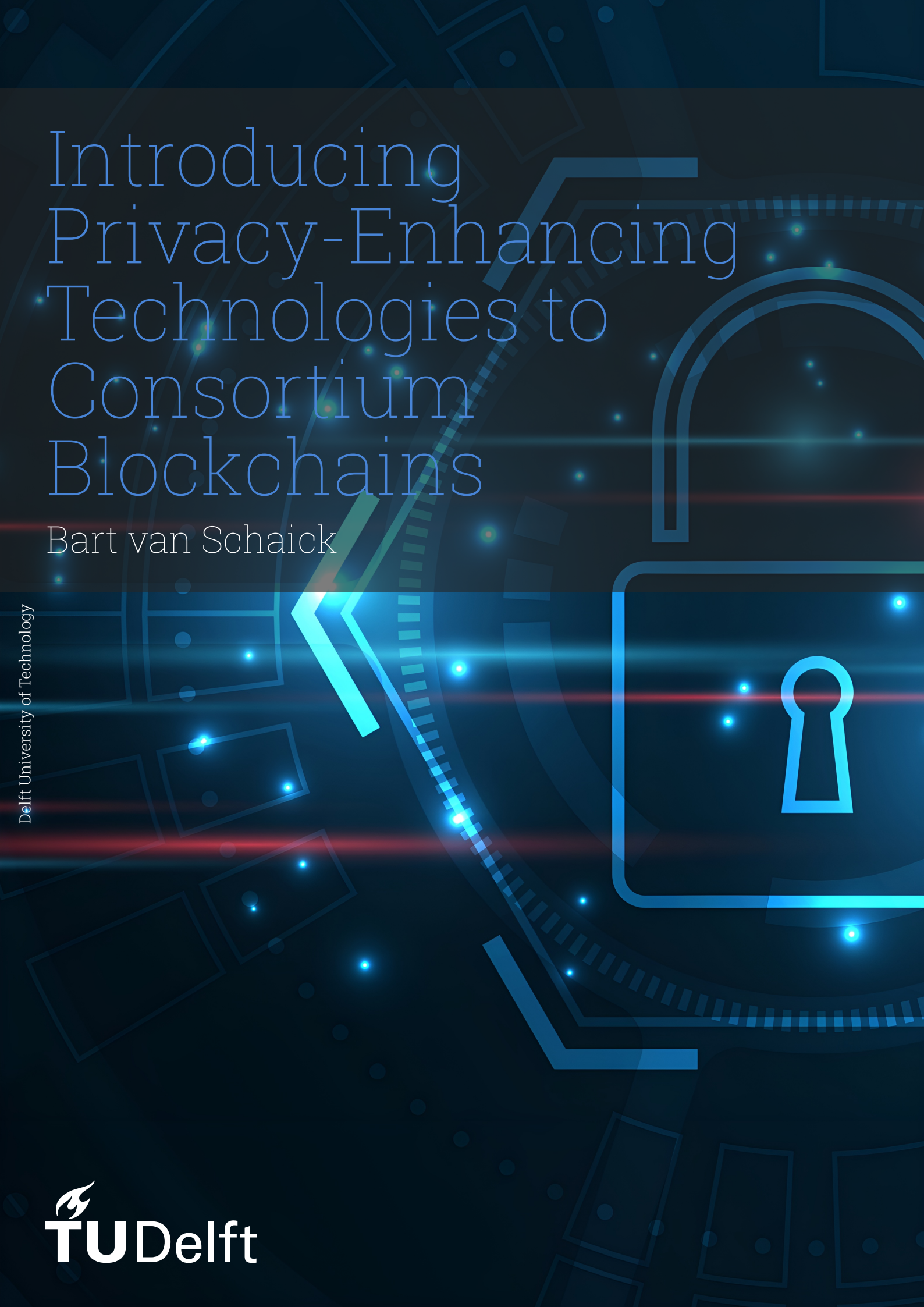
Introducing Privacy-Enhancing Technologies to Consortium Blockchains

Bart van Schaick

Delft University of Technology



TU Delft

Introducing Privacy-Enhancing Technologies to Consortium Blockchains

by

Bart van Schaick

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday February 20, 2023 at 10:00 AM.

Student number:	4489357	
Project duration:	January 31, 2022 – February 20, 2023	
Thesis committee:	Prof. Georgios Smaragdakis,	TU Delft, advisor
	Dr. Kaitai Liang,	TU Delft, supervisor
	Dr. Stefanie Roos,	TU Delft
	Dr. Roland Kromes,	TU Delft, supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis was written as the final fulfillment of the requirements of the degree of Master of Science in Computer Science at the Delft University of Technology. As such, it concludes my academic career and marks the start of a career in research or industry.

During my side job as a software developer at a wholesale company, I stumbled upon a logistical problem that required a level of trust and transparency lacking in centralized database solutions. It sparked my interest in blockchain technology and specifically consortium blockchains. Furthermore, I have always found cryptography to be deeply fascinating. I still remember the illuminating feeling the moment the Diffie-Hellman key exchange protocol clicked in my head.

I would like to thank my supervisor Kaitai Liang for providing me with the chance to work on this project, even though I did not follow the intended cybersecurity track. Furthermore, I want to thank George Smaragdakis for his intermediate feedback on my work.

Most of all, I want to thank Roland Kromes. Roland, thank you for the wonderful way you supervised me over the past year. The tips and insights you have given me throughout this research project have shaped the thesis as it is. Especially during the last few months, when I struggled to articulate the research, you pulled me through multiple times. I wish you all the best in your future academic career.

Finally, I would like to thank you, the reader, for your interest in my work. I hope you will find the remainder of this thesis informative and engaging.

Bart van Schaick
Delft, February 2023

Summary

Blockchain technology has revolutionized the way data is stored, managed, and shared across various industries. Its decentralized nature and immutability make it highly attractive in use cases that require transparency, integrity, and accountability. However, some applications demand confidentiality, necessitating the development of permissioned/consortium blockchains that try to strike a balance between transparency and privacy. Hyperledger Fabric is a permissioned blockchain that has gained popularity due to its modular architecture, performance, and scalability. Despite its strengths, the current set of privacy-enhancing features leaves room for improvement.

Therefore, this master thesis aims to explore the potential of introducing various privacy-enhancing technologies to Hyperledger Fabric, including Dynamic Searchable Symmetric Encryption (DSSE), Multi Authority Attribute-based Encryption (MA-ABE), and Trusted Execution Environments (TEE). Based on the promising results of our study, we decided to implement DSSE and MA-ABE. The combination of blockchain technology with TEE was ruled out after the thorough analysis of two research papers on the subject.

Our main result is the first implementation of a provable secure DSSE scheme in the context of consortium blockchains. Moreover, we developed a blockchain-enabled MA-ABE system that utilizes a novel and generic approach to foreign function invocation. Finally, the thesis discusses unexplored challenges in the field of logistics related to electronic consignment notes used in road transport. To address these issues, we designed a blockchain architecture that incorporates our developed privacy-enhanced technologies.

Contents

1 Introduction

Blockchain is well-known as the technology behind cryptocurrencies such as bitcoin and Ethereum. Unlike traditional databases, blockchain stores data across a network of distributed computing nodes. Transactions on the network are grouped together and recorded in so-called blocks, which are cryptographically linked together, creating an ever-growing chain of blocks. The cryptographic link and the distributed nature of the blockchain ensure that records cannot be altered retroactively. The integrity offered by this immutable ledger is considered one of the key benefits of blockchain technology. Additionally, the technology features a decentralized and distributed network, eliminating a single point of failure or need for a governing authority.

Initially, blockchain was proposed as a solution to transfer funds directly between individuals, bypassing the need for traditional financial institutions[1]. Nevertheless, due to the aforementioned properties, the technology has the potential to be disruptive in other industry sectors, including healthcare, supply chain logistics, and more. Smart contracts facilitate the adoption in the industry. They are small programs stored on the blockchain network that can verify and enforce the execution of business processes in a transparent and secure manner. However, the transparent and open-access nature of traditional blockchains also poses a threat to data confidentiality. All the information shared on the network is publicly accessible by anyone, including bad actors. Moreover, for many enterprise applications, the performance of public blockchains is insufficient due to their high latency and low transaction throughput.

1.1. Consortium Blockchains

Consortium blockchains address the privacy issues and performance deficits surrounding their public counterparts [2]. In a consortium blockchain, access is limited to and controlled by a small group of users or organizations (the consortium). The privacy concerns that existed with public blockchains are addressed in this manner as sensitive information stored in the ledger is only disclosed to consortium members. Furthermore, as the number of nodes in a permissioned network is generally much smaller and malicious users can quickly be identified and excluded, the distributed decision-making process is much more streamlined. Efficient coordination between nodes is especially important in the context of blockchain for achieving consensus. Consensus refers to the process of reaching an agreement among participants about the current state of the data on the blockchain. A more efficient consensus mechanism means higher throughput and lower latency for transactions in the network.

Hyperledger Fabric (HLF) is a framework for developing enterprise-grade permissioned blockchains. The HLF architecture features a modular design [3], allowing enterprises to tailor it to their specific requirements through the use of pluggable components, such as consensus, privacy, and membership services. Due to the smaller number of nodes and permissioned design, achieving consensus is much simpler, resulting in better performance compared to public blockchain solutions. Data privacy is achieved through the use of channels. Channels provide an additional layer of privacy within the main network, designed to facilitate private and confidential transactions between participants. Chapter 2

contains more background information on Hyperledger Fabric.

The modular architecture of HLF makes it a perfect candidate for building consortium blockchains that satisfy a broad range of industry use cases. However, while HLF offers private transactions out of the box over channels, the access control offered by this approach is limited. Moreover, many use cases require transactions to be (partially) transparent to other participants of the network or even outsiders, such as in the case of an audit. In summary, the current set of privacy-enhancing technologies offered by HLF is limited but its modular design encourages extending the system.

1.2. Introducing Privacy-enhancing Technologies

Privacy-enhancing technologies (PET) are a set of tools, systems, and methods used to prevent the processing of confidential data without loss of functionality. The term PETs encompasses a wide range of technologies, including anonymization, secure multi-party computation, trusted execution environments, zero-knowledge proofs, and various forms of encryptions such as homomorphic, attribute-based, and searchable encryption. Given the privacy and functionalities offered, PETs seem like a natural extension to consortium blockchains. Furthermore, we will show that merging these technologies benefits the privacy and functionality provided by PETs.

For this thesis, we have selected three privacy-enhancing technologies that we believe are well-suited to combine with Hyperledger Fabric. A short description of the chosen technologies and rationale behind our selection follows:

Searchable Encryption A form of symmetric encryption that allows a data owner to outsource encrypted documents to an untrusted cloud server while retaining the ability to search over the data. We envision a searchable encryption scheme in which the cloud server is replaced by our consortium blockchain and query functionality is implemented by smart contracts. This way, a data owner is certain they receive the correct documents.

Attribute-based Encryption A type of public-key encryption where users can only decrypt a ciphertext if their set of attributes matches the attributes of the ciphertext. Data owners can specify an access policy during encryption that determines the set of users with decryption ability. In this way, attribute-based encryption can provide fine-grained access control to confidential data shared on the ledger.

Trusted Execution Environment A secure environment for executing code that prevents unauthorized entities outside of the environment from tampering with the execution. Typically, solutions are hardware-based and involve an isolated part of the processor to which program instruction can be loaded. The confidentiality and integrity offered by these techniques could be beneficial to smart contract execution as well.

1.3. Research Questions

The goal of the research presented in this thesis is to elevate the privacy and functionalities offered by current consortium blockchain solutions, by integrating certain privacy-enhancing technologies. To accomplish this goal we ask ourselves the following research question:

How to enhance the trust and functionality of consortium blockchain applications through the deployment of privacy-enhancing technologies?

In order to answer the research question, we formulate 5 sub-questions.

1. How can smart contracts provide search functionality over dynamically encrypted data stored on the blockchain?
2. How can we share encrypted data only with a subset of users in the blockchain consortium?
3. How can we employ trusted execution environments to enforce correct smart contract execution and what are the limitations?
4. What are the benefits and limitations of a privacy-enhanced blockchain solution that incorporates the technologies mentioned in the previous research questions?
5. What new use cases can be tackled by this privacy-enhanced blockchain-based system?

1.4. Contributions and Outline of Thesis

The research presented in this thesis brings several contributions to the fields of consortium blockchains and privacy-enhanced technologies.

- Chapter 3 features our implementation of a secure and efficient dynamic searchable symmetric encryption scheme in Hyperledger Fabric. To the best of our knowledge, this is the first time a dynamic construction of a proven searchable symmetric encryption scheme is ported to a consortium blockchain. We have submitted a paper containing this result to the ESORICS 2023 conference.
- In chapter 4, we present our blockchain-enabled multi-authority attribute-based encryption scheme. In contrast with many other attribute-based encryption implementations in blockchains, our solution requires no central authority. Moreover, we present a generic approach for creating a foreign function interface with little overhead.
- Chapter 5 contains a literature review of the applications of trusted execution environments in blockchain technology. Particularly, we will highlight an approach that allows Hyperledgder Fabric peers/nodes to keep a secret.
- Finally, in chapter 6, we will show how the privacy-enhanced blockchain-based information system we developed can tackle a highly-prevalent problem encountered in the domain of logistics.

Chapter 2 provides background information on Hyperledger Fabric, for readers unfamiliar with the technology. The chapters on searchable encryption, attribute-based encryption, and trusted execution environments, respectively, chapter 3, chapter 4, and chapter 5 are self-contained and can be read independently or in an alternative order. Chapter 6 assumes knowledge about the previous chapters. Finally, chapter 7 provides a conclusion to our work.

2

Hyperledger Fabric

This chapter provides a brief background on Hyperledger Fabric. We will discuss its approach to implementing a permissioned blockchain, consensus, identity management, and many more. The aim of this chapter is to set the stage for the subsequent chapters that will present the main contributions of our research. All the information presented in this chapter has been selected from the Hyperledger Fabric whitepaper [3] and its latest documentation[1].

2.1. General Overview

The Hyperledger[2] project is an open-source collaborative effort hosted by the Linux Foundation, created to advance cross-industry blockchain technology. Fabric is one of the projects within the Hyperledger project. Like other blockchain technologies, it has a distributed network of nodes that maintain a ledger and uses smart contracts to interact with the data. Unlike other blockchain solutions, Fabric is private and permissioned. Authorization is required for users to participate in the network. As a result, consensus can be reached more efficiently and transaction throughput is increased. Furthermore, Fabric's use of channels allows a group of participants to transact confidentially. The high performance and privacy offered by Fabric combined with its modular and extensible design make it particularly suitable for enterprise solutions.

2.2. Network Structure

At a conceptual level, a Hyperledger Fabric network is made up of multiple stakeholders that correspond to real-world organizations. The network provides ledger and smart contract functionality to users. Smart contracts contain the business logic in the consortium; they define assets and the set of instructions to interact with them. When an asset is modified, the smart contract generates a transaction that is subsequently distributed to all participants and immutably recorded on the ledger. A private communication channel can be established between a selected group of organizations within the network, enabling confidential transactions among its members. Each channel has a separate ledger and smart contract definitions assigned to it.

Concretely, the blockchain network consists of several nodes with various responsibilities, hosted by the participating organizations. Each organization can deploy one or multiple peer nodes. Peers are fundamental in HLF as they manage the ledgers and smart contracts for the organizations. Fabric uses the term *chaincode* to refer to an implementation of smart contracts on a peer. Each peer hosts an instance of the ledgers and chaincodes defined for the channels it is affiliated with. Through a peer, clients can query the ledger and perform transactions by invoking chaincode.

[1] Hyperledger Fabric documentation: `https://hyperledger-fabric.readthedocs.io/en/latest/`
[2] Official Hyperledger project website: `https://www.hyperledger.org/`

2.3. Identity Management

All actors in an HLF blockchain require a digital identity to consume the services provided by the network. The identity is encapsulated in an X.509 certificate and determines the exact permissions over resources and access to information the actor has. For an actor to participate in the network, her identity must be issued by an authority trusted by the network. Certificate authorities (CAs) are commonly used in internet security protocols and public key infrastructures to fulfill the role of the identity issuer. The certificates dispensed by a CA are digitally signed using the CA's private key.

Membership service providers (MSPs) are affiliated with an organization in the network and determine which CAs are trusted by the organization. The certificate issued by a CA is merely a way by which the identity of an actor can be proved. Instead, the MSP is a mechanism to turn this provable identity into a network role. In chapter 4, we will show how HLF's MSPs and CAs can prevent collusion between users in our attribute-based encryption implementation.

2.4. Ledger

The ledger is compromised of the world state and the blockchain. The blockchain is a transaction log that records all the changes made to the ledger. Transactions are collected in a block that is appended to the blockchain by cryptographically linking it to the previous block. The world state is implemented as a key-value database that holds the current value or state of assets. The current state of an asset in the database is determined by the history of transactions involving this asset in the blockchain. The addition of the world state database allows a peer to query the value of assets in an efficient manner while the immutability of the ledger is maintained.

2.5. Transaction Flow

The first step in updating the ledger is initiated by a client. The client prepares a transaction proposal which is submitted to one or more peers. The peers simulate the transaction on their local version of the ledger and produce a transaction result including a response value, read set, and write set (more on this later). If the transaction executes successfully, the peer endorses the transaction result by signing it, no updates are made to the ledger at this point. The endorsed transaction result is then sent to the ordering service, which groups transactions in blocks and distributes them to the peers in the network. Peers validate the transactions received by the orderer and commit them to their local copy of the ledger. In the end, the blockchain is updated and the transaction is finalized.

The read set and write set are sets of key/value pairs representing assets that were read or updated during the endorsement. The write set contains a list of unique keys and their latest value set by the transaction. The read set contains a list of unique keys and the version of the committed value read during simulation. During transaction validation, a committing peer uses the read set and write set of a transaction to check its validity. A transaction is considered valid if the version of each key present in the read set matches the current version of the key in the committed ledger state. The read set and write set ensure that transactions can happen concurrently in Fabric, but prevents an inconsistent state from occurring.

3

Searchable Encryption

Transparency is one of the key features of blockchain technology. However, in the case of confidential data, transparency is not necessarily considered a good trait. Encryption forms a simple means to an end, however, by encrypting information a data owner losses the ability to search over the data. Dynamic searchable symmetric encryption (DSSE) is a type of encryption that allows a data owner to search and update encrypted data efficiently. DSSE schemes have been successfully implemented in cloud service providers in the past. In this chapter, we propose the first implementation of a provably secure DSSE scheme in the context of blockchain. We not only add search functionality to the blockchain through our implementation, but we also obtain verifiable search results with our approach. We evaluate the performance of our implementation and discuss several points of improvement.

3.1. Introduction

Following the Software-as-a-Service trend in enterprise computing, outsourcing data storage to cloud service providers (CSP) has gained widespread adoption, due to its promise of high scalability and performance at low operational costs. However, data owners are understandably hesitant with sharing privacy-sensitive data with an untrusted third party. While conventional encryption addresses these privacy concerns, it also results in the loss of critical data processing functionalities such as search.

Searchable symmetric encryption (SSE) is a form of encryption, first proposed by Song *et al.* in 2000, which retains search functionality [4]. With Searchable Encryption, a data owner is able to share encrypted confidential documents with an untrusted CSP. Later on, the data owner can send the CSP an encrypted search instruction. The CSP can both perform the instruction and obtain its results but cannot learn any new information about the contents of either.

Searchable symmetric encryption has gained lots of interest in the research world, and many improvements have been proposed over the initial scheme by Song *et al.* [4]. While much attention has been devoted to improving the privacy, efficiency, security, verifiability, and query expressiveness of schemes, a particularly interesting research direction is that of dynamic searchable symmetric encryption (DSSE). In contrary with static schemes like [4], DSSE introduced by Kamara *et al.* allows users to update their set of documents efficiently [5].

Static SSE schemes have successfully been applied in a blockchain context before [6, 7]. As mentioned previously, the verifiability of search results is an important aspect of SSE schemes. These schemes benefit from the verifiability that blockchain provides out-of-the-box in the form of smart contracts. Only very recently have dynamic SSE implementations in blockchain been proposed, but they either did not contain a security proof [8] or required a redactable blockchain for rewriting history [9].

In this chapter, we present the first known dynamic searchable symmetric encryption scheme implemented in a blockchain context that is secure against adaptively chosen keyword attacks. In section 3.2, related work is discussed. In section 3.3, a formal definition of DSSE and security are given. Section 3.4 explains our implemented scheme and system in detail. Finally, in section 3.5, a performance analysis of the system is given.

3.2. Related Work

Searchable symmetric encryption can be achieved in full generality using the pioneering work of Goldreich and Ostrovsky on oblivious random access machines (ORAM) [10]. Their main result is an efficient simulation of an arbitrary program on an oblivious machine, for which the sequence in which it accesses memory locations is independent of the input given. A simple oblivious SSE scheme is then created by ORAM simulation of a data structure that supports fast searches on document collections (e.g. an inverted index). While ORAM schemes achieve strong privacy guarantees, their high computational and storage costs make them only practical for small datasets.

Song *et al.* showed that a more efficient solution for searchable encryption exists for a weaker security model with small leakage of the client's access pattern [4]. In this first practical scheme, a ciphertext of a document contains special two-layered encryption constructs for each word in the document. To search for a keyword, the client generates a trapdoor that strips the outer layer of the construct and asserts whether the inner layer is of the correct form. The server then has to sequentially scan all ciphertexts, resulting in a search complexity linear in the number of words per document. Furthermore, the construction of this scheme leaks the word distribution of underlying plaintexts and is therefore vulnerable to statistical attacks. Nevertheless, the authors proved their scheme to be a secure pseudo-random generator. Later on, Kamara *et al.* would even prove this initial scheme to be indistinguishable against chosen plaintext attacks (IND-CPA) secure [5].

Goh recognized IND-CPA security was not sufficient to describe the security of searchable encryption schemes, as it does not consider leakage coming from trapdoors or queries [11]. They proposed a new notion of security specific to the context of SEE, which they named indistinguishable against chosen keyword attacks (IND-CKA). An SSE scheme is considered IND-CKA secure if an adversary is unable to distinguish indexes from two equally-sized documents. A secure index is a data structure that can be queried with a trapdoor τ_w generated for a word w to test whether the index contains w without

revealing w or the index's contents. In the same paper, they propose their own SSE scheme that assigns an index to each document during construction. Their index implementation relies on Bloom filters (BF), a probabilistic data structure based on hashing, which can efficiently test set membership. The search complexity of this scheme is therefore linear in the number of documents stored by the server. Goh proved their scheme to be IND-CKA secure.

Chang and Mitzenmacher introduced a new simulation-based definition of security for SSE schemes, with stronger properties compared to Goh's IND-CKA game-based definition [12]. Their definition would consider trapdoor leakage, and guarantee indistinguishability for two documents of unequal size. Goh would later introduce IND2-CKA security, which does protect index size but still does prevent trapdoor leakage. The scheme proposed by Chang and Mitzenmacher uses dictionaries instead of Bloom filters for building the indexes, yielding the same linear search complexity as Goh's implementation.

Curtmola *et al.* showed the security definitions as proposed by [11, 12] to be inadequate in capturing the security of SSE schemes [13]. They pointed out two limitations in the previous definitions: (1) leakage from the trapdoor construction was not explicitly captured; (2) the power of the adversary was implicitly limited. Starting with the first issue, while previous definitions acknowledged leakage caused by the search results being revealed to the server, this access pattern leakage was not treated appropriately. Furthermore, search pattern leakage (i.e. whether a search query is being repeated) was not incorporated. Curtmola *et al.* addressed this issue by incorporating explicitly allowed leakage in their IND-CKA1 security definition. Later on, Chase and Kamara would revisit this approach and generalize the definition for any kind of leakage [14].

To address the second issue, in addition to IND-CKA1 security, Curtmola *et al.* proposes a new adaptive adversarial model IND-CKA2. Unlike in the non-adaptive setting, the adaptive definition allows an adversary to choose their queries as a function of previously obtained trapdoors and search results. This model is better suited to represent real-world scenarios in which the adversary is a server that can learn new information during every new round of interaction with a client. Curtmola *et al.* also proposes two new constructions and proves them to be secure under their new security definitions. The idea is to have an inverted index per distinct word in the database instead of per document. A trapdoor is generated by applying a pseudo-random function to the keyword being queried. This approach achieves a search time linear in the number of documents containing the keyword, which is optimal. The difference between the two proposed schemes is that the first one is only secure in the non-adaptive setting, while the second scheme is also secure against an adaptive adversary, at the cost of higher communication overhead and server storage requirements.

IND-CKA2 is considered a strong security definition for SSE schemes, as it adequately describes the information leakage coming from index and trapdoor construction in the adaptive setting. The definition protects user privacy against a passive adversary but is unable to detect a malicious server withholding or forging search results. To provide security in the active adversary model, Kurosawa and Ohtaki introduce a verifiable SSE scheme [15]. In this scheme, a client is able to link a query and server response due to MAC tags being included in both encrypted indexes and trapdoors. Linear search complexity is achieved due to the number of table lookups on the server, each resulting in a MAC that requires verification on the client.

Kamara *et al.* propose the first practical dynamic SSE scheme with efficient updates of documents [5]. Their construction extends the inverted index approach of Curtmola *et al.* by issuing a deletion array, an encrypted data structure that allows for the efficient deletion of a file by tracking all pointers in other lookup tables and arrays corresponding to the file. Furthermore, homomorphic encryption is used to modify pointers in the data structures without revealing their contents to the server. This combination allows the efficient addition and deletion of files without having to re-index the entire data collection. The construction preserves the optimal search time of its predecessors. As the IND-CKA2 security definition only considers static schemes, an extension of the definition called dynamic CKA2-security is given to account for the new set of operations. The scheme is proven to be dynamic CKA2 secure in the random oracle model.

3.3. Definitions

In this section, we formalize the dynamic searchable symmetric encryption scheme and present our main security definition. Before doing so, however, we would like to clarify some of the notation used in our definitions. Furthermore, appendix A serves as a quick reference guide to the notation and terminology assumed throughout this section.

A searchable symmetric encryption scheme considers two participants: a client in the possession of a set of documents or files; and an untrusted server with the ability to store this data. The goal of an SSE scheme is to store the client's files in an encrypted format on the server's storage while retaining the ability of the client to (efficiently) search over the files without revealing any information about their contents. In contrast with previous definitions [5, 13] that implicitly create an association between files and keywords by considering each file as a sequence of words, we make this association explicit. Let $\boldsymbol{D} = \{D_1, \ldots, D_n\}$ be a set of private documents and let $\sigma_w(D_i)$ be the predicate that signifies an association between keyword $w \in W$ with the encrypted document D_i, where W is the universe of keywords. We define the set of files $\boldsymbol{f} = \{f_1, \ldots, f_n\}$ complementary to $\boldsymbol{D}$, with $f_i = \{w \in W \mid \sigma_w(D_i)\} \in \mathcal{P}(W)$ to be the set of documents associated to D_i by σ. Note that by our explicit definitions of $\boldsymbol{f}$ and σ, the definition of document collection $\boldsymbol{D}$ is kept intentionally abstract, such that $D_i \in \boldsymbol{D}$ can be of any data type (e.g. binary files, photos, health records, etc.). Let $\mathsf{id}(D)$ be the unique identifier of document $D \in \boldsymbol{D}$, where the identifier can be any string, such as a name, hash, or memory location. Let $\boldsymbol{f}_w = \{f_i \in \boldsymbol{f} \mid \sigma_w(D), w \in W, D \in \boldsymbol{D}\}$ the set of files in $\boldsymbol{f}$ that contain w. Similarly, let I_w be the set of file identifiers that contain w.

Definition 3.1 (Searchable Symmetric Encryption (SSE)) *A searchable symmetric encryption scheme is a tuple of four polynomial-time algorithms* $\mathsf{SSE} = (\mathsf{Gen}, \mathsf{Setup}, \mathsf{Token}, \mathsf{Search})$ *such that:*

> $K \leftarrow \mathsf{Gen}(1^k)$: *is a probabilistic algorithm that takes as input a security parameter* $k \in \mathbb{N}$*. It outputs a secret key* K*.*
>
> $I \leftarrow \mathsf{Setup}(K, \boldsymbol{f})$: *is a probabilistic algorithm that takes as input as secret key* K *and set of files* $\boldsymbol{f}$*. It outputs an encrypted index* I*.*
>
> $\tau \leftarrow \mathsf{Token}(K, w)$: *is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a keyword* w*. It outputs a search token* τ*.*
>
> $I_w := \mathsf{Search}(I, \tau)$: *is a deterministic algorithm that takes as input a secret key an encrypted index* I *and a search token* τ*. It outputs a set of file identifiers* I_w*.*

An SSE *scheme is correct if for all* $k \in \mathbb{N}$*, for all* K *generated by* $\mathsf{Gen}(1^k)$*, for all* $\boldsymbol{f} \subseteq \mathcal{P}(W)$*, for all* I *output by* $\mathsf{Setup}(K, \boldsymbol{f})$*, and for all* $w \in W$*,*

$$\mathsf{Search}(I, \mathsf{Token}_k(w)) = I_w.$$

Contrary to the definition for SSE given by [13], we do not consider the encryption and decryption algorithms, respectively Enc and Dec, as part of our definition. Our definition of SSE only considers algorithms directly related to the encrypted index operations. In section 3.4 it will become clear why this distinction was made. For practical use of this SSE scheme, we require a second non-searchable symmetric encryption scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Intuitively, a private-key encryption scheme SKE is indistinguishable against chosen plaintext attacks (IND-CPA) if a polynomial-size adversary that can adaptively query an encryption oracle is unable to distinguish pairs of ciphertexts based on the message they encrypt. Formal definitions of both symmetric encryption schemes and IND-CPA security are provided in definition A.3 and A.4, respectively. We note that common symmetric key block ciphers such as AES-CTR (Counter mode of operation) satisfy the IND-CPA security definition.

A searchable symmetric encryption scheme as defined in definition 3.1 is considered static, as it does not allow for the addition and removal of files. For practical purposes, dynamic SSE schemes are of more interest, because in addition to search, they support altering the encrypted index by insertion and deletion of documents.

Definition 3.2 (Dynamic Searchable Symmetric Encryption (DSSE)) *A dynamic symmetric searchable encryption scheme is a tuple of eight polynomial-time algorithms* $\mathsf{DSSE} = (\mathsf{Gen}, \mathsf{Setup}, \mathsf{SearchToken}, \mathsf{AddToken}, \mathsf{DeleteToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Delete})$ *such that:*

$K \leftarrow \mathsf{Gen}(1^k)$: *is a probabilistic algorithm that takes as input a security parameter* $k \in \mathbb{N}$. *It outputs a secret key* K.

$I \leftarrow \mathsf{Setup}(K, \boldsymbol{f})$: *is a probabilistic algorithm that takes as input as secret key* K *and set of files* $\boldsymbol{f}$. *It outputs an encrypted index* I.

$\tau_s \leftarrow \mathsf{SearchToken}(K, w)$: *is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a keyword* w. *It outputs a search token* τ_s.

$\tau_a \leftarrow \mathsf{AddToken}(K, f)$: *is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a file* f. *It outputs an add token* τ_a.

$\tau_d \leftarrow \mathsf{DeleteToken}(K, f)$: *is a (possibly probabilistic) algorithm that takes as input a secret key* K *and a file* f. *It outputs a delete token* τ_d.

$I_w := \mathsf{Search}(I, \tau_s)$: *is a deterministic algorithm that takes as input a secret key, an encrypted index* I, *and a search token* τ_s. *It outputs a set of file identifiers* I_w.

$I' := \mathsf{Add}(I, \tau_a)$: *is a deterministic algorithm that takes as input a secret key, an encrypted index* I, *and an add token* τ_a. *It outputs a new encrypted index* I'.

$I' := \mathsf{Delete}(I, \tau_d)$: *is a deterministic algorithm that takes as input a secret key, an encrypted index* I, *and a delete token* τ_d. *It outputs a new encrypted index* I'.

A DSSE *scheme is correct if for all* $k \in \mathbb{N}$, *for all* K *generated by* $\mathsf{Gen}(1^k)$, *for all* $f \in \mathcal{P}(W)$, *for all* I *output by* $\mathsf{Setup}(K, \boldsymbol{f})$, *for all* $w \in W$, *and for all* I' *resulting from any sequence of* Add *or* Delete *operations on* I,

$$\mathsf{Search}(I', \mathsf{SearchToken}_k(w)) = I_w.$$

Note that in the research domain of searchable symmetric encryption, the terms TrapDoor and SearchToken are used interchangeably to define an operation run by the data owner to generate a search trapdoor. In this work, we opted for the latter term, to have consistent naming across all the token-generating functions.

A security definition for a DSSE scheme should, in addition to the leakage of the access pattern consider leakage caused by the dynamic algorithms. We adopt the definition by [5] for adaptive semantic security against chosen-keyword attacks in the dynamic setting. They define a tuple $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4)$ of four stateful functions to capture the leakage of their scheme. These leakage functions correspond to the leakage caused by the Setup, SearchToken, AddToken, and DeleteToken algorithms, respectively. The parameters for the functions are dependent upon the construction of the scheme. Now follows the definition for dynamic CKA2-security.

Definition 3.3 (Dynamic CKA2-security) *Let* $\mathsf{DSSE} = (\mathsf{Gen}, \mathsf{Setup}, \mathsf{SearchToken}, \mathsf{AddToken}, \mathsf{DeleteToken}, \mathsf{Search}, \mathsf{Add}, \mathsf{Delete})$ *be a dynamic searchable symmetric encryption scheme. Consider the following probablistic experiment, where* $\mathcal{A}$ *is a stateful adversary,* $\mathcal{S}$ *is a stateful simulator, and* $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, *and* $\mathcal{L}_4$ *are stateful leakage functions:*

$\boldsymbol{Real}_{\mathcal{A}}(k)$**:** *The challenger runs* $\mathsf{Gen}(1^k)$ *to generate a key* K. $\mathcal{A}$ *outputs* $\boldsymbol{f}$ *and receives* $I \leftarrow \mathsf{Setup}_K(\boldsymbol{f})$ *from the challenger. The adversary makes a polynomial number of adaptive queries* $\{w, f_1, f_2\}$. *For each query* q, *it receives from the challenger a search token* $\tau_s \leftarrow \mathsf{SearchToken}_K(w)$, *an add token* $\tau_a \leftarrow \mathsf{AddToken}_K(f_1)$, *or a delete token* $\tau_d \leftarrow \mathsf{DelToken}_K(f_2)$. *Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

$\boldsymbol{Ideal}_{\mathcal{A},\mathcal{S}}(k)$**:** $\mathcal{A}$ *outputs* $\boldsymbol{f}$. *Given* $\mathcal{L}_1$, $\mathcal{S}$ *generates and sends* I *to* $\mathcal{A}$. *The adversary makes a polynomial number of adaptive queries* $\{w, f_1, f_2\}$. *For each query* q, *the simulator is given either* $\mathcal{L}_2(\boldsymbol{f}, w)$, $\mathcal{L}_3(\boldsymbol{f}, f_1)$, *or* $\mathcal{L}_4(\boldsymbol{f}, f_2)$ *and receives* τ_s, τ_a *or* τ_d *respectively. Finally,* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

DSSE *is said to be* $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4)$*-secure against adaptive dynamic chosen-keyword attacks if for all polynomial-size adversaries* $\mathcal{A}$, *there exists a polynomial-size adversary* $\mathcal{S}$ *such that*

$$|Pr\left[\boldsymbol{Real}_{\mathcal{A}}(k) = 1\right] - Pr\left[\boldsymbol{Real}_{\mathcal{A},\mathcal{S}}(k) = 1\right]| \leq \mathsf{negl}(k),$$

3.4. Blockchain-based DSSE

Our system is an implementation of the DSSE scheme presented by Kamara *et al.* in a blockchain context [5]. Their construction relies on four encrypted data structures to efficiently make queries with minimal leakage. We will adopt these four data structures in our design to achieve the same asymptotically optimal query performance and efficient updates (more on this in section 3.4.1). Furthermore, they proved their scheme IND-CKA2 secure against adaptive adversaries.

While leakage of the access patterns is restricted, the scheme only considers passive adversaries who will not deviate from the defined protocol. An active adversary can still supply the user with incorrect results. A simple example would be a server that always returns an empty list as search output. To tackle this issue, verifiable SSE was proposed by Kurosawa and Ohtaki, but their solution is neither dynamic nor has it efficient search complexity [15]. A dynamic verifiable scheme proposed by Bost *et al.* achieves optimal search time but requires a client to keep track of its search queries, resulting in a space complexity linear in the number of keywords Bost *et al.*

In this work, we take an entirely different approach by introducing a blockchain, more specifically Hyperledger Fabric (HLF), to tackle the issue of verifiability. Blockchain peers take over the role of the cloud service provider in this setting. The search, add and delete functionality is implemented in chaincode. In addition, the InitLedger smart contract is defined to set up all data structures on-chain, given an encrypted index. Correct execution of these operations is enforced by the blockchain through consensus.

3.4.1. Implementation

To showcase the feasibility of our blockchain-based DSSE concept, we implemented DSSE from scratch in the Go programming language[1]. As mentioned previously, our implementation relies heavily on the construction given by [5]. To apply their ideas effectively in the context of HLF, some alterations to the algorithms and data structures used were required to safeguard the security and performance of the implementation. These modifications will be discussed in detail in this section.

The scheme of [5] has an optimal search time complexity of $O(|\boldsymbol{f}_w|)$. A search time linear in the number of files that contain the queried keyword w was first achieved by [13], through a technique regarded as the *inverted index approach*. Kamara *et al.* use the same approach, but instead of using a single lookup table and array like [13], their dynamic scheme requires four data structures: a search table T_s, deletion table T_d, search array A_s, and deletion array A_d. The function(s) of these data structures will become clear in the example given in section 3.4.2. To achieve said search complexity, access and update operations on lookup tables and arrays should have expected constant time complexity.

The current status of assets stored in the HLF ledger is projected as the world state. This world state is implemented by means of a highly efficient key-value store with constant time search, insert, and delete operations. However, the key-value store is not an in-memory database on blockchain peers. Accessing an element from an array stored as an asset on HLF requires $O(n)$ time, with n the array's capacity, as the whole array needs to be loaded. We get around this problem by storing each element contained within T_s, T_d, A_s, and A_d as a separate key-value pair. For the lookup tables, this conversion is self-evident, for key-value pair (k, v) and lookup table T with $T[k] = v$, we obtain key-value pair $(\mathsf{id}(T)||k, v)$, where $\mathsf{id}(T)$ is a predefined string identifying T. For arrays, this conversion is less obvious, but we opt for a similar method as we observe that the construction by [5] does not utilize the arrays' in-order traversal or contiguous memory properties. Let A be an array, and let $A[i] = v$ denote the element stored in A at index i, then we obtain key-value pair $(\mathsf{id}(A)||i, v)$, where $\mathsf{id}(A)$ is a predefined string identifying A.

3.4.2. An Illustrative Example

Figure 3.1 contains an overview of the different parties in our system and an example of the interactions between them. We consider the toy example of an educational institution. A data owner can be thought of as the institution's content management system, while data users can be faculty members or students. The files in this context can be considered the marks of a test or assignment. The role of the blockchain is multi-functional, in addition to providing the users with verifiable results for their queries, its immutability can also be used to prevent tampering with the marks.

[1] The official Go programming language website: `https://go.dev/`

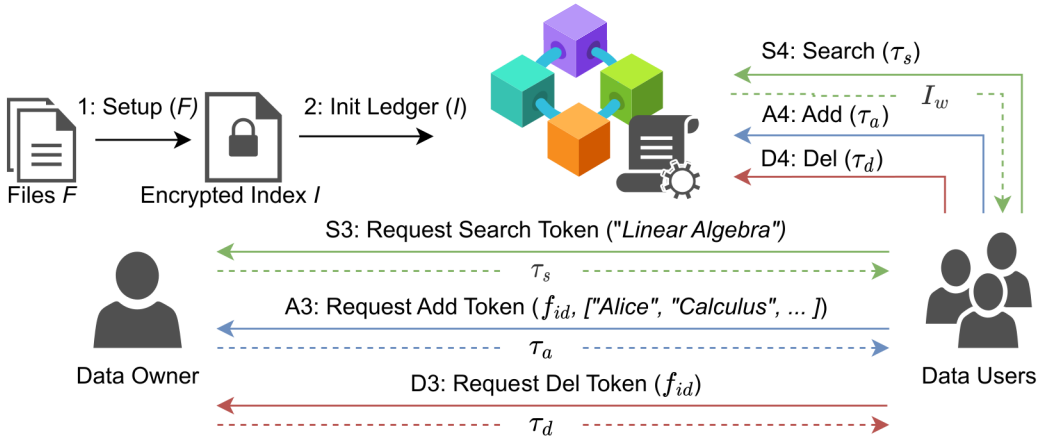


Figure 3.1: Overview of our blockchain-based DSSE system including example interactions between parties.

Initialization To start working with the system an initialization step is required. This step needs to be run only once by the data owner. The data owner starts by running the Gen algorithm and obtains the secret key K. He then uses this K together with an initial (or empty) set of files **f** to run the Setup algorithm and obtains the encrypted index I (1). At last, the data owner saves I on the blockchain by invoking the InitLedger smart contract function (2).

Search Now consider a student who wants to obtain all their marks for the course of "$Linear Algebra$". The student starts by requesting a search token from the data owner (S3), specifying the course name as the keyword w. The data owner receives the request, checks the student is authorized, and uses its secret key K to generate a search token τ_s through the SearchToken function. The data user receives τ_s and invokes the SearchWord function in the chaincode (S4). In the end, the data owner receives a list of file identifiers I_w previously associated with w.

Addition A teacher might want to upload a new mark for the *Calculus* course obtained by student *Alice*. A request containing the mark, identified by f_{id} and the list of keywords ["$Alice$","$Calculus$"], is sent to the data owner (A3). With its secret key k, the data owner proceeds by running the AddToken function and obtains an add token τ_a that is returned to the data user. The last step in to making the file queryable is invoking the AddFile chaincode with τ_a (A4).

Deletion Suppose now a teacher has made a mistake and wants to remove a mark from the system. To update the search index, they first request a deletion token τ_d from the data owner specifying the mark to be deleted by identifier f_{id} (D3). The data owner checks the authorization of the user and creates the delete token τ_d by running the DeleteToken function with its security key K. The data user receives τ_d and invokes the Delete chaincode to remove the mark identified by f_{id} and all its associations from the search index (D4).

3.5. Performance Analysis

To evaluate the performance of our blockchain-based DSSE system we made a distinction between the operation performed by the client, and the chaincode run by the blockchain peer. In addition to the execution time of operations, we will also measure latency, success rate, and transaction size when applicable. For all experiments involving chaincode execution, a Hyperledger Fabric blockchain network with a single channel containing four peers and one orderer was deployed. All the nodes were hosted by the same machine to minimize the cost of network traffic. All experiments are run on an 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz 8 CPU computer with 16 GB RAM. The implementation of the DSSE chaincode is available in the accompanying code repository[2].

For all our benchmarks we used the open source English words dataset[3]. During our experiments, we selected the unit of measurement to be the number of *associations*. The set of all associations of a file

[2]DSSE repository: `https://github.com/bartvsdev/thesis-sse`
[3]English words dataset: `https://github.com/dwyl/english-words`

f is the set compromised of all unique keywords w associated with f. Unlike [5] we did not consider real-world datasets of emails, documents, and MP3 files, as we want to highlight the influence of the number of associations on the performance, rather than the total document size. The represented measurements are the mean result of 10 executions.

Size of keyword set (KB)	25	50	75	100
Index Size (MB)	7.94	15.76	25.43	33.83
Execution Time Client (ms)	115.66	204.68	300.31	398.82
Execution Time CC (s)	16.91	34.15	48.47	72.50

Table 3.1: Setup - Encrypted index size, generation, and chaincode initialization.

Table 3.1 shows the size, client generation time, and chaincode initialization time for an encrypted index generated for a variable number of associations. The execution for the client and chaincode (CC) indicate the time required for generating the encrypted index on the client and initializing the index on the ledger, respectively. For this synthetic experiment, a keyword set of size defined in the first row is generated from the dataset. The keywords are evenly distributed and associated across 5 made-up files. Note that the total number of associations is equal to the number of keywords, and the deciding factor for index size. As expected, the size, generation time, and chaincode initialization time of the encrypted index grow linearly with the number of associated keywords. The chaincode execution time is high, in the order of minutes in the case of a 100KB keyword set. Possible causes of this unsatisfactory result are discussed in section 3.6.

In general, we expect our dynamic SSE scheme will be initialized with an empty set of files. In our second experiment, we measured the maximum capacity of the number of keyword/file associations that could be added in this case. The maximum number of associations the encrypted index can hold is determined at **414K**. The time execution time of the chaincode to initialize an empty encrypted index of this size on the ledger is **3.2 minutes**. This upper bound on index size is determined by the default chaincode time-out period. The impact of adjusting this default time-out and alternatives for improving initialization time and index size are discussed in section 3.6.

# Input Load	1500	2000	3000	4000
Throughput	1483	1995	2678	3191
Response Time (ms)	7.96	19.33	57.39	-
Number of Rejects	0	0	0	658

Table 3.2: Search - Load testing search chaincode.

For our third experiment, the search performance is analyzed. The search procedure consists of two consecutive operations, obtaining a token through the **SearchToken** operation, and querying the ledger through the **Search** chaincode. As the DSSE construction by Kamara *et al.* indicates, the search token generation is a constant time operation, irrespective of the provided keyword [5]. The mean time for computing a search token on the client is **6.03 ms**.

In table 3.2 the throughput, response times, and in some instances the number of rejects are displayed for different query loads. The experiment was conducted as follows: 5 clients use a set of precomputed search tokens to query multiple peers in parallel for a fixed total amount of 1500, 2000, 3000, and 4000 requests per second. The throughput shows the actual number of transactions per second. This load test shows the chaincode is able to handle high demand concurrently, with fast response times and high throughput. However, at some point, the chaincode starts rejecting requests.

# Keywords / File	2	4	6	8	10
Execution Time Client (μs)	25.80	37.92	47.85	58.10	64.69
Execution Time CC (s)	2.108	2.115	2.120	2.128	2.133

Table 3.3: AddToken and **Add** - Generation of add token and chaincode for adding a file.

Table 3.3 shows the relationship between the number of keywords and execution time for adding a file. Like the search procedure, file addition consists of two steps, first generating an add token on the client

side, continued by executing a chaincode with this token. The add token generation seems to grow linearly with the number of keywords associated with the file being added. The chaincode execution time seems constant, but as we will discuss in section 3.6, this result is flawed due to a HLF property.

Deletion of a file from the HLF is again a two-step process of generating a token using the **DeleteToken** operation and subsequently calling the **Delete** chaincode. Generating a delete token is a constant time operation with mean of **25.1 μs**. Execution of the delete chaincode experiences the same problem as the chaincode for addition. The mean execution time of this chaincode averages around **2.1 s** as well.

3.6. Discussion

Our blockchain-based DSSE system shows how smart contracts can provide secure and verifiable query functionality over dynamic data stored in a decentralized ledger. The consensus property assures the data owner's encrypted data is updated according to specification. Our system is able to handle high search throughput with fast response time and this performance should scale horizontally due to the distributed nature of the network. No fair comparison can be drawn between the original scheme of Kamara *et al.* and our implementation, due to the dissimilarities in datasets and environment (in-memory vs client-server operations) [5]. Xu *et al.* very recently proposed a DSSE scheme in a blockchain context and obtained similar search performance compared to our results [8]. However, they do not provide a security proof for the construction of their system.

The experiments considering the initialization of the ledger pointed out two limitations of this operation: the long execution time and an upper bound on the index size. The default time-out causing the bound on the size of the encrypted index can be easily altered in the HLF settings. However, this modification might not be desirable. Instead, we would like to drive research into the possibility of creating a dynamically-sized index. Additionally, with a resizable index (or multiple indices) a data owner is relieved of the inconvenient task of correctly predicting the final index size in advance.

In section 3.5 we already mentioned the results from the chaincode for adding and deleting were flawed due to an HLF property. In HLF, transactions have to be approved by an endorsement peer before they are committed to the ledger [3]. During endorsement, a proposal is simulated against the endorser's local blockchain. The fundamental problem of double spending is prevented by verifying a single key-value pair is not altered in multiple transactions. The construction of [5] relies on *free* pointers, to decide on the next location to place a new association. Our implementation of the add chaincode works similarly but naturally, a free pointer corresponds to a key-value pair in our context. In chapter 2, we saw that HLF prevents multiple transactions within the same block to read and alter the same key-value pair. So while all the add operations are successfully executed and validated by peers, only a single one can be committed in the next block. Block creation takes approximately 2 seconds in our test network, which explains the chaincode execution time. The same logic applies to our chaincode implementation of the delete operation.

A trivial solution for improving the performance of the add and delete chaincode seems to reduce the block time. However, this approach seems infeasible, as our experiments were conducted in an ideal environment with a small number of peers and little communication delay. A second option would be to introduce a two-step process for the addition and/or deletion of files. During step 1, add and delete tokens are collected on the ledger and committed into a new block. Step 2 is triggered right after a collection of tokens is committed. In this step, the add and delete operations corresponding to the tokens collected in (1) are executed as a single batch. Hyperledger Fabric offers built-in events for such a scenario. While this second approach would allow for significantly higher throughput, the time between the proposal of an update and its commitment doubles. Whether such a trade-off is justifiable, likely depends on the number of update transactions in the network and the use case.

4

Attribute-Based Encryption

How do we efficiently share a secret on a consortium blockchain only with a specific set of participants? In this chapter, we showcase how attribute-based encryption (ABE) can be used to restrict the ability to decrypt a ciphertext to a subset of users in the network. Our system requires no central authority, organizations can deploy their own authority and start issuing private keys to users that reflect their attributes. Our system relies on the secure multi-authority ABE construction of Lewko and Waters [17] and performant implementation of their scheme in Rust. We developed a foreign function invocation (FFI) wrapper, based on protobuf, to make it accessible for clients. Finally, we evaluate the performance of the wrapper and show its small overhead for general use cases.

4.1. Introduction

The key exchange problem is a fundamental problem in cryptography and is defined as the challenge of two or more parties agreeing on a shared secret key over an insecure communication channel. Traditional public key encryption (PKE) schemes solve this problem using two different but mathematically related cryptographic keys. A public key is shared among multiple users and is used to encrypt messages, such that only the holder of the corresponding private key can decrypt resulting ciphertexts. However, PKE introduces a new problem as users now have to verify the authenticity of received public keys. The general solution involves deploying a public key infrastructure (PKI), in which all parties should trust a third party called the certificate authority (CA). Users can apply to the CA for a digital certificate by which they can be identified by others.

Having a PKI in place is costly and encrypting a message is a time-consuming process that includes public key certificate retrieval, certificate verification, and message encryption by the receiver's public key. Moreover, broadcasting a secret within a group is highly inefficient as all steps have to be repeated for each member of the group separately. In 1984 Shamir suggested identity-based encryption (IBE) as a solution to this problem [18]. Instead of generating a random public key like in traditional PKE, the idea of IBE is to let a publicly known string represent the user as its public key. However, [18] was unable to come up with a concrete implementation of an IBE scheme, which was only much later proposed by Boneh and Franklin in 2001 [19].

While in theory identity-based encryption solves the problem of public key distribution and authentication, it still feels cumbersome in practice. For example, when Alice wants to send a message to all the members of the sales department in her company, then she needs to know the identities of all the employees in the department and encrypt the message for each recipient separately. As the example shows, individuals are often characterized by their attributes, rather than simply their identity. This led to the introduction of the fuzzy IBE scheme by Sahai and Waters in 2005 which included attributes in the design [20]. In their design, messages are encrypted using a set of attributes. A central authority exists with whom users authenticate themselves to obtain a private key based on their attributes. The decryption of a ciphertext is only feasible for users with private keys that match the attributes of the ciphertext. Later on, this type of cryptographic primitive became known as attribute-based encryption (ABE).

Lots of improved ABE schemes have been proposed that can be categorized into two groups. The fuzzy IBE scheme proposed by Sahai and Waters is considered a key-policy ABE (KP-ABE) scheme [20]. In KP-ABE, ciphertexts are associated with sets of attributes, while users' secret keys are generated based on an access policy that defines the privileges and rights of the concerned user. In ciphertext-policy ABE (CP-ABE) schemes, first introduced by Bethencourt *et al.*, the roles are reversed [21]. In CP-ABE, ciphertexts are associated with access policies, while a user's private key is generated based on its attributes. The interest in CP-ABE seems to prevail over KP-ABE, as the data owner retains access policy determination in the former case.

ABE-enabled access control systems have seen successful implementations in the context of cloud computing and IoT applications [22, 23]. In theory, blockchain applications seem to be an even better fit for ABE, as on top of the added expressiveness and scalability that ABE offers over PKE and IBE, it provides anonymity between users. An encrypting data owner is not required to know the identity of its decrypting counterpart (in advance), as users are merely identified by their attributes. In its recent SoK on cryptography used in blockchain, Raikwar *et al.* identifies the construction of an ABE based (or ABE related) permissioned blockchain network as a research challenge [24]. Zhang *et al.* provides a comprehensive overview of blockchain technology and acknowledges that "an implementation of attribute-based encryption utilizing a blockchain approach remains to be an open challenge" [25].

Like many of its successors, the initial ABE scheme proposed by Sahai and Waters relies on a central authority for issuing private keys to users. For a decentralized network like blockchain, such a dependency on a single entity is not desirable. Sahai and Waters recognized this and asked themselves whether it was possible to create an ABE scheme "where the attributes come from multiple authorities" [20]. Chase was the first to consider a multi-authority ABE scheme, but the author's design still relied on the presence of a trusted central authority [26]. Lewko and Waters were the first to come up with a fully secured decentralized multi-authority design that is central authority free [17]. Like in single-

authority schemes, the biggest obstacle in multi-authority schemes is preventing collusion between users. In the design of Lewko and Waters this is achieved through linking private keys of different authorities by assigning each user a unique global identifier (GID). To build a secure, decentralized, collusion-resistant ABE system, this uniqueness constraint of the GID should be enforced.

In this chapter, we present our ABE-enabled consortium blockchain. In our system, any institution part of the consortium can become an authority and issue private attribute keys to users. Secrets can be securely shared with a subset of users defined by the access policy in the ciphertexts. Collusion between users is prevented by linking private keys issued by different authorities together using the blockchain identity. Our system is fully decentralized and does not require institutions and users to trust a single designated entity. Finally, we provide a working example including implementations of the authorities, chaincode, and foreign function interface that enables encryption and decryption on clients.

In section 4.2, related work is discussed. Section 4.3 provides formal definitions of concepts and problems required for the construction and proof of our multi-authority ABE implementation. In section 4.4, we present our ABE-enabled consortium blockchain implementation and discuss its security guarantees. In section 4.6, we analyze the performance of our system. Finally, in section 4.7, our results are discussed.

4.2. Related Work

In 2005, Sahai and Waters proposed their fuzzy identity-based encryption scheme, which would become known as the first attribute-based encryption scheme [20]. Unlike previous IBE schemes, Sahai and Waters identity is viewed as a collection of characteristics (attributes) instead of a unique string. Consequently, multiple users can decrypt the same ciphertext, provided there exists conformity among the attributes associated with the user's private keys and those specified during cipher encryption. Their scheme is proven secure by a reduction. In addition to data confidentiality, Sahai and Waters stress the importance of collusion-resistance in ABE schemes, i.e. no group of users should be able to collectively decrypt a ciphertext that none of the members alone could. Collusion is prevented by incorporating the user's identity in private keys generated by the authority through a user-specific random polynomial. The scheme supports threshold access policies, i.e. a user is required to possess t out of n defined attributes, and is proved selectively secure using the decisional BDH assumption (see definition 4.5).

While the authors of [20] were the first to coin the term "attribute-based encryption", their fuzzy IBE scheme was intended to provide error-tolerance to encryptions of biometric identities. The limited expressiveness of their threshold access policy in ABE reflects this original intent. Goyal *et al.* developed an ABE scheme that allowed for more fine-grained sharing of encrypted data, which they called key-policy attribute-based encryption (KP-ABE) [27]. In their scheme, ciphertexts are marked by a set of attributes, and access structures are defined for private keys that control which ciphertexts the key owner is able to decrypt. Like [20], the security of the scheme is proved through a reduction to the hardness of the decision BDH assumption. The scheme is collusion resistant by design, as attributes are associated with ciphertexts, rather than with users.

Bethencourt *et al.* proposed the first ciphertext-policy attribute-based encryption (CP-ABE) scheme [21]. In contrast to KP-ABE, in CP-ABE the user's private key is associated with a set of attributes and an access policy is defined for each ciphertext. The primary advantage of this design over KP-ABE lies in the data owner's power to determine the access policy. CP-ABE is therefore conceptually closer to traditional access control methods. The scheme of [21] prevents collusion attacks by including in each attribute secret key a random user-specific blinding value. Their scheme allows for tree-based access policies but is only proven secure in the idealized generic group model.

A significant drawback of any ABE scheme discussed so far is its reliance on a single trusted server (authority) to monitor all attributes. In reality, it is more likely that multiple entities are responsible for maintaining this data. To solve this issue of disjoint attribute sets, Chase proposed the first multi-authority attribute-based encryption scheme [26]. Her design allows specifying for each authority a set of attributes. Ciphertexts contain for each authority a minimum number of attributes a decrypting user should possess. In addition to the limited expressiveness that this threshold scheme achieves, it is flawed in another way. It still requires a central authority, and this central authority has a master key

that allows it to decrypt any ciphertext.

Lewko and Waters designed a fully decentralized multi-authority ABE scheme. Their construction does not rely on any global coordination than the creation of an initial set of global parameters. Their system associates access structures on ciphertexts (CP-ABE) through the use of linear secret-sharing schemes (see definition 4.2), which is considered more efficient than the tree structure used by [21]. A hash function on the user's global identity is used to obtain collision resistance across attribute key generation of autonomous authorities. Their construction is based on groups of composite order and is proven secure in the random oracle model based on the subgroup decision problem (see definition 4.6). No implementation of the scheme is provided.

In addition to multi-authority ABE, many enhancements have been proposed to the standard CP-ABE. Rouselakis and Waters have enriched single and multi-authority schemes with large universe constructions, which compared to small universe constructions, do not have a public key size of authorities that grow linearly with the number of attributes [28, 29]. Attrapadung and Imai among others, proposed revocable ABE, in which a user's ability to decrypt a cipher can be revoked [30]. Policy-hiding schemes allow for the sharing of the cipher without leakage of the access policy or its attributes [31]. Furthermore, a significant amount of research is aimed at improving the performance of ABE schemes, with FAME being a well-known example [32].

ABE-enabled access control systems have seen successful implementations in the context of cloud computing and IoT applications [22, 23, 33]. ABE implementations in the context of blockchain have also been proposed. Rahulamathavan *et al.* proposed a blockchain-based IoT ecosystem with decentralized authorities [34], but the expressiveness of access policies in their scheme is limited. Wang and Song proposed a cloud-based electronic health record (EHR) system with blockchain to ensure the integrity and traceability of medical data but has a single point of failure, as a central authority is used.

At last, the theoretical and practical security of ABE needs to be addressed. The work of [36] shows, that there is no black-box construction of IBE (and therefore ABE) schemes. This entails an impossibility result for the creation of (very) efficient ABE schemes. In particular, some so-claimed "pairing-free" schemes [37], cannot be secure. Moreover, many schemes are proven secure in theory but employ certain shortcuts in their implementation to achieve reasonable performance. Venema and Alpár proposed a linear approach to analyze the security of many ABE implementations [38]. They found eleven schemes, most notably [33], for which they are able to decrypt any ciphertext. Their recommendation is to only rely on ABE schemes that do not have integer exponents in the keys, such as [17, 29] in multi-authority, and [32] in single-authority settings.

4.3. Definitions

In this section, we will formalize multi-authority attribute-based encryption and lay a foundation for the construction of our system and its security proofs in section 4.4 and section 4.5 respectively. We start by providing definitions for access structures and linear secret-sharing schemes that will be used to specify the expressiveness of our scheme. We then turn to pairing-based cryptography and define bilinear maps and the security assumptions that form the backbone of nearly all ABE schemes. At last, we formally define multi-authority attribute-based encryption and the security model.

4.3.1. Access Structures and Linear Secret Sharing Schemes

Definition 4.1 (Access Structure) *Let $\mathcal{U}$ be the universe of attributes. An access structure on $\mathcal{U}$ is a collection $\mathbb{A}$ of non-empty sets of attributes, i.e. $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, while the sets not in $\mathbb{A}$ are called the unauthorized sets.*

Additionally, an access structure is called monotone if $\forall B, C \in 2^{\mathcal{U}}$ if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$.

The access structure definition is adapted from [39], with attributes taking up the role of parties in our context. In this work we only consider monotonic access structures, as a result, users cannot lose decryption privileges by acquiring more attributes. As shown by Beimel *et al.*, any monotone access structure can be realized by a linear secret sharing scheme (LSSS) [39]. Following is the definition of an LSSS adapted from [39].

Definition 4.2 (Linear Secret Sharing Scheme (LSSS)) *Let p be prime and $\mathcal{U}$ the universe of attributes.*

A secret sharing scheme Π with domain of secrets over $\mathbb{Z}_p$ realizing access structures on $\mathcal{U}$ is linear over $\mathbb{Z}_p$ if

1. *The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.*
2. *For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a matrix A of size $m \times n$ with elements in $\mathbb{Z}_p$, called the share-generating matrix, and a function ρ, that assigns the rows of A attributes from $\mathcal{U}$, i.e. $\rho\colon \{1, \ldots, m\} \to \mathcal{U}$. Given a column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret share, and $r_2, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$. Then $A\vec{v}$ is the vector of m shares of the secret s according to Π. The share $\lambda_i = (A\vec{v})_i$ belongs to attribute $\rho(i)$.*

   We will be referring to the pair (A, ρ) as the policy of the access structure $\mathbb{A}$.

Each linear secret sharing scheme should satisfy the linear reconstruction property defined as follows: Suppose Π is an LSSS for access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be an authorized set, and let $I \subseteq \{1, \ldots, m\}$ be defined as $I = \{i \mid \rho(i) \in S\}$. There exists constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ satisfying $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$, such that for any valid shares $\{\lambda_i\}$ of a secret s according to Π, we have $\sum_{i \in I} \omega_i \lambda_i = s$. Furthermore, these constants $\{\omega_i\}$ can be found in time polynomial in the size of share-generating matrix A [39].

Finally, we note that there are standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix. A boolean formula over the universe of attributes can therefore be represented as an access tree, with AND and OR gates as interior nodes, and attributes as leaf nodes. Polynomial algorithms for this conversion from boolean formula to access structure are known [17, 39].

4.3.2. Pairing-based Cryptography

Since the pioneering work of Boneh and Franklin on identity-based encryption, pairing-based cryptography has formed the backbone of IBE and ABE schemes. The general idea is to use a pairing (mapping function) between elements of two groups to an element in a third group is used to reduce a hard problem in one group to a different, easier problem in another group. Bilinear maps form the basis of pairing-based cryptography.

Definition 4.3 (Bilinear Map) *Let G_1, G_2, and G_T be cyclic groups of the same order. A function $e\colon G_1 \times G_2 \to G_T$ is called bilinear if the following property holds:*

1. *(Bilinearity) for all $g \in G_1$, $h \in G_2$, and $a, b \in \mathbb{Z}$*

$$e(g^a, h^b) = e(g, h)^{ab}$$

Additionally, the bilinear maps we consider will have the following properties:

2. *(Non-degeneracy) There exists a generator $g \in G_1$ and $h \in G_2$ such that $e(g, h) \neq 1$*
3. *(Computability) The group operation in G_1, G_2, and G_T as well as the bilinear map e are efficiently computable.*
4. *(Symmetricity) When $G = G_1 = G_2$, then by (1) for all $g \in G$, and $a, b \in \mathbb{Z}$*

$$e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$$

When in addition to property (1) a bilinear map also has properties (2) and (3) then we call the map *admissible*. For our purpose, we are only interested in admissible bilinear maps. The fourth property is not necessarily desirable, but rather a consequence of the limited number of bilinear groups known with the other properties. Note that because G_1 and G_2 are cyclic groups, we can define the product rule that will be useful in the construction of the system in section 4.4.1.

5. (Product Rule) For all $g, g_1, g_2 \in G_1$, $h \in G_2$, and exponents $a, b, x_1, x_2 \in \mathbb{Z}$, such that $g_1 = g^{x_1}$

and $g_2 = g^{x_2}$, by (1) we obtain:

$$\begin{aligned} e(g_1^a \cdot g_2^b, h) &= e(g^{ax_1+bx_2}, h) \\ &= e(g,h)^{ax_1+bx_2} \\ &= e(g,h)^{ax_1} \cdot e(g,h)^{bx_2} \\ &= e(g^{ax_1}, h) \cdot e(g^{bx_2}, h) \\ &= e(g_1^a, h) \cdot e(g_2^b, h) \end{aligned}$$

The existence of a bilinear map with the properties defined in definition 4.3 has a direct implication on the hardness assumption of the decisional Diffie-Hellman (DDH) problem.

Definition 4.4 (Decisional Diffie-Hellman (DDH) Problem) *Let G be a cyclic group of order q and a generator g. The decisional Diffie-Hellman problem is, given (g, g^a, g^b, g^z) for $a, b, z \xleftarrow{\$} \mathbb{Z}_q$, to decide whether $z \equiv ab(\mod q)$. In other words, the advantage of an adversary $\mathcal{A}$ in distinguishing between tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^z), where $a, b, z \xleftarrow{\$} \mathbb{Z}_q$ is defined as*

$$Adv_{\mathcal{A}}^{\mathsf{DDH}} = \left| Pr\left[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - Pr\left[\mathcal{A}(g, g^a, g^b, g^z) = 1\right] \right|$$

In general, DDH is assumed to be hard for any polynomial-size adversary, but Joux and Nguyen pointed out that DDH in G is easy when a bilinear map $e\colon G \times G \to G_T$ is known [40]. To see this, observe that by the bilinear property of e, for $g, g^a, g^b, g^z \in G$, the problem reduces to checking whether $e(g^a, g^b) = e(g, g^z)$, $e(g^a, g^b) = e(g,g)^{ab} = e(g, g^{ab})$). Notice that the computational Diffie-Hellman (CDH) problem (determining z exactly) is still hard [40].

This observation was used by Boneh and Franklin to develop the first IBE scheme [19]. Since DDH is easy in a group G when a bilinear map $e\colon G \times G \to G_T$ is known, the DDH assumption cannot be used to build a cryptosystem in G. Instead, they used a variant of CDH called the Bilinear Diffie-Hellman (BDH) assumption for their security proof. The first ABE scheme proposed by Sahai and Waters relied on the decisional variant of this problem for their security proofs [20].

Definition 4.5 (Decisional Bilinear Diffie-Hellman (BDH) Problem) *Let G be a bilinear group of order q, generator $g \in G$, and map $e\colon G \times G \to G_T$. The decisional bilinear Diffie-Hellman problem is, given $(g, g^a, g^b, g^c, e(g,g)^z)$ for $a, b, c, z \xleftarrow{\$} \mathbb{Z}_q$ to decide whether $e(g,g)^z \equiv e(g,g)^{abc}(\mod q)$. In other words, the advantage of an adversary $\mathcal{A}$ in distinguishing between tuples $(g, g^a, g^b, g^c, e(g,g)^{abc})$ and $(g, g^a, g^b, g^c, e(g,g)^z)$, where $a, b, c, z \xleftarrow{\$} \mathbb{Z}_q$ is defined as*

$$Adv_{\mathcal{A}}^{\mathsf{BDH}} = \left| Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1\right] - Pr\left[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z) = 1\right] \right|$$

Sahai and Waters show that their ABE scheme can be reduced to BDH, which is assumed to be infeasible for any polynomial-size adversary [20]. Many ABE schemes would use the decisional BDH assumption as a basis for their security proofs [21, 26].

At last, we need to introduce the subgroup decision problem.

Definition 4.6 (General Subgroup Decision Problem) *Let G be a bilinear group of composite order $N = p_1 \cdot p_2 \cdot \cdots \cdot p_n$, where $p_1, p_2, \ldots, p_n$ are distinct primes, with bilinear map $e\colon G \times G \to G_T$. There is a subgroup of order $\Pi_{i \in S} p_i$ for each subset $S \subseteq \{1, \ldots, n\}$. The subgroup decision (SD) problem is, given two distinct subsets S_0, S_1 and random generator $g \xleftarrow{\$} G_{S_j}$ for $j \in \{0,1\}$, to decide the order of g. In other words, the advantage of an adversary $\mathcal{A}$ in distinguishing between tuples (N, G, G_T, e, g_0) and (N, G, G_T, e, g_1), where $g_0 \in G_{S_0}$ and $g_1 \in G_{S_1}$ are randomly chosen generators, is defined as*

$$Adv_{\mathcal{A}}^{\mathsf{SD}} = \left| Pr\left[\mathcal{A}(N, G, G_T, e, g_0) = 1\right] - Pr\left[\mathcal{A}(N, G, G_T, e, g_1) = 1\right] \right|$$

The subgroup decision problem is assumed to be infeasible for any polynomial-time adversary for sufficient big N.

The decentralized ABE scheme by [17] defines for its security proof 4 instances of the subgroup decision problem. Set for example $S_0 = \{1, \ldots, n\}$, in that case, g_0 becomes a random generator for G and the subgroup decision problem becomes to decide whether g_0 is also in a proper subgroup of G.

As our ABE implementation is based on the construction by [17], we will rely on their proof using the subgroup decision problem for our security analysis. Their scheme is based on a bilinear group of which the order is the product of 3 primes. In general, security proofs in composite order groups are easier to formulate than those in the prime order setting, especially against adaptive adversaries. Let G and G_T be groups of composite order $N = pq$. G has two subgroups, a subgroup G_p of order p with generator g^q, and a subgroup G_q of order q with generator g^p. For some α_1, α_2, consider members $h_1 = (g^q)^{\alpha_1} \in G_p$ and $h_2 = (g^p)^{\alpha_2} \in G_q$, we have:

$$e(h_1, h_2) = e(g^{q\alpha_1}, g^{p\alpha_2}) = e(g^{\alpha_1}, g^{\alpha_2})^{pq} = 1.$$

The bilinear property causes the generators of both subgroups to cancel each other out, $\{g_q, g_p\}$ can be thought of as an orthogonal basis for the 2-dimensional vector space. The bilinear functionality can be exercised in one of the dimensions, while the second dimension is used to introduce a *blinding factor*. Orthogonality ensures that after applying the bilinear map the blinding factors disappear.

In practice, prime order groups are preferred over composite order groups, as pairings on the latter are roughly 50 times slower to an equivalent secure pairing on the former [41]. The root cause of this significant difference is that the security proofs in composite order groups rely on the group order factorization remaining unknown to potential adversaries. Freeman shows a series of general transformations to construct prime order bilinear groups from groups of composite order, preserving their properties [41]. In section 4.4 the pairings and groups of our implementation will be discussed in detail.

4.3.3. Multi-Authority Attribute-based Encryption

In the multi-authority setting, each attribute $a \in \mathcal{U}$ is controlled by a specific authority $\theta \in \mathcal{U}_\Theta$, where $\mathcal{U}$ and $\mathcal{U}_\Theta$ are the universes of attributes and authorities respectively. While in practice, multiple authorities might internally use the same attributes for access control, e.g. “Admin”. In the implementation of the scheme this is solved by enforcing a unique identifier $\mathrm{id}(\theta)$ for each authority $\theta \in \mathcal{U}_\Theta$. We can then think of attributes as a string attribute concatenated by $\mathrm{id}(\theta)$, e.g. “Admin@corp1”. Now follows the definition of multi-authority attribute-based encryption.

Definition 4.7 (Multi-Authority Attribute-based Encryption (MA-ABE)) *A multi-authority attribute-based encryption scheme is a tuple of five polynomial-time algorithms* $\text{MA-ABE} = (\text{Setup}, \text{AuthSetup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ *such that:*

$GP \leftarrow \text{Setup}(1^k)$: *is a probabilistic algorithm that takes as input a security parameter $k \in \mathbb{N}$. It outputs the global parameters GP for the system.*

$SK, PK \leftarrow \text{AuthSetup}(GP, \{a\})$: *is a probabilistic algorithm run by each authority with global parameters GP and a set of attributes $\{a\}$ as input. It outputs the authority's secret key SK and public key PK.*

$c \leftarrow \text{Encrypt}(GP, \{PK\}, \mathbb{A}, m)$: *is a (possibly probabilistic) algorithm that takes as input the global parameters GP, an access structure $\mathbb{A}$, a set of public keys $\{PK\}$ of authorities corresponding to the attributes in $\mathbb{A}$, and a message m to be encrypted. It outputs a cipher c.*

$K_{a,GID} \leftarrow \text{KeyGen}(GP, SK, a, GID)$: *is a (possibly probabilistic) algorithm that takes as input the global parameters GP, an attribute authority's secret key SK, an attribute a, and a general identifier GID. It outputs a key $K_{a,GID}$ for the given attribute and identifier.*

$m := \text{Decrypt}(GP, \{K_{a,GID}\}, c)$: *is a deterministic algorithm that takes as input the global parameters GP, a set of keys corresponding to attributes with fixed identifier GID, and cipher c. It outputs either the message m when the set of keys satisfies the access structure defined during encryption of c, or decryption fails.*

An MA-ABE *scheme is correct if for all* $k \in \mathbb{N}$*, attributes* $a \in \mathcal{U}$*, access structures* $\mathbb{A} \in$ *and* $GID, m \in \{0,1\}^*$*, we have for all* GP *generated by* Setup(1^k)*, for all key pairs* SK, PK *generated by* AuthSetup(GP)*, and for all keys* $K_{a,GID}$ *generated by* KeyGen(GP, SK, a, GID)*, then for a set of keys* $\{K_{a,GID}\}_{a \in A}$ *where* $A \in \mathbb{A}$ *and* GID *equal for all elements,*

$$\text{Decrypt}(GP, \{K_{a,GID}\}, \text{Encrypt}(GP, \{PK\}, \mathbb{A}, m)) = m.$$

4.4. Multi-Authority ABE and Blockchain

We developed a system of authorities, clients, and blockchain peers that provides a fine-grained solution to access control. We use an efficient implementation of the multi-authority attribute-based encryption scheme by [17] that is proven secure against adaptive adversaries. Furthermore, the decentralized nature of the blockchain is maintained as this scheme does not require a central authority. The Hyperledger Fabric (HLF) platform was used to create a modular blockchain network that users can extend to fit their specific use cases [3].

The MA-ABE construction proposed by [17] is highly regarded, but their work does not contain an actual implementation of the proposed scheme. In this work, we identified and resolved the two main challenges that would hinder such an implementation. The first obstacle is related to the reliance on the global identity, GID to achieve collusion resistance. A hash function is used to map GID to a group element and is blinded by an exponent. Collusion between users with different global identifiers GID, GID' is prevented as they are unable to "unblind" this exponent. However, the uniqueness of a GID should be verified to prevent an exploit where multiple users issue private keys from different authorities using the same GID. As described in section 2.1, identity on the blockchain is managed through a combination of certificate authorities (CA) and membership service providers (MSP) in HLF. CAs do not have to be company-specific and their certificates can be accepted by multiple MSPs in the consortium. We will use the unique identity contained within those certificates for the global identifier in our implementation. This solution might not be satisfactory, as it introduces centralization into the network, therefore, in section 4.7 we will discuss some potential solutions.

The second problem with an implementation of the MA-ABE system proposed by [17], is the construction of the scheme in composite order groups. Cryptosystems using composite order groups are known to perform poorly. The authors provide a construction in prime order groups but their dual system encryption proof technique does not work in this setting. The construction in prime order groups is therefore only proven secure against static adversaries. Different techniques for converting proofs in composite order groups to prime order have been proposed [41, 42]. However, we are unaware of any successful attempt in the case of [17]. Moreover, to the best of our knowledge, no decentralized ABE scheme with construction with proven adaptive security in prime order groups is known to exist [43]. Note that the absence of such a proof does not imply a construction in prime order groups is not secure against adaptive adversaries. Our work will be based on the prime order construction of [17], in section 4.5 we discuss the implications of this choice.

In section 4.4.1, we discuss the construction of [17] in prime order groups. In section 4.4.2, we describe the implementation of authorities, client applications, and the interactions between them. Section 4.4.3 gives insight into the generic wrapper we developed to interface with our MA-ABE implementation from diverse contexts. At last, in section 4.4.4, a use case will be discussed that highlights the entities and interactions in the system.

4.4.1. Construction

Our multi-authority attribute-based encryption system is based on the MA-ABE construction in prime order groups presented in [17] appendix D. The main difference between the construction presented here, and that of Lewko and Waters, is that instead of a symmetric group G, we use an asymmetric pairing $e\colon G_1 \times G_2 \to G_T$ of prime order p. We used the generic conversion discussed in [29] to convert the symmetric construction into an asymmetric one.

Collusion is prevented by embedding the user's identity into the attribute keys generated by authorities. To be more precise, a function H mapping identities to elements in G_1 is combined with the authority's secret key to generate user-bounded keys. During encryption the message m is blinded by $e(g_1, g_2)^s$,

where g_1 G_1, $g_2 \in G_2$, and random $s \in \mathbb{Z}_p$. According to the access matrix, the value s is split into shares λ_x, while the value 0 is split into shares w_x. During decryption, the attribute keys of a user with identity *GID* will introduce terms of the form $e(g_1, H(\textit{GID}))^{w_x}$ into the equation. If the user has a satisfying set of attribute keys, these terms will cancel each other out, and the user will be left with $e(g_1, g_2)^s$. Colluding with a user with identity GID' would introduce the term $e(g_1, H(\textit{GID'}))^{w_{x'}}$ that would not cancel out en thereby prevent recovery of $e(g_1, g_2)^s$.

The construction consists of the following five algorithms:

GP $\leftarrow$ **Setup**(1^k)**:** Choose bilinear groups G_1 and G_2 of order $p = \Theta(k)$ and select generators $g_1 \in G_1$ and $g_2 \in G_2$. Also, pick a function H mapping global identifiers *GID* to elements of G_1. Output:

$$\textit{GP} = (p, G_1, G_2, g_1, g_2, H).$$

SK, ***PK*** $\leftarrow$ **AuthSetup**$(GP, \{a\})$**:** Choose two random exponents $\alpha_a, y_a \in \mathbb{Z}_p$ for every attribute a belonging to the authority. Output:

$$\textit{SK} = \{\alpha_a, y_a \forall a\}, \quad \textit{PK} = \{e(g_1, g_2)^{\alpha_i}, g_2^{y_a} \forall a\}.$$

$c \leftarrow$ **Encrypt**$(\textit{GP}, \{\textit{PK}\}, \mathbb{A}, m)$**:** Let the access structure $\mathbb{A} = (A, \rho)$ define an $n \times l$ share-generating matrix A with function ρ mapping the rows of $\mathbb{A}$ to attributes. Choose random $s, w \in \mathbb{Z}$ and random vectors $\vec{v}, \vec{w} \in \mathbb{Z}_p^l$ with s and 1 as their first entry, respectively. We define $\lambda_x = A_x \cdot v$ and $w_x = A_x \cdot \vec{w}$, where A_x is row x of A. At last, for each row A_x of A, choose a random $r_x \in \mathbb{Z}_p$. Output ciphertext $c = (C_0, \{C_{1,x}, C_{2,x}, C_{3,x} \forall x\})$, where:

$$C_0 = m \cdot e(g_1, g_2)^s, \quad C_{1,x} = e(g_1, g_2)^{\lambda_x} \cdot e(g_1, g_2)^{\alpha_{\rho(x)} r_x}, \quad C_{2,x} = g_2^{r_x}, \quad C_{3,x} = g_2^{y_{\rho(x)} r_x} g_2^{w_x}.$$

$K_{a,\textit{GID}} \leftarrow$ **KeyGen**$(\textit{GP}, \textit{SK}, a, \textit{GID})$**:** Output:

$$K_{a,\textit{GID}} = g_1^{\alpha_a} H(\textit{GID})^{y_a}.$$

$m :=$ **Decrypt**$(\textit{GP}, \{K_{a,\textit{GID}}\}, c)$**:** Let (A, ρ) be the access policy of the ciphertext c. For each x in the subset of secret keys $K_{\rho(x),\textit{GID}}$ such that $(1, 0, \ldots, 0)$ is in the span of the rows A_x of A, compute $C_{4,x}$:

$$\begin{aligned}
C_{4,x} &= \frac{C_{1,x} \cdot e(H(\textit{GID}), C_{3,x})}{e(K_{\rho(x),\textit{GID}}, C_{2,x})} \\
&= \frac{e(g_1, g_2)^{\lambda_x} e(g_1, g_2)^{\alpha_{\rho(x)} r_x} \cdot e(H(\textit{GID}), g_2^{y_{\rho(x)} r_x} g_2^{w_x})}{e(g_1^{\alpha_{\rho(x)}} H(\textit{GID})^{y_{\rho(x)}}, g_2^{r_x})} && \text{Substition} \\
&= \frac{e(g_1, g_2)^{\lambda_x} \cdot e(g_1, g_2)^{\alpha_{\rho(x)} r_x} \cdot e(H(\textit{GID}), g_2^{y_{\rho(x)} r_x + w_x})}{e(g_1^{\alpha_{\rho(x)}}, g_2^{r_x}) \cdot e(H(\textit{GID})^{y_{\rho(x)}}, g_2^{r_x})} && \text{Product rule, 4.3 (5)} \\
&= \frac{e(g_1, g_2)^{\lambda_x} \cdot e(g_1, g_2)^{\alpha_{\rho(x)} r_x} \cdot e(H(\textit{GID}), g_2)^{y_{\rho(x)} r_x + w_x}}{e(g_1, g_2)^{\alpha_{\rho(x)} r_x} \cdot e(H(\textit{GID}), g_2)^{y_{\rho(x)} r_x}} && \text{Bilinearity, 4.3 (1)} \\
&= e(g_1, g_2)^{\lambda_x} \cdot e(H(\textit{GID}), g_2)^{w_x} && \text{Divide}
\end{aligned}$$

Then choose constants $c_x \in \mathbb{Z}_p$ such that $\sum_x c_x A_x = (1, 0, \ldots, 0)$ and compute:

$$\prod_x C_{4,x}^{c_x} = \prod_x \left(e(g_1, g_2)^{\lambda_x} \cdot e(H(\textit{GID}), g_2)^{w_x}\right)^{c_x} = e(g_1, g_2)^s$$

Recall that $\lambda_x = A_x \cdot \vec{v}$ and $w_x = A_x \cdot \vec{w}$, where $v \cdot (1, 0, \cdots, 0) = s$ and $\vec{w} \cdot (1, 0, \ldots, 0) = 0$. Output:

$$m = C_0 / e(g_1, g_2)^s.$$

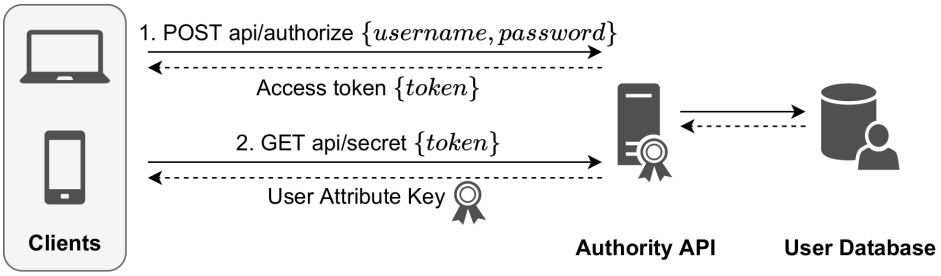


Figure 4.1: Interactions between client, authority, and database for acquiring set of attribute keys.

4.4.2. Authorities and Clients

To verify our imagined MA-ABE scheme works as intended in the blockchain context, we developed authorities, as well as client applications. We did not implement the cryptographic algorithms described in section 4.4.1 from scratch, instead, we used the Rabe[1] library that contains several ABE implementation, including the scheme of [17]. The library makes use of an efficient bilinear pairing on the Barreto-Naehrig curve [44]. Unlike the group G used in the proof of [17], this curve generates two asymmetric groups G_1, G_2. The work of [28] shows assumption and security proofs in the symmetric setting can be translated to the asymmetric setting in a generic way.

The authorities are created in the Rust programming language[2]. This seemed like an obvious choice, as the implementation of MA-ABE is in Rust as well. An authority is implemented as a simple service with an accompanying SQL database. Interaction with clients happens through a RESTful application programming interface (API). The authority is able to communicate with databases of different vendors (PostgreSQL, MySQL, SQLite), depending on the scale of the system. A seed can be provided during authority setup to initialize the set of attributes, users, and assignments of the former to the latter.

A client communicates with an authority by sending HTTP requests to the API of the authority. A client library was created to make this process easy. The authority offers several endpoints for clients to request public or private keys from the authority. For the private attribute keys, the client needs to provide an access token that can be obtained through the authorization endpoint. Our implementation of the client library, attribute authority, and foreign function interface, including working examples, can be found in our code repository[3].

Figure 4.1 showcases an interaction between clients, an authority API, and its database, the public endpoints of the API are deliberately left out in order to keep the example small. Consider an employee of an organization hosting the authority displayed in figure 4.1. Whenever this employee wants to decrypt an ABE encrypted ciphertext obtained from the blockchain, they need to get a set of attribute keys that match the access policy defined for the ciphertext. The attribute keys can be obtained from a secured endpoint, for which the client has to first authorize itself to gain access. This two-step process was implemented to mimic real-world scenarios more closely, and simplify adoption in content management systems (CMS) with an authentication layer already in place. A description of the interaction displayed in figure 4.1 between client and authority follows:

Authorization Through a client application, the user sends a POST request containing their credentials to the 'api/authorize' endpoint of the authority API (1). The server checks that the supplied username and password combination matches an entry in the database. If this check is successful, the server generates an access token. This token is valid for a limited amount of time and specifies the claims the user has. In our MA-ABE context, the only important claim is the general identifier (GID) of the user. In the end, the access token is signed by the attribute authority and supplied to the user.

Key Generation After authorization, the follow-up step is to request the private keys associated with the attributes the user is entitled to. The client sends a GET request to the secured 'api/secret' endpoint with the previously obtained access token in the header. The server checks whether

[1] Rabe, library of ABE implementations in Rust: `https://github.com/Fraunhofer-AISEC/rabe`
[2] The official Rust programming language website: `https://www.rust-lang.org/`
[3] MA-ABE repository: `https://github.com/bartvsdev/thesis-abe`

the received access token is (still) valid. When validity is confirmed, the identity of the user is extracted from the access token. The database is queried for all attributes this identity is qualified for. The attribute performs the KeyGen algorithm discussed in section 4.4.1 using its master key, the obtained set of attributes, and the user's identity. The resulting set of private keys is returned to the client sending the request.

4.4.3. Foreign Function Interface

In the previous section, we mentioned how Rust was the obvious programming language of choice for authorities. The reasoning behind this decision was that the ABE functionalities were implemented in this language. The same logic holds true for the implementation of the client. However, the client needs to interact with HLF far more often than authorities do. In the case of the latter, the only interaction is sharing of the public key after setup, which can be performed by a third party. HLF's official APIs and software development kits (SDKs) are written in the Go programming language[4] but Javascript, Java, and Python are also supported. Rust is not, therefore, Go was used to implement our client library.

However, the use of Go for the implementation of the client library presents a new challenge. Data owners and data users are unable to perform the encryption and decryption operations locally, respectively. Providing this functionality in a second Rust library to clients would be inconvenient while implementing the algorithms in Go would be time-consuming and error-prone. We opted for a third option, namely, creating a foreign function interface (FFI) for the ABE Rust implementation.

An FFI is a layer of software by which functions in one programming language can be invoked by programs in a different language. The main goal of this wrapper is to integrate the semantics and calling conventions of the host language with those of the guest language. Other language differences to take into consideration when developing such an interface are: runtime environments, mapping of complicated data types, and memory management i.e. manual versus garbage collection. In general, solutions involve wrapping the guest-language functions with glue code, which performs a translation from a mutually understood input data type (e.g. string) to the function's parameter types. Serialization of variables is common practice, but passing references or pointers can be an efficient alternative in certain niche cases.

In our case, we are not too concerned with the serialization overhead between our host language, Go, and guest language, Rust. These languages are both known for achieving high performance, and we expect the notorious expensive ABE operations to diminish this burden. Characteristics of an FFI implementation we do keep in high regard are consistent mapping between languages, good developer experience, and a generic approach. The first two are especially important in the context of cryptosystems, for instance, different string representations between languages can be a source of bugs. The last point is important in case when we want to provide ABE client support in languages other than Rust and Go in the future.

Grishkov *et al.* provide an appealing take on an FFI implementation [45]. They propose to compile the guest-language code into WebAssembly[5], a portable binary format, primarily intended for web applications. Execution of a WebAssembly binary by the host language requires a WebAssembly runtime. This adds some overhead, but more importantly, string support is limited as they are not part of the WebAssembly specification. Functions in the guest language require non-intuitive glue code to transform input parameters and output values to byte arrays.

Inspired by the work of [45] we came up with our own solution, based on protocol buffers (protobuf). Protobuf is a language-neutral mechanism for efficient serialization of structured data, mainly used in gRPC[6]. Protobufs are a more efficient alternative compared to other structured serialization techniques such as JSON or XML. A `.proto` file describes the structure of data types, which are neatly compiled into programming language-specific structures or classes. In our solution, a single protobuf file defines the structure of all data types that cross the language boundary. During the project's build stage, the language-specific structures for Rust and Go are generated. Nowadays, all modern programming languages have protocol buffer compilers to accomplish this step.

[4] The official Go programming language website: `https://go.dev/`
[5] The official WebAssembly website `https://webassembly.org/`
[6] gRPC: Remote Procedure Call framework initially created by Google, `https://grpc.io/`

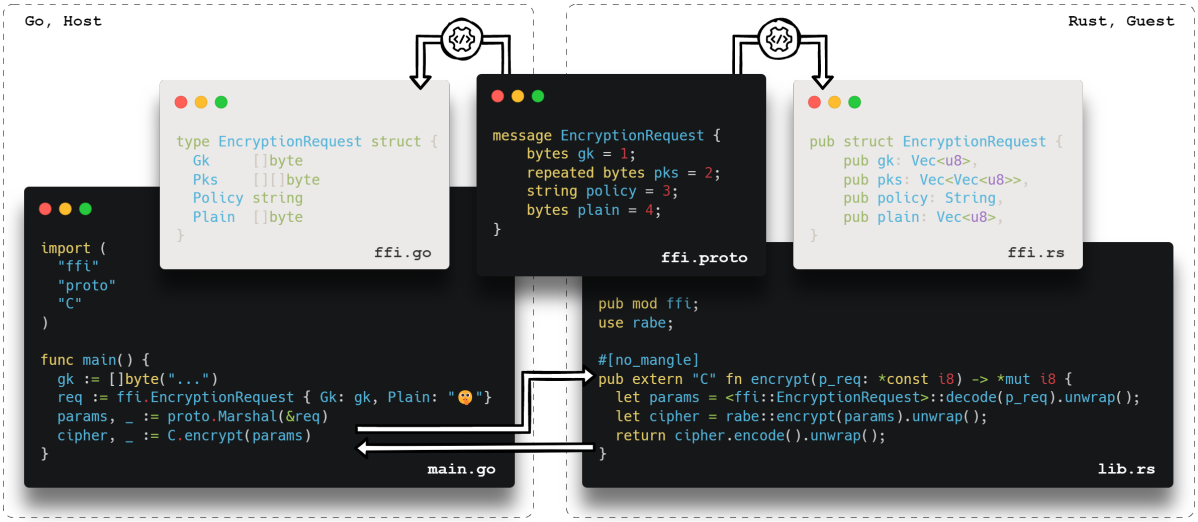


Figure 4.2: Example of an Encrypt foreign function invocation, enabled by our language-neutral approach.

An example foreign function invocation using our generic approach is shown in figure 4.2. A definition of the FFI is provided through the `ffi.proto` file. In this file, we describe the structure of the expected function parameters and return values. Protobuf compilers are used to generate two files, the `ffi.go` for our host language and the `ffi.rs` for our guest language, which contain language-specific data for each. Apart from the data type description shown in figure 4.2, functionality for object (de)serialization is generated as well. In the host language, a new record is created specifying all the parameters to perform the encryption request. The foreign `encrypt` function is invoked using the serialized record parameters. In the guest language, a little glue code is required to deserialize the received request and extract the input parameters. The `rabe.encrypt` function performs the actual encryption of the plaintext, after which the resulting ciphertext is returned to the host language. Note that this last step is simplified for the purpose of keeping the example concise. In reality, the resulting ciphertext would need to be serialized before being returned.

In section 4.6, we analyze the overhead of our FFI implementation. Furthermore, in section 4.7, we discuss several strategies to reduce the amount of glue code users have to write.

4.4.4. An Illustrative Example

Figure 4.3 contains an overview of the different parties in our system and an example of their interactions. We will refer to the toy example we considered in chapter 3. This time, consider the educational institution, a university, that wishes to collaborate with the local library and potentially other organizations. In particular, students of the university who are members of the library as well should be able to access certain information. Additionally, this information should be accessible to teachers as well. As it's cumbersome to identify all individuals that comply with these conditions, these properties are modeled as attributes in our ABE system. The order of operations to securely share the information with the specified subset of users is as follows:

Setup For the system to become operational, a trusted setup needs to be performed. During this step, the Setup algorithm is used to generate the global parameters GP of the system. The identity of the admin performing this operation is not important, just that the chosen random parameters are truly random. GP is then shared with other users by saving it on the ledger (1). The entity that performed the trusted setup is free to leave the system at this point.

Authority Setup Every entity in the system that wishes to become an authority needs to perform an individual setup as well. To start, the global parameters GP of the system are obtained (2). Then the AuthoritySetup algorithm is run, with inputs GP, and set of attributes A provided by the authority. For instance, our example university has attribute set $A = A_u = \{Student, Teacher, MSc, \dots\}$. The algorithm outputs a public/private key pair. The university's private key SK_u is locally stored, while the public key PK_u is shared with other users through the blockchain (3).

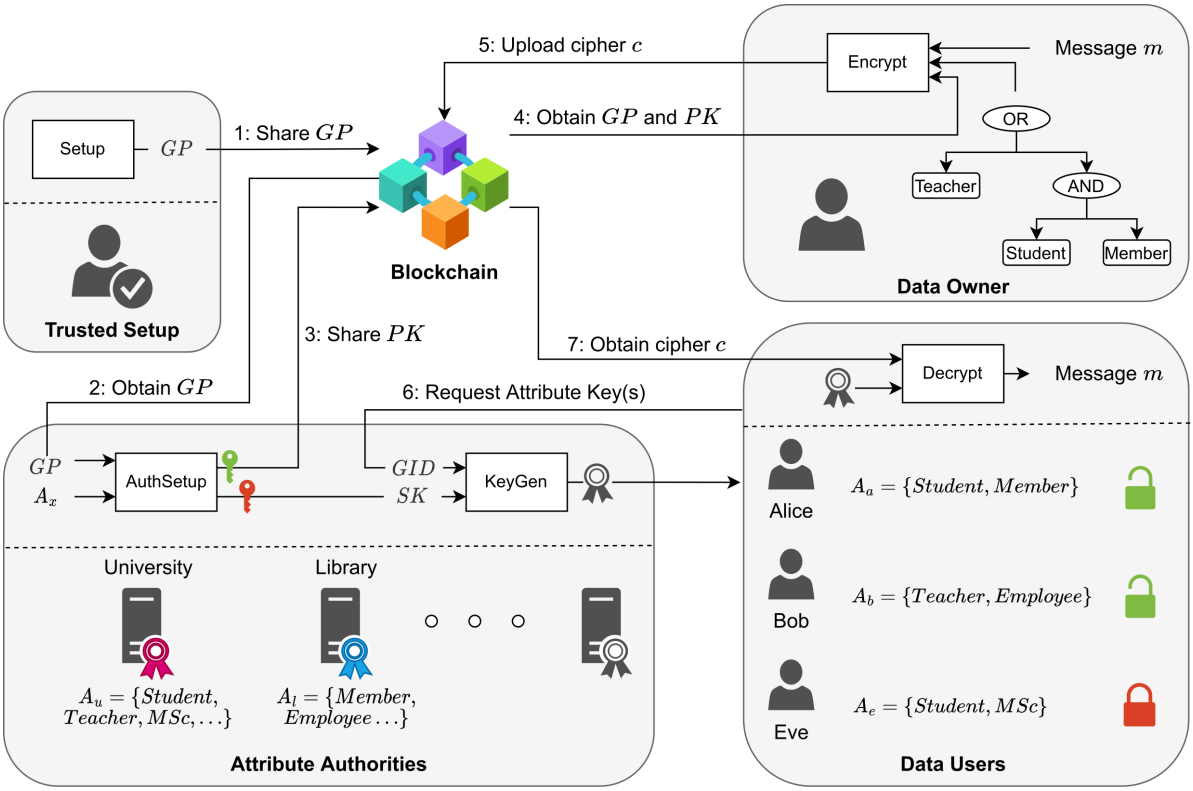


Figure 4.3: Overview of the ABE interactions in our system.

Encryption Now consider a data owner who decides to share a piece of information contained by message m. At first, the data owner determines an access structure for the data to be encrypted. As mentioned previously, this policy should only grant teachers or students who are also members of the library the ability to decrypt. In the example in figure 4.3, the access structure is depicted as an access tree with corresponding boolean formula $\mathbb{A} = Teacher \vee Student \wedge Member$. The data owner then collects the general parameters GP and the public keys of all authorities with attributes in $\mathbb{A}$, $PK_{\mathbb{A}=\{PK_u, PK_l\}}$, from the ledger (4). At last, the cipher c is computed by performing the Encrypt algorithm on inputs $GP, GK_{\mathbb{A}}, \mathbb{A}$, and$m$, and shared on the ledger (5).

Key Generation To decrypt a cipher, data users first need to obtain the private key(s) corresponding to their attributes. For Alice in the example of figure 4.3, this means she will need to send a request to both the authorities of the university and the library (6). The key generation algorithm requires 4 inputs: general parameters GP, authority's secret key SK, set of attributes A, and general identifier GID. During its setup, the authority already obtained GP and generated SK. A and GID can be supplied by the user or are known by the authority. The attribute set $A = \{a \mid a \in A_{authority}, a \in A_{user}\}$ contains a subset of all attributes belonging to the authority for which the user is authorized. The general identifier GID is a string that uniquely identifies the user across all institutions, e.g. an email address, name, or SSN. The authority performs the KeyGen algorithm for all attributes $a \in A$ and returns for every attribute a a key $\{K_{a,GID}\}$ under her identity GID. Alice repeats this process for all other authorities and obtains private key set $\{K_{a,GID}\}$ for all attributes $a \in A_a$.

Decryption Finally, Alice would like to read the message m encrypted for her (and others). To start, she obtains the cipher c and general parameters GP from the blockchain (7). Furthermore, she collects a set of private keys that satisfies the access structure defined for c, i.e. $\{K_{Student,GID}, K_{Member,GID}\}$, as described in the previous step. She then locally performs the Decrypt algorithm and obtains message m. In the same manner, Bob can decrypt c, but he will only have to authorize himself to the university's attribute authority. Eve cannot decrypt c as her set of attributes does not satisfy the access structure.

4.5. Security

Given the implementation of our system relies on the construction in prime order groups of [17], we will not rehearse their full proof here. Instead, we would like to redirect any interested reader to appendix E of said work. In the remainder of this section we will discuss the implications of the achieved level of security.

The authors of [17] proof their prime order group construction in the generic group model [46]. The generic group model is an idealised cryptographic model, where the adversary is only given black-box access to the group operations. Oracles are used to model group operations, pairings and the identity mapping function H. Security proofs in the generic group model provide a weaker notion of security compared to proofs in the standard or random oracle model. Dent showed a number of cryptographic schemes that are provably hard in the generic group model, but easy to break in practice [47]. Particularly, in the real-world, problems arise when the random encoding functions of the oracles is replaced by a specific encoding function.

When applied to cryptographic schemes that rely on the hardness of the discrete logarithm (DLP), like ABE, the generic group model is more limited than the random oracle model [48]. In the random oracle model, the idealization of the real-world is generally limited to hash functions. This assumption is realistic in the sense that for a good hash function H, the output of H will be almost indistinguishable from a random function. On the other hand, the generic group model assumes the group in question has no special structure or property aiding in solving DLP. This assumption might not hold when considering the groups actually used in real-world cryptography.

4.6. Performance Analysis

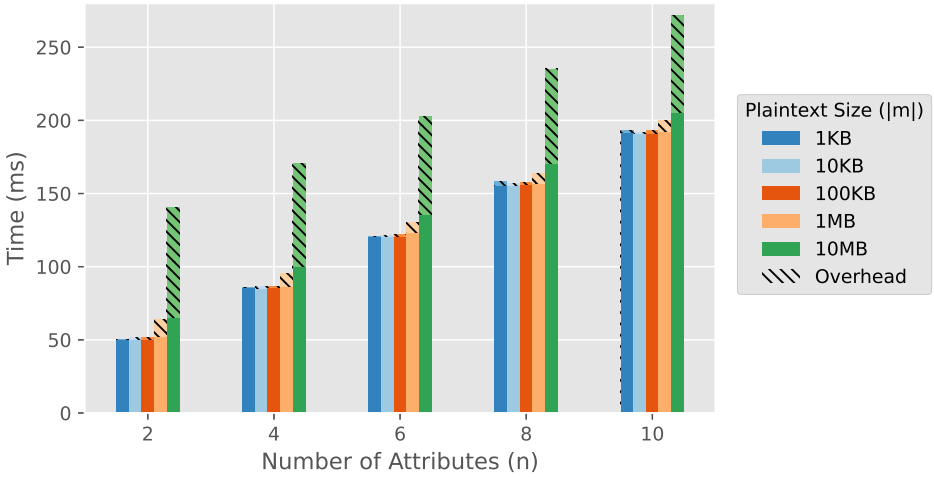
As discussed in the previous sections, our system relies on the Barreto-Naehrig curve for pairings. The curve is defined over a finite field $\mathbb{F}_q$ with prime $q \approx 2^{256}$, providing a 128-bit security level [44]. To determine the overhead of our FFI implementation, we compared its performance with the native implementation. Specifically, we measured the execution time and memory consumption of the ABE encrypt and decrypt operations. All measurements presented below are the averages taken over 100 iterations. All experiments are run on an AMD Ryzen 5 4600H @ 3.00GHz 6 CPU computer with 16 GB RAM. The benchmark implementations can be viewed in our code repository as well[7].

The most expensive cryptographic operation in any ABE scheme is the bilinear pairing, followed by exponentiation and point multiplication. Judging from the construction in section 4.4.1, the number of iterations of the encryption and decryption algorithms depends on the number of distinct attributes in the access structure. Therefore, we deploy access policies of the form $a_1 \wedge a_2 \wedge \cdots \wedge a_n$, as all n attributes are then required for decryption. We measure the performance for attribute sets of size $n \in \{2, 4, 6, 8, 10\}$. We expect only a limited number of use cases would require access policies with conjunctions of more than 10 attributes.

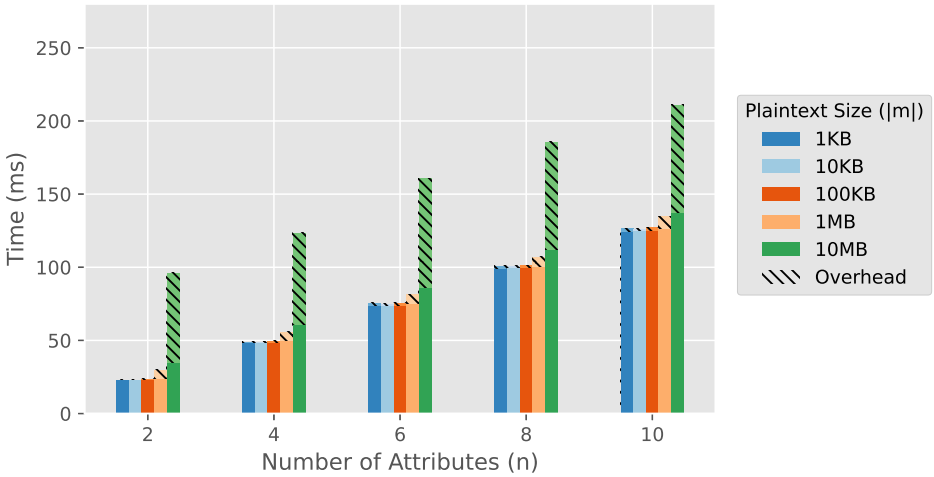
Compared with the cost of a bilinear pairing operation, the cost of multiplicating this pairing with a plaintext is negligible. However, the size of the plaintext cannot be ignored considering the memory consumption of the system, especially towards the (de)serialization overhead introduced by our FFI approach. For our experiments, the plaintext m is randomly generated with size $|m| \in \{1$ KB, 10 KB, 100 KB, 1 MB, 10 MB$\}$. As we consider the ABE implementation in the context of blockchain, and blockchains are inefficient for storing large files, we will not consider plaintext sizes larger than 10 MB.

Figure 4.4 shows the time required for encryption (a) and decryption (b) for a range of plaintext sizes $|m|$ and the number of attributes n. As expected, the experiments show a linear relationship between the running time of both operations and the access policy's complexity n. The influence of $|m|$ on the execution time is minimal in both cases. Only for $|m| \geq 1$ MB we notice a significant increase due to the (de)serialization overhead introduced by FFI. At last, we would like to shift the attention of the reader to the relatively high constant cost of the scheme. Encryption and decryption using a 10-attribute policy take approximately 190 ms and 125 ms on our machine. In comparison, at the same 128-bit security level, the equivalent 3072-bit RSA operations take 0.42 ms and 4.9 ms.

[7] MA-ABE repository: `https://github.com/bartvsdev/thesis-abe`



(a) Encryption



(b) Decryption

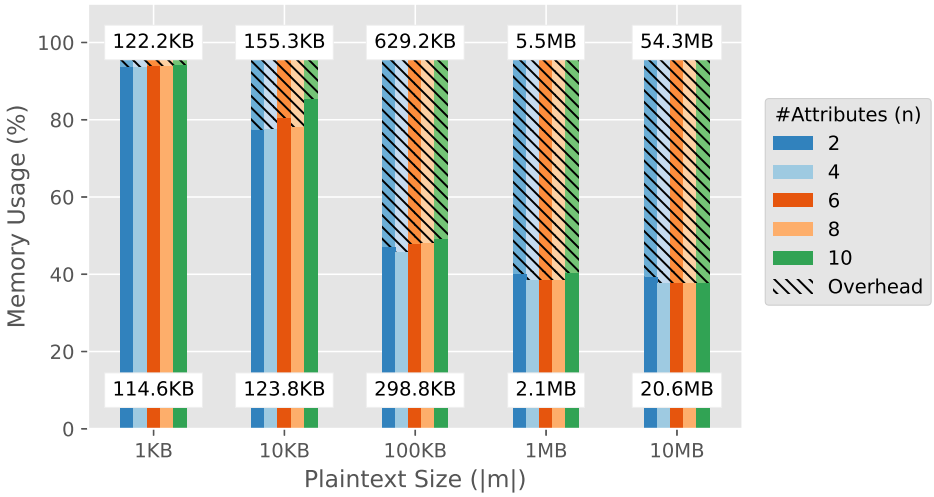
Figure 4.4: Execution time of ABE operations through native and FFI invocation. The number of attributes n corresponds to the complexity of the access policy $a_1 \wedge a_2 \wedge \cdots \wedge a_n$. The timings represent the average of 100 iterations.

In figure 4.5, we measured the overhead in memory consumption of our FFI implementation for the encrypt (a) and decrypt (b) operations. The experiment was conducted by measuring the maximum amount of heap-allocated during a single invocation of the operation. As expected, the memory usage grows linearly with the size of the plaintext. To display the results in a clear and space-efficient manner, the y-axis in figure 4.5 is normalized. The bottom and top labels on the bars represent the average memory usage (across all attribute sizes) for the operation invoked natively and through the FFI, respectively. The overhead of the FFI wrapper is obtained by subtracting the former from the latter.

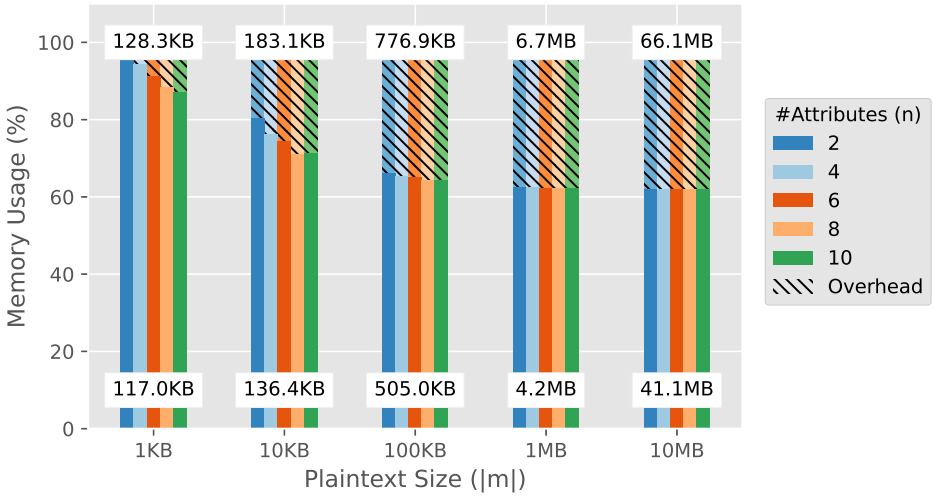
In percentage terms, the memory overhead induced by the FFI wrapper seems to approach a limit for bigger plaintexts of 166% and 61% over the native approach for the encrypt and decrypt operation, respectively. In absolute terms, in the case of $|m| = 10$ MB, decryption uses an additional 34 MB $-$ 25 MB $= 9$ MB. This absolute difference is caused by an extra serialization step used in the glue code of the encrypt operation of the Rust FFI wrapper.

4.7. Discussion

Our proposed multi-authority attribute-based encryption system shows how fine-grained sharing of encrypted data on a blockchain can be achieved. The decentralized nature of the network is preserved as our approach does not require a central authority, unlike similar proposals [35]. To securely use ac-



(a) Encryption



(b) Decryption

Figure 4.5: Normalized memory usage of ABE operations through native and FFI invocation, the dashed parts indicate the overhead induced by the FFI approach. The absolute memory usage displayed by the labels is: bottom: native, top: FFI. The overhead of the FFI approach is computed as FFI - native. Memory usage is measured as the average maximum amount of heap allocations over 100 iterations.

cess policies containing attributes of multiple authorities we use the identities provided by Hyperledger Fabric. We discussed how this required trust in a single certificate authority from these attribute authorities and why this could be undesirable. Decentralized identity systems and decentralized identifiers (DID) are popular research topics, without a clear generally applicable solution [49]. An interesting HLF-specific solution is proposed by [50] that involves the linking of several permissioned networks to create a secure distributed identity management infrastructure.

In our experiments, we showed that the computational overhead introduced by our FFI implementation was minimal for small up to medium-sized plaintexts. For large plaintexts the additional execution time and memory consumption were substantial. However, we deem this overhead a fair trade-off considering the improved developer experience our approach utilizing protocol buffers offers. Moreover, like other forms of public-key encryption, we expect our ABE implementation to be mostly used to transmit keys used in symmetric-key cryptography. It would be interesting to see how this approach performs when other programming languages and use cases are considered.

5

Trusted Execution Environment

A key feature of blockchain is the transparency of transactions, but sometimes a client requires confidentiality, even from the network node handling its request. For example, in e-voting or sealed-bid-auctions, it is essential that votes and bids, respectively, are kept private. Trusted execution environments (TEEs) have been proposed to facilitate confidential transactions in blockchain. A TEE is a tamper-resistant processing environment that guarantees confidentiality and integrity to programs running inside it. Not even the peer hosting the TEE is able to observe its contents. In this chapter, we analyze two research projects: Fabric Private Chaincode[51] and Ekiden[52]. They aim to achieve the same goal: confidential transactions through TEEs on blockchain networks. Both projects take opposing approaches to solve the same challenge. After having analyzed both works, we must conclude that the benefits of integrating TEE in blockchain do generally not outweigh the disadvantages of the additional overhead and centralization of the network.

5.1. Introduction

As modern systems continue to increase in complexity, the task of analyzing and securing them becomes more difficult. This has led to the rise of the trusted computing trend that aims to create a foundation of trust for software processes in hardware [53]. The idea is to establish a trust relationship under the assumption that the system is potentially running compromised software. The original implementation of this idea was the Trusted Platform Module (TPM). TPMs provide a range of security and trust-related functionalities such as a random number generator, remote attestation, and cryptographic key storage among others. The primary scope of TPMs is to provide evidence of platform integrity by creating a hardware root of trust.

A major drawback of TPMs is that they do not provide an isolated execution environment for third-party applications, severely limiting their use cases. To address this limitation, Trusted Execution Environments (TEEs) have emerged. TEEs are fully isolated and secure processing environments that protect the integrity of programs. It enables part of the code of the program to operate in private hardware-encrypted areas of the memory, so-called enclaves. This prevents other software applications, the OS, or the host owner from tampering with or observing the state of a program running inside the enclave. Major CPU vendors have their own TEE implementations, such as Intel Software Guard eXtensions (SGX) [54] and ARM TrustZone [55].

The use of trusted execution environments within a blockchain context sounds promising due to the complementary properties of both technologies. Blockchain can guarantee strong availability and has persistent storage capabilities, whereas a TEE's availability and reliable storage access depend on its untrusted host. On the flip side, a blockchain is unable to guarantee confidentiality as its state is shared, whereas TEEs offer verifiable computation with a confidential state through remote attestation. By combining the strengths of both technologies, new and exciting use cases can be enabled.

Research on combining TEE with blockchain has focused on various areas, including TEE-based consensus mechanisms, TEE-based wallets, and confidential smart-contract execution using TEE. However, the development of hybrid implementations is challenging due to a set of subtle but difficult-to-solve pitfalls that arise from the fundamental limitations of TEEs. For example, malicious hosts can terminate TEE execution at any point, posing a risk of a lost or conflicting state. Additionally, software running inside an enclave does not have reliable means of determining time, which makes it difficult to validate the recentness of the provided blockchain state. Finally, a TEE must always be able to recognize a forged blockchain state provided by a malicious host.

In this chapter, we will provide an overview of recent and interesting research into combining trusted execution environments with blockchain technology. Section 5.2 will describe TEE and Intel's SGX platform in more detail. Section 5.3 highlights two research projects on combining TEE and blockchain technology in section 5.3 we find particularly interesting. Finally, in section 5.4, we assess both frameworks in terms of the amount of additional privacy and trust they achieve.

5.2. Background

A trusted execution environment is a tamper-resistant processing environment that runs on a separation kernel [56]. A separation kernel is a concept first introduced by Rushby in 1981 as a way to simplify the development and verification of large complex security kernels [57]. By this technique, the system is divided into a number of partitions between which information flow cannot occur unless explicitly permitted. Typically, the separation kernel isolates trusted applications running inside the TEE from the client application running in the untrusted rich execution environment (REE). Rich in REE refers to the feature richness one would expect from modern-day operating environments. TEEs on the other hand provide a limited set of features intended only to address the security-critical part of an application.

The isolation mechanism of TEEs is typically hardware-enforced but software-based solutions do exist [58]. The more common option prevents the simulation of the TEE by a malicious entity through a so-called hardware root of trust. This root of trust consists of a set of private keys directly embedded into the TEE during the manufacturing process. A third party with knowledge about the corresponding public keys can therefore verify the integrity of the TEE. This process is called remote attestation and is what TEEs rely on to build trust in a system.

Intel Software Guard Extensions (SGX) is one such trusted execution environment that allows for secure computation on untrusted platforms [54]. SGX enables secure computation by allowing application developers to partition their code and data into isolated regions called enclaves. These enclaves are hardware-protected and can only be accessed by the application itself. This ensures that the application's secrets, such as cryptographic keys or passwords, cannot be leaked or tampered with by the underlying operating system, hypervisor, or any other application running on the same machine. Intel SGX also supports remote attestation, allowing users to verify the identity and integrity of an SGX enclave even if they do not have direct access to the hardware it is running on.

5.3. Promosing Solutions

In this section, we will consider two research projects that have integrated trusted execution environments with blockchain. Both frameworks aim to achieve confidential transactions in smart contracts, are reasonably successful in doing so, work with Hyperledger Fabric (HLF), and are open-source. However, their approach differs greatly. The work of [51] achieves confidentiality in layer 1 of the blockchain, by performing smart contract execution directly in the TEE of peers. Meanwhile, [52] achieves confidentiality in layer 2 by decoupling smart contract execution from the underlying blockchain system and executing them in an independent trusted network outside the blockchain.

5.3.1. HLF Private Chaincode

Brandenburger *et al.* introduces an architecture and implements a prototype[1] to perform secure smart contract execution using Intel SGX for Hyperledger Fabric [51]. Recall from chapter 2 that in HLF after a peer receives a transaction proposal, the transaction gets simulated, and an endorsement containing the proposed state changes is produced on the ledger. For some use cases, the proposal, the endorsement, the ledger, and the intermediate simulation results may contain sensitive information that is confidential. In addition to the leakage of confidential information to other network participants, we must also consider the threat peers themselves pose. Peers could become malicious, but a bigger concern is that by compromising sensitive information a peer could potentially maximize its own profits. It is important to acknowledge peers have complete control over the OS and applications (including chaincode) and can view all data residing in memory. Sealed-bid auctions and e-voting are examples of use cases that need to consider untrusted peers.

Approach The authors propose to equip every peer with an SGX-enabled CPU and perform all chaincode executions inside the enclave. This approach keeps the sensitive information private as a peer cannot observe code execution or data inside the enclave. As a result, the chaincode running in an enclave will perform according to its specification and reveal no information apart from the resulting state change. However, a malicious peer still fully manages the inputs received by the chaincode running in the enclave. Furthermore, when the chaincode accesses assets on the ledger the peer may feed it a stale or incorrect blockchain state.

An ordering service that produces a signed sequence of transactions should mitigate this attack vector. In [51] the ordering service is assumed to be trusted in the sense that it cannot be rolled back. Subsequently, an enclave is able to verify transactions originate from the orderer, have not been tampered with, and are in proper order. The enclave should also track transaction history to prevent transaction ordering violations or replaying. A malicious peer might still provide a stale blockchain state, but the resulting state change will be discarded by the ordering service like other outdated transactions.

Implementation Figure 5.1 presents the peer architecture proposed by [51], four components are added to enable confidential transactions on Hyperledger Fabric using SGX enclaves. The chaincode enclave enables the isolated execution of a single chaincode and has access to HLF functionalities through an integrated chaincode library. The ledger enclave enables all chaincode enclaves to verify the integrity of the blockchain state. The enclave registry maintains a list of all chaincode enclaves in the network and is responsible for attesting the chaincode actually runs inside of an enclave. Finally, the Enclave Transaction Validator ensures the validity of transactions created by the chaincode enclave by checking that the result has a valid signature of a registered chaincode.

[1] Fabric Private Chaincode: `https://github.com/hyperledger/fabric-private-chaincode`

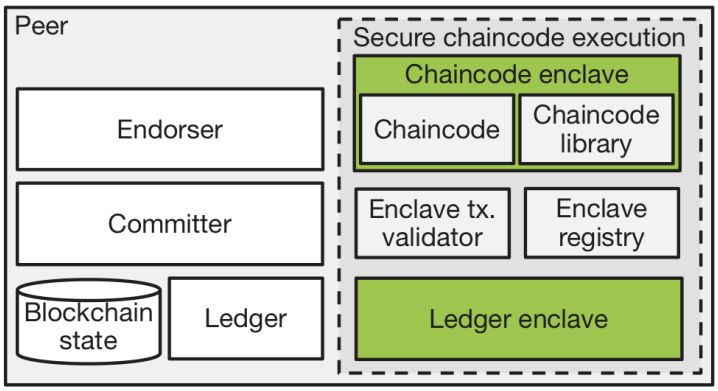


Figure 5.1: Peer architecture of HLF Private Chaincode. The dashed box contains all the new components required for this approach. Green components are expected to operate inside SGX enclaves. The figure is derived from [51].

When a peer joins a channel, its ledger enclave is initialized by the admin. During initialization, the blockchain configuration containing the identities of peers, clients, and ordering services is embedded in the ledger enclave. The ledger enclave then generates a public/private key pair by which it can be uniquely identified. The peer initialization is completed by instantiating the enclave registry.

The admin is also responsible for installing the chaincode enclave. Like the ledger enclave, the chaincode enclave generates a public/private key pair by which it can later be identified. The chaincode enclave will then register itself with the enclave registry and produce an attestation report to prove it was properly installed. If the enclave registry is able to verify the authenticity of the report it will sign the attestation result and together with the chaincode enclave's public key share it on the ledger. At this point, the admin has the option to inject other secrets, for example, an encryption key, into the chaincode enclave for later use. The last step of the chaincode enclave initialization process is binding the enclave to the ledger enclave through another round of attestation.

Before a client provides sensitive information to this chaincode, they will query the chaincode's public key and signed attestation result. The client verifies the authenticity of the report, encrypts a transaction proposal using the obtained public key, and sends this encrypted proposal to the peer. Only the chaincode enclave with the corresponding private key is able to decrypt the proposal and generate a signed response. The response is sent back to the client and submitted to the ordering service. To finalize the transaction, the ordering service broadcasts the newly proposed operations to all endorsing peers. The endorsing peers' enclave transaction validator checks the transactions are signed by a registered chaincode enclave and commits the transaction to the ledger.

5.3.2. Ekiden

Cheng *et al.* present Ekiden, a system that combines blockchains with trusted execution environments [52]. Ekiden takes a different approach to enabling efficient confidentiality-preserving smart contracts. It adopts an architecture where smart contract execution is separated from the consensus mechanism. An off-chain network consisting of SGX-enabled computing nodes is employed to perform smart contract computations over private data. The correct execution of these computing nodes is attested on-chain by the regular consensus nodes which do not need to use trusted hardware. Through this design, Ekiden is blockchain-agnostic and can be deployed along Ethereum smart contracts as well as Hyperledger Fabric chaincode. HLF peers become consensus nodes in this system and are tasked with validating attestation reports generated by compute nodes.

Approach By decoupling smart contract execution from the blockchain consensus mechanism, Ekiden avoids the computational burden and latency of on-chain execution. Compared to public blockchains like Ethereum, the prototype of [52] achieves 600 times more throughput. For permissioned blockchains like Hyperledger Fabric, this applies to a lesser extent, as their consensus mechanisms are more efficient. However, they could also benefit from performing costly cryptographic computations once in an off-chain network instead of multiple times during endorsement.

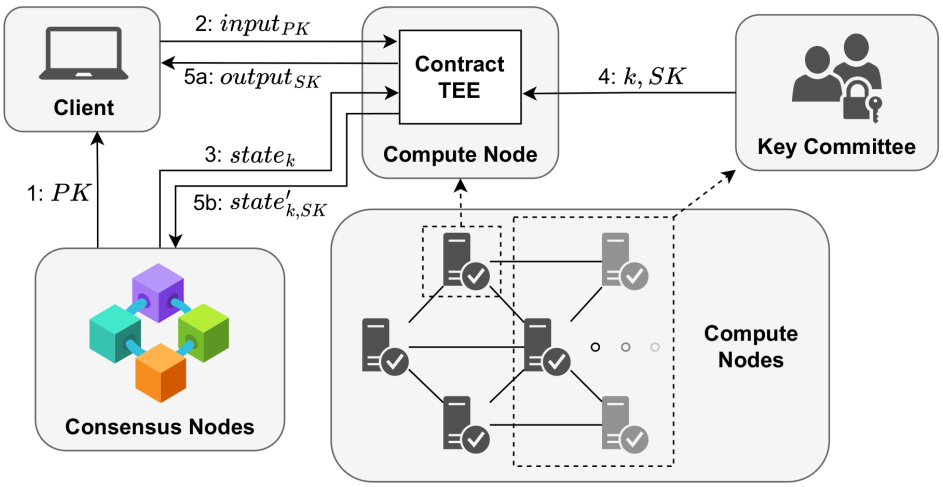


Figure 5.2: System architecture and trusted smart contract execution of Ekiden [52]. The client is connected to a single compute node, while a subset of compute nodes forms a key management committee.

Harmonizing TEEs and blockchain in a hybrid system through a separate network also poses new technical challenges. For instance, TEEs cannot ensure availability and should therefore be treated as expendable and interchangeable. Ekiden solves this by making the blockchain responsible for resolving issues resulting from concurrency. As discussed by [51], another problem of TEEs is the lack of a trusted source of time. The approach taken by [52] is to not rely on the TEE to distinguish a stale or incorrect blockchain state. Instead, Ekiden relies on the blockchain to reject updates based on incorrect states. This is accomplished by letting consensus nodes verify that the transaction result coming from a compute node was computed using a valid and recent ledger state. The last challenge is persisting the symmetric key used for encrypting the private blockchain state. The approach taken by Ekiden is to replicate the keys across multiple TEEs. Key multiplication increases the risk of a data breach posed by side-channel attacks. To mitigate this risk, Ekiden uses short-lived keys for encryption and decryption of the blockchain state. The encryption keys are derived from a less-exposed long-term master key that is distributed using a secret sharing scheme.

Implementation Figure 5.2 contains an overview of the system proposed by [52]. Ekiden distinguishes three types of entities: clients, consensus nodes, and compute nodes. Clients are end users of smart contracts. Consensus nodes maintain the distributed ledger in the case of Hyperledger Fabric these are the peers. Compute nodes are the TEE-enabled nodes for processing requests from clients. Each compute node contains one or multiple contract TEEs, SGX enclaves, for the trusted execution of a smart contract. A quorum of compute nodes forms a key management committee that can manage and generate keys for the contract TEEs to use.

We define a smart contract to be a function f that expects a blockchain state s and an input parameter i and generates a new state s' and output o, i.e. $(s', o) := f(s, i)$. To create a smart contract f in the Ekiden system, a client connects with a compute node and provides f. The compute node creates a contract TEE by loading c into an SGX enclave. During initialization, the contract TEE obtains a public/private key pair (PK, SK) and symmetric key k from the key management committee. The TEE creates an initial blockchain state s_k encrypted under k and an attestation ϕ proving a correct initialization. The compute node verifies ϕ by forwarding it to the Intel Attestation Service (IAS) and generates attestation result π. The compute node then sends f, PK, s_k, and π to the consensus nodes. Consensus nodes verify π before committing f, PK, and s_k to the ledger.

Figure 5.2 shows the steps involved in the execution of a previously registered smart contract. First, the client obtains the public key PK associated with the smart contract (1). The client computes $input_{PK}$ by encrypting its input i under PK and sends a request containing $input_{PK}$ to the compute node (2). The compute receives the request and loads the smart contract code f into the contract TEE. It then loads the current state $state_k$ from the blockchain (3). From the key management committee the contract TEE

receives the short-lived symmetric key k, and private key SK (4). Using PK and k, the TEE decrypts $input_{PK}$ and $state_k$, respectively, it obtains input i and state s. It then executes the smart contract, i.e. $(s', o) := f(s, i)$, it obtains new state s' and outputs o. The TEE optionally encrypts o and signs it using SK, and the result $output_{SK}$ is returned to the client (5a). The consensus nodes receive a state update requesting containing $state'_{k,SK}$, the encryption of s' under k and signed using SK. When the signature is verified by these nodes the ledger is updated.

5.4. Discussion & Conclusion

HLF Private Chaincode puts much emphasizes on ordering service and requires it to be fully trusted. However, the potential security implications in scenarios where this requirement is not satisfied are not well defined. An additional concern with [51] is the system administrator. The admin is responsible for the initialization of the ledger and chaincode enclaves running on all the peers in the network. This centralization introduces a single point of failure into the system. Furthermore, it will be challenging to find a mutually trusted administrator for a consortium consisting of multiple organizations with different interests.

The solution provided by [52], seems to be more suitable for public blockchains. Permissioned blockchains like Hyperledger Fabric have a more efficient mechanism for consensus that benefits less from the proposed off-chain smart contract execution. Furthermore, a lot is unclear about the blockchain state that is required for smart contract execution in compute nodes. Up until the compute node starts execution this state remains unknown. To ensure proper operation, a consensus node must either provide the complete blockchain state to the TEE before execution or, alternatively, the TEE must be able to request the necessary state during execution. The former option appears to be infeasible due to the substantial size of the ledger and the associated network overhead. The latter option seems more logical, but in scenarios with a high number of sequential reads this approach, may result in inadequate performance due to the latency of network requests.

While the results and benefits presented by both solutions look promising, their solutions introduce new problems. Integrating TEEs directly in peers as proposed by [51] requires a system administrator to initialize peers and chaincode. Instead, letting the confidential transactions be processed by a TEE-enabled trusted third party can result in an unacceptable performance overhead. It will be interesting to see if future research is able to solve these challenges.

6

Blockhain-based Consignment Notes

The Contract for the International Carriage of Goods by Road (CMR) facilitates safe, efficient, and compliant cross-border transportation of goods within the European Union. Currently, the paper-based consignment note is being replaced by an electronic version. Organizations are however reluctant to switch, as current proprietary implementations of the digital protocol lack transparency and trust. Furthermore, these platforms are unable to provide conclusive evidence about the deliverance of goods, leaving the door open for illegal activities. Through blockchain technology, we propose a design that offers a transparent, and traceable alternative. Our solution employs Hyperledger Fabric to facilitate performant and confidential transactions for all parties involved in the consignment. Furthermore, we fit our previously implemented privacy-enhancing technologies into the design to offer additional functionality.

6.1. Introduction

Advancements in human civilization have been closely related to the exchange of goods and thus to the development of transportation. Efficient transportation is a prerequisite for our increasingly globalized economies and the operation of international trade. Within the EU, the vast majority of goods are transported by road. The dominant position of road transport can be explained by the high quality and density of the road network in the EU and the absence of a reliable rail network. Additionally, road transport provides flexibility, speed, convenience, and adaptability among other advantages compared to other forms of transportation [59]. To facilitate safe, efficient, and compliant cross-border transportation a transport contract is required to accompany the goods. The contract outlines the terms and conditions under which the goods will be transported, including the responsibilities of all parties involved. Differing laws and regulations between countries in Europe regarding such a contract would make compliance with regulations complicated and could harm the development of international trade. Fortunately, in Geneva in 1956 the United Nations (UN) proposed the convention on the contract for the international carriage of goods by road (CMR) [60].

The purpose of the CMR convention is to standardize the conditions for the contract of cross-border transport of goods. Currently, the convention is signed by 58 countries, including virtually all European nations and a number of countries in northern Africa, the Middle East, and Central Asia [61]. The International Road Transport Union (IRU) has developed a CMR consignment note in compliance with the CMR convention. The consignment note consists of three color-coded carbon copies and represents an acknowledgment of receipt and delivery of the goods [62]. Although the convention does not specify that the transport contract has to be in paper form, based on the language used in its provisions and from the date of adoption, it can be safely assumed the regulation refers to a physical document. In an increasingly digitalized world, the administrative burden of documents in paper format was prevalent. On top of that, a digital variant of the contract significantly reduces costs, improves the visibility and efficiency of the supply chain, and lowers the environmental footprint of the process [63]. These insights caused the UN to propose an additional protocol to the CMR concerning the electronic consignment note (e-CMR) [64].

The first cross-border usage of e-CMR took place on 19 January 2017, between Spain and France. Currently, 33 countries have ratified the electronic addition out of the 58 states that signed the original convention [65]. It seems like the perfect time for organizations to make the switch to digital, now that many EU countries, including major economies and critical transit nations such as Germany and France, have started down this path. But a switch to what? In contrast to the paper consignment note developed for the original convention, the e-CMR convention lacks a standard implementation. Several solutions have been proposed by companies in the Netherlands, Denmark, Spain, France, and the United Kingdom among others. The IRU has partnered up with TransFollow, a platform developed by a Dutch company, and recommends this e-CMR implementation. However, like other proprietary software implementations, the solution proposed by TransFollow lacks transparency and trust. All CMR-related requests to their application programming interface (API) are performed in a non-transparent way on their private servers. Understandably, companies will be reluctant to switch to such a platform. Furthermore, tax authorities and other government agencies will demand access to these platforms to monitor the transactions. This is understandable as the current paper consignment note has become an important part of corporate accounting, see section 6.3.1 for more details. Lastly, proprietary e-CMR platforms have inadequate traceability to detect a recent phenomenon called CMR “Neutralization” [66]. Neutralization of transport documents is in most cases associated with illegal logistics activity called “parallel trade”, see section 6.3.2 for more details.

Blockchain technology has shown great potential in easing transparency and traceability-related problems in supply chain management and logistics [67, 68]. But while today we can trace the origin of every cocoa bean in a bar of chocolate in a completely transparent way using distributed ledger technology, transparency and traceability of delivery of goods within the EU are provided by means of 3 sheets of paper. Blockchain seems to be the ideal foundation to build a secure, transparent, and traceable e-CMR system. However, it seems like the scientific community is simply unaware of existing problems in the industry concerning the CMR convention. We were only able to find a single literature survey that deemed the digitization of the CMR consignment note as interesting [69].

In agreement with our previous observations, we propose the first design of a blockchain-based e-CMR

system. Our design offers a decentralized, transparent, and open alternative to the current proprietary implementations of the e-CMR convention. Through the use of the permissioned Hyperledger Fabric framework, our system can offer real-time transactions at a low cost to all consortium members. Furthermore, we will employ our previously developed dynamic searchable symmetric encryption (DSSE) and multi-authority attribute-based encryption (MA-ABE) schemes to provide secure search functionality and fine-grained access control, respectively.

In section 6.2, we will analyze the original CMR convention, as well as, its later electronic addition. In section 6.3, we discuss two CMR-related challenges in industry. Section 6.4 describes our system and provides an architectural overview. Finally, in section 6.5, we conclude our work.

6.2. CMR: A Standard for International Road Transport

CMR stands for “Convention relative au contrat de transport international de marchandises par route” which is a United Nations convention that governs the international carriage of goods by road [60]. The CMR convention lays down the rights and obligations of the parties involved in the carriage of consignments by a road vehicle. It outlines the principles of liability of the carrier, as well as the process for filing damages claims. The convention applies when the place of acceptance of the consignment and the intended place of delivery are in two different states, of at least one is a contracting country of the Convention (Article 1) [60]. The CMR convention demands a consignment note with specific properties accompanying the carriage of goods.

The CMR consignment note is an official document in a standardized format, written in two languages to ease inspection by an authority. It is made up of at least three color-coded copies. The first red copy of the consignment note is intended for the consignor and serves as proof that the consignor has handed over the goods to the carrier. The second blue form is meant for the consignee as proof of receipt of the goods. The third green form is kept by the carrier of the goods and confirms that the carrier has handed over the goods to the consignee. All three copies should be signed by all three participants with exception of the first red form that only contains the signatures or stamps of the consignor and carrier (Article 5) [60].

Article 6 of the CMR convention describes the details the consignment note should contain such as a description of the goods, packaging, and quantities [60]. Furthermore, the consignment note to contain the names and addresses of all participants, as well as the place and date of departure and intended delivery of the goods. The subsequent articles describe the responsibilities and privileges of all participants involved in determining liability.

6.2.1. Digital Equivalent

In 2008, the United Nations signed the “Additional Protocol to the CMR concerning the electronic consignment note” [64]. This addition to the convention explicitly confirms an electronic consignment note is allowed and shall be considered equal to the original paper variant (Article 2). To be considered equal, the electronic consignment note should include the same details (Article 6 of [60]) as its physical version (Article 4). Furthermore, the procedure to issue the electronic consignment note should ensure the integrity of the consignment note’s details and verify these remain complete and unaltered (Article 4). The convention also specifies that implementation of the electronic consignment note should provide all involved parties with a method to obtain the consignment note, as well as the capability to observe changes regarding the delivery of the consignment (Article 5 & 6).

For the electronic consignment note to be legally valid, all the involved parties should authenticate the contract of carriage by means of a reliable electronic signature (Article 3). This electronic signature must:

(a) be uniquely linked to the signatory;

(b) be capable of identifying the signatory;

(c) be created using means that the signatory can maintain under his sole control; and

(d) is linked to the data to which it relates in such a manner that any subsequent change in the data is detectable.

6.3. Industry Challenges

This section will describe two CMR-related challenges encountered in the industry. The proprietary and centralized software systems proposed by competitors are unable to tackle these issues.

6.3.1. Intra-Community Transaction

An intra-community transaction occurs whenever a business sells goods or provides services to a business located in another European Union member state. From the perspective of the seller, the transaction is referred to as an Intra-Community Supply (ICS), and from the buyer's perspective, it is called an Intra-Community Acquisition (ICA). A supplier of goods or services may apply the zero rate of value-added tax (VAT) if the following conditions are met [70]:

1. The customer must be an entrepreneur and have a valid VAT registration number in the other EU member state.
2. The goods must actually be dispatched and transported to the other EU member state and the supplier's administration must provide proof of this.

fulfilling the first requirement is simple by using the VAT number validation tool provided by the European Union[1]. Satisfying the second criterion is harder. The administration of the supplier will generally only contain some customer details, an order confirmation, and an invoice. As of January 1, 2020, the requirements for applying the zero VAT rate have become even stricter [71]. The proof required for the second criterion must consist out of 2 non-contradictory pieces of evidence, issued by two independent parties. The new regulation does mention a signed CMR consignment note can serve as one item of evidence. However, the consignment note obtained by the supplier or consignor only contains proof of departure of the goods. Moreover, the consignor's copy of the consignment note does not contain the signature of the consignee.

6.3.2. Neutralization

The term "neutralization" was first coined by Cheu *et al.* in 2019 to describe a business practice used in the transport sector for some years now [66]. Neutralization of a consignment note means canceling the effect and therefore the validity of a consignment note by replacing it with a second consignment note. In the context of neutralization, in addition to the seller/consignor and buyer/consignee of goods, we consider a middle-man we call the trader. By neutralization, the trader aims to keep either the source or the destination of the goods secret from the buyer or seller, respectively. Generally, the trader commissions the carrier to perform the actual neutralization of the CMR consignment note. In a survey conducted by [66] across several transport companies, 66% of the correspondents confirm they perform neutralization of documents at the request of their clients.

The neutralization of CMR consignment notes is fraudulent and all parties involved are at risk of criminal charges [72]. Moreover, the underlying reason for neutralizing transport documents is in most cases associated with an illegal activity in logistics called "parallel trade" [66]. Parallel trade is the cross-border sale of goods within the EU by traders outside of the manufacturer's distribution system without the manufacturer's consent [73]. The product is still authentic but is imported without the permission of the intellectual property owner. For instance, a manufacturer thinks they are shipping their product to a low-wage country with a minimal profit margin, while in reality the product is sold by a trader with high margins outside of the official channels of the manufacturer.

6.4. Blockchain-Based E-CMR System

We propose our concept design for the first blockchain-based e-CMR system. Through distributed ledger technology we can issue, observe, and process electronic consignment notes in a decentralized manner, without the need for a trusted third party that introduces a single point of control and failure. All transactions concerning consignments are immutably recorded on the blockchain, thus preventing parties from altering terms in retrospect. Furthermore, the network offers transparency of all the operations and assets to all the parties involved in a consignment. The Hyperledger Fabric framework forms the foundation of the blockchain network. Its permissioned design and ability to interact with other

[1] VIES VAT number validation: `https://ec.europa.eu/taxation_customs/vies/#/vat-validation`

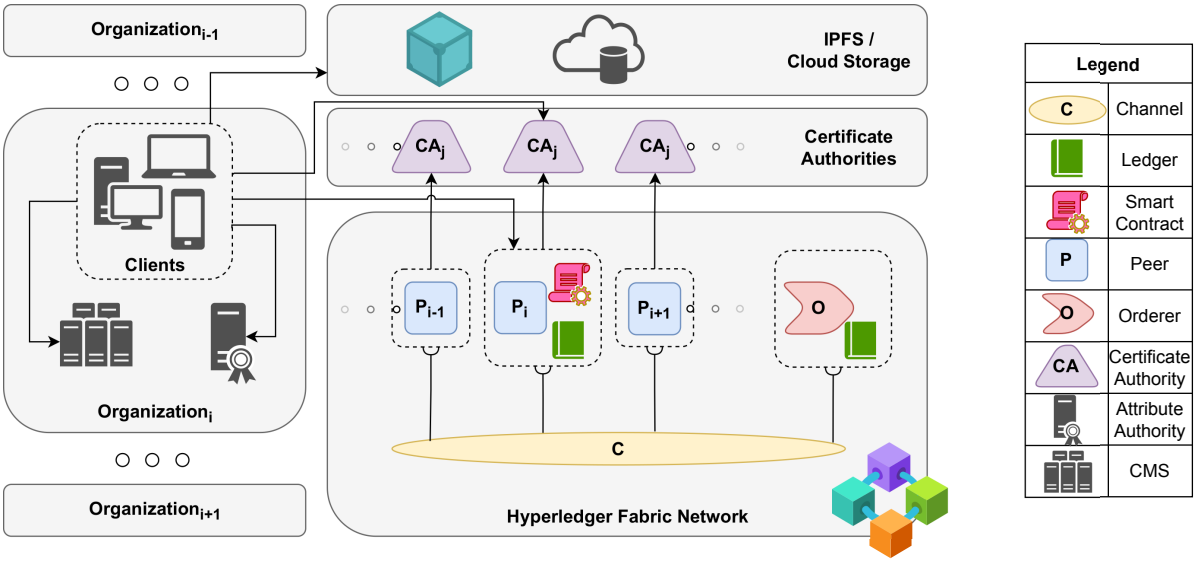


Figure 6.1: Overview of the privacy-enhanced blockchain system.

network members through confidential channels should provide the required level of trust and privacy for this corporate use case. Lastly, we will integrate our previously developed searchable encryption and attribute-based encryption schemes to provide search capabilities over and fine-grained sharing of encrypted consignment notes.

6.4.1. Architecture

In figure 6.1, an overview of the components in our privacy-enhanced blockchain system is provided. Notice that this system is not specifically tailored to solve the challenges concerning electronic consignments notes. Instead, figure 6.1 depicts a general blockchain-based system that could be applied to other use cases in supply chain and logistics or even in different fields such as healthcare or finance. In the remainder of this chapter, however, we will consider the system only in the context of the e-CMR convention.

In figure 6.1, four different environments can be distinguished, i.e. the organizations, blockchain network, certificate authorities, and storage environments. In the context of the electronic consignment use case, the organizations to consider are the consignor, carrier, and consignee. An organization is represented consisting of three entities. The content management system (CMS) represents the yet existing software stack of the organization. In the case of a logistics company, this stack generally consists of an enterprise resource planning (ERP) software system. Clients are managed by members of the organization and run the applications that interact with all the other entities in the system. At last, within the context of an organization we consider the attribute authority. The organization's members require private keys served by the attribute authority to decrypt assets related to electronic consignment notes on-chain and in the storage system(s). Clients can locally perform the MA-ABE decryption algorithm using the foreign function interface described in chapter 4.

The blockchain infrastructure consists of the standard Hyperledger Fabric components already discussed in chapter 2, to keep the diagram concise, membership service providers were left out. Each organization in the system manages a peer in the HLF network. All the peers are connected to the same channel. An ordering service ensures all peers receive the transactions on the channel in consistent order. Each peer maintains a local copy of the assets in the network in its local ledger. We consider electronic consignment notes by assets in this context. The business logic to interact with assets is contained in smart contracts and installed on the peers in the form of a chaincode. The chaincode contains the logic to issue and sign consignment notes, but also the DSSE-related algorithms for querying consignments. Relevant keywords concerning consignment notes could include the date of departure, the names and addresses of parties involved, and the type of goods.

Certificate authorities (CAs) are depicted as a separate category since they are not 1-to-1 related to

organizations. Peers need to authenticate themselves through a trusted CA to perform actions in the blockchain network. Similarly, clients need to acquire a certificate generated by a CA that verifies their identity. The identity provided through the certificate issued by a CA is especially important to clients considering the MA-ABE scheme. As described in chapter 4, this identity serves as a linchpin for tying users' keys together and preventing collusion.

The last environment in figure 6.1 is the data storage environment. A concrete implementation of the proposed system would have the option to utilize the InterPlanetary File System (IPFS), a cloud storage provider, or any other type of data storage system. Consignment nodes may contain a lot of information and be accompanied by additional documents e.g. invoices, photos, etc. The blockchain is already constantly growing as its an immutable and append-only data structure. It is therefore ill-advised to use the blockchain to store big chunks of data such as a consignment note and its associated files. Therefore, we will use a distributed file storage system like IPFS or a cloud storage provider like Amazon Web Services (AWS) to store the consignment note. A hash of the consignment note that uniquely identifies the note is stored on the blockchain. By using a secure hashing algorithm we are assured that any attempt to alter the details of the consignment will result in a different hash value of the consignment note and will therefore be detected [74].

6.4.2. Compliance with E-CMR protocol

In section 6.2, we mentioned several requirements of an e-CMR protocol implementation. In this section, we track back and highlight how our proposed design implements these requirements.

In our proposed system, an electronic consignment note f is considered a structured data string or JSON. The e-CMR convention demands an electronic consignment note should contain the same details as its carbon equivalent (Article 4) [64]. To meet this requirement, we define the fields of f in accordance with the particulars defined by Article 6 of the original convention [60]. Furthermore, we define dedicated fields in f for arranging liability concerns in accordance with the other articles [60].

Encryption As per the convention (Article 5 & 6) we should provide all parties with a means to obtain f. We already discussed why saving f directly on-chain is not a good idea due to storage concerns of the ledger. However, f contains confidential information and can therefore not be shared in plaintext with an untrusted storage provider. We develop the following method to overcome this problem. First we pick a secure symmetric encryption scheme $\mathsf{SE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such as AES and generate random key $k \leftarrow \mathsf{Gen}()$ and compute the encryption of f, $c_f := \mathsf{Enc}_k(f)$. Then we pick a secure hash algorithm H such as SHA-3 and obtain the hash of the obtained cipher $h_f := H(c_f)$. We send h_f and c_f to the storage provider, who can later provide us c_f when receiving a request with h_f.

To allow fine-grained sharing of the encrypted consignment note, we will employ our MA-ABE scheme discussed in chapter 4. We assume the system setup and attribute authority setup have been performed and the public keys resulting from these operations are generally known by all clients. We define an access policy $\mathbb{A} = S \vee C \vee B$, where S, C, B are boolean formulas made up of attribute literals corresponding to the seller (consignor), carrier, and buyer (consignee) of f, respectively. For example, a trivial access policy could look like $\mathbb{A} =$ *merchant@seller* $\vee$ *driver@carrier* $\vee$ *purchaser@buyer*. We define the tuple $d_f = (h_f, k)$ and using the system parameters, the public keys of the authorities of the involved parties, and $\mathbb{A}$ we compute $C_f = \mathsf{Enc}_{\mathsf{ABE}}(d_f)$. Through a smart contract invocation, C_f gets saved on-chain.

Decryption All clients within organizations connected to the channel can now obtain C_f. A client obtains proof of their identity GID from a certificate authority. Using this identifier they can request private keys from their organization's attribute authority. When the set of attributes satisfies the access policy defined for C_f, the client can perform the MA-ABE decryption algorithm locally, they obtain $d_f = (h_f, k)$. A request to the storage provider, specifying h_f, yields c_f. Using the decryption algorithm of the SE scheme defined in the previous paragraph, the client obtains $f = \mathsf{Dec}_k(c_f)$. The client can verify the contents of the electronic consignment note by examining the fields of f, and determine whether they wish to sign off on the consignment.

Signing Article 3 of the e-CMR convention demands the electronic signature has several properties [64], which are enlisted in section 6.2.1. A secure electronic signature can be achieved in multiple ways. For this design, we propose to use the Elliptic Curve Digital Signature Algorithm (ECDSA). ECDSA is a type of secure public-key cryptosystem for generating digital signatures. We define a public-key digital signature scheme DSS $= (\text{Gen}, \text{Sign}, \text{Verify})$ consisting of a generation, signing, and verification algorithm. A client that wants to sign off a consignment first generates public/private key pair $(PK, SK) \leftarrow \text{Gen}()$. For each organization, the blockchain maintains a distinct set of public keys. Through a smart contract invocation, the client adds PK to the set of their organization, and the contract verifies the client's membership in the organization. The client recovers C_f from the ledger and signs it using its private key and obtains the tag $t_f := \text{Sign}_S K(C_f)$ (we assume any hashing of C_f to be part of the signing algorithm). The client then saves its signature on the blockchain using another smart contract. This smart contract obtains C_f and PK, and checks acceptance of $\text{Verify}_P K(C_f, t_f)$ before finalizing the transaction.

Search Through encryption and decryption of the consignment notes, we can provide private access to notes for all the parties involved in the consignment. However, the current system is unworkable for clients that are looking for a specific consignment f of which the ciphertext C_f is unknown to them. Therefore, we propose to use our DSSE scheme described in chapter 3 to obtain searchable consignment notes.

We start implementing search by running the DSSE key generation algorithm to obtain symmetric key k. We then initialize an empty index structure on the ledger by running the setup algorithm with an empty set of files. Whenever a CMR consignment note f is issued, we go through the procedure of encryption to obtain C_f. The consignment issuer can now select a number of keywords from f, for instance, the consignment number, names of the involved parties, type of goods, and the date. This keyword set W_f is then used to generate an add token τ_a. By calling our implementation of the add function in chaincode, we associate these keywords with the cipher C_f.

Later, for example, if a client wants to retrieve all files from a certain. The client requests a search token from the consignment note issuer and provides a keyword, the date, w. The client obtains search token τ_s. Through the search chaincode, the client obtains all consignment notes from the date w by providing τ_s.

6.4.3. Solving Industry Challenges

In our proposed blockchain-based e-CMR system, we use unforgeable signatures that are saved on an immutable ledger to track the progress of the consignment. Therefore, in our system, the proof of delivery of a consignment is conclusive. Through the implemented MA-ABE scheme, all parties involved in the consignment can observe this proof. A consignor can therefore safely apply the zero VAT rate to all its intra-Community supplies. Tax authorities could even be added to the blockchain network to observe the process.

Tomicová *et al.* has previously examined the impact the e-CMR protocol could have on the neutralization of consignment notes [75]. they concluded the adoption of the electronic consignment note could reduce neutralization. However, this conclusion is based on the assumption fraud is prevented because the signer's device registers the GPS location during document signing in the TransFollow application. A simple GPS spoofer bypasses this protection with ease, however. Our system prevents the neutralization of consignment notes altogether by just not supporting this type of transaction. A consignment may be canceled only according to a specific policy defined during the issuance of the consignment and agreed upon by all parties.

6.5. Conclusion

Blockchain technology may play a critical role in the future of the electronic CMR consignment note. Distinctive features of blockchain such as immutability and transparency align seamlessly with the requirements of the e-CMR convention. Our proposed design for a privacy-enhanced blockchain system additionally provides performant and confidential transactions to all participants of the network. Current challenges encountered in the industry concerning the CMR protocol are also solved in our design. It would be interesting to see how a concrete implementation of our design performs for this use case

compared to existing proprietary solutions. Additionally, it would be interesting to conduct a case study to gauge how companies view an open alternative. Lastly, we would like future work to consider other use cases and problems that could be tackled by the designed system.

7

Conclusion

In this thesis, we proposed several tools to increase the set of privacy-focused features at the disposal of members of a consortium blockchain. While combining blockchain technology with trusted execution environments sounded interesting, we deem the technology to be unfit for use in a decentralized network. On the other hand, we successfully implemented both searchable encryption and attribute-based encryption in Hyperledger Fabric without introducing a central authority. Our attribute-based encryption implementation uses a novel and generic approach to foreign function invocation by the means of protocol buffer serialization. Furthermore, we implemented the algorithms of a provably adaptive secure dynamic searchable symmetric encryption scheme in smart contracts. To our knowledge, this is the first implementation that achieves this. We submitted a paper containing this result to the ESORICS 2023 conference. Lastly, we proposed a blockchain-based system to solve the transparency and traceability issues concerning proprietary software implementations of the electronic consignment CMR protocol.

7.1. Research Questions

After conducting the research, we are ready to present the answer to the research question.

How to enhance the trust and functionality of consortium blockchain applications through the deployment of privacy-enhancing technologies?

We showed that both searchable encryption, as well as attribute-based encryption, provides consortium blockchains with extra functionality while maintaining the confidentiality of the data to which these functionalities apply. Searchable encryption provides a client with the ability to outsource a data set to an untrusted third party while maintaining search capability. In a blockchain context, a data owner additionally achieves verifiability of its search results. Attribute-based encryption (ABE) provides a data owner with fine-grained sharing of confidential information. A multi-authority ABE keeps the consortium network decentralized while also allowing data owners to define sophisticated access policies for their private data.

1. **How can smart contracts provide search functionality over dynamically encrypted data stored on the blockchain?** Through the implementation of a dynamic searchable symmetric encryption scheme in smart contracts, we can achieve search functionality over encrypted data on the blockchain. The smart contracts require a token issued by the owner of the data to perform their search or update functionality.
2. **How can we share encrypted data only with a subset of users in the blockchain consortium?** We can assign this subset of users in the consortium a specific set of attributes. Using a multi-authority attribute-based encryption scheme, consortium members can transform these attributes in access policy on encrypted data and private keys without introducing a central authority to the network.
3. **How can we employ trusted execution environments to enforce correct smart contract execution and what are the limitations?** Correct smart contract execution can be enforced by

trusted execution environments (TEEs) in two distinct ways. Either the TEE is contained by the peers that also perform consensus, or, a separate network of TEE-enabled nodes is used. Both approaches are limited due to the trusted admin setup.

4. **What are the benefits and limitations of a privacy-enhanced blockchain solution that incorporates the technologies mentioned in the previous research questions?** A privacy-enhanced blockchain provides extra functionalities to its users. These functionalities are opt-in, so there is no real disadvantage compared to a non-enhanced blockchain.
5. **What new use cases can be tackled by this privacy-enhanced blockchain-based system?** We discussed in detail how a privacy-enhanced blockchain-based system tackles challenges concerning the implementation of an electronic consignment note in logistics. We expect other privacy-sensitive use cases in the fields of healthcare or finance could be tackled as well.

Bibliography

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, p. 21 260, 2008.

[2] O. Dib, K.-L. Brousmiche, A. Durand, E. Thea, and E. B. Hamida, "Consortium blockchains: Overview, applications and challenges," *Int. J. Adv. Telecommun*, vol. 11, no. 1, pp. 51–64, 2018.

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

[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*, IEEE, 2000, pp. 44–55.

[5] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965–976.

[6] L. Chen, W.-K. Lee, C.-C. Chang, K.-K. R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future generation computer systems*, vol. 95, pp. 420–429, 2019.

[7] B. B. Gupta, K.-C. Li, V. C. Leung, K. E. Psannis, S. Yamaguchi, *et al.*, "Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 12, pp. 1877–1890, 2021.

[8] C. Xu, L. Yu, L. Zhu, and C. Zhang, "A blockchain-based dynamic searchable symmetric encryption scheme under multiple clouds," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 3647–3659, 2021.

[9] M. Li, C. Jia, R. Du, W. Shao, and G. Ha, "Dse-rb: A privacy-preserving dynamic searchable encryption framework on redactable blockchain," *IEEE Transactions on Cloud Computing*, 2022.

[10] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM (JACM)*, vol. 43, no. 3, pp. 431–473, 1996.

[11] E.-J. Goh, "Secure indexes," *Cryptology ePrint Archive*, 2003.

[12] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International conference on applied cryptography and network security*, Springer, 2005, pp. 442–455.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 79–88.

[14] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International conference on the theory and application of cryptology and information security*, Springer, 2010, pp. 577–594.

[15] K. Kurosawa and Y. Ohtaki, "Uc-secure searchable symmetric encryption," in *International conference on financial cryptography and data security*, Springer, 2012, pp. 285–298.

[16] R. Bost, P.-A. Fouque, and D. Pointcheval, "Verifiable dynamic symmetric searchable encryption: Optimality and forward security," *Cryptology ePrint Archive*, 2016.

[17] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2011, pp. 568–588.

[18] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the theory and application of cryptographic techniques*, Springer, 1984, pp. 47–53.

[19] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Annual international cryptology conference*, Springer, 2001, pp. 213–229.

[20] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2005, pp. 457–473.

[21] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *2007 IEEE symposium on security and privacy (SP’07)*, IEEE, 2007, pp. 321–334.

[22] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, “Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption,” *IEEE transactions on parallel and distributed systems*, vol. 24, no. 1, pp. 131–143, 2012.

[23] X. Yao, Z. Chen, and Y. Tian, “A lightweight attribute-based encryption scheme for the internet of things,” *Future Generation Computer Systems*, vol. 49, pp. 104–112, 2015.

[24] M. Raikwar, D. Gligoroski, and K. Kralevska, “Sok of used cryptography in blockchain,” *IEEE Access*, vol. 7, pp. 148 550–148 575, 2019.

[25] R. Zhang, R. Xue, and L. Liu, “Security and privacy on blockchain,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–34, 2019.

[26] M. Chase, “Multi-authority attribute based encryption,” in *Theory of cryptography conference*, Springer, 2007, pp. 515–534.

[27] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 89–98.

[28] Y. Rouselakis and B. Waters, “Practical constructions and new proof methods for large universe attribute-based encryption,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 463–474.

[29] Y. Rouselakis and B. Waters, “Efficient statically-secure large-universe multi-authority attribute-based encryption,” in *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, Springer, 2015, pp. 315–332.

[30] N. Attrapadung and H. Imai, “Conjunctive broadcast and attribute-based encryption,” in *Pairing-Based Cryptography–Pairing 2009: Third International Conference Palo Alto, CA, USA, August 12-14, 2009 Proceedings 3*, Springer, 2009, pp. 248–265.

[31] T. Nishide, K. Yoneyama, and K. Ohta, “Attribute-based encryption with partially hidden encryptor-specified access structures,” in *Applied Cryptography and Network Security: 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings 6*, Springer, 2008, pp. 111–129.

[32] S. Agrawal and M. Chase, “Fame: Fast attribute-based message encryption,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 665–682.

[33] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, “Dac-macs: Effective data access control for multiauthority cloud storage systems,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, 2013.

[34] Y. Rahulamathavan, R. C.-W. Phan, M. Rajarajan, S. Misra, and A. Kondoz, “Privacy-preserving blockchain based iot ecosystem using attribute-based encryption,” in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, IEEE, 2017, pp. 1–6.

[35] H. Wang and Y. Song, “Secure cloud-based ehr system using attribute-based cryptosystem and blockchain,” *Journal of medical systems*, vol. 42, no. 8, p. 152, 2018.

[36] D. Boneh, P. Papakonstantinou, C. Rackoff, Y. Vahlis, and B. Waters, “On the impossibility of basing identity based encryption on trapdoor permutations,” in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 2008, pp. 283–292.

[37] V. Odelu, A. K. Das, M. K. Khan, K.-K. R. Choo, and M. Jo, “Expressive cp-abe scheme for mobile devices in iot satisfying constant-size keys and ciphertexts,” *IEEE Access*, vol. 5, pp. 3273–3283, 2017.

[38] M. Venema and G. Alpár, “A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption,” in *Topics in Cryptology–CT-RSA 2021: Cryptographers’ Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings*, Springer, 2021, pp. 100–125.

[39] A. Beimel *et al.*, “Secure schemes for secret sharing and key distribution,” *PhD thesis*, 1996.

[40] A. Joux and K. Nguyen, “Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups,” *Journal of cryptology*, vol. 16, no. 4, pp. 239–247, 2003.

[41] D. M. Freeman, “Converting pairing-based cryptosystems from composite-order groups to prime-order groups,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2010, pp. 44–61.

[42] A. B. Lewko, “Tools for simulating features of composite order bilinear groups in the prime order setting.,” in *Eurocrypt*, Springer, vol. 7237, 2012, pp. 318–335.

[43] P. Liang, L. Zhang, L. Kang, and J. Ren, “Privacy-preserving decentralized abe for secure sharing of personal health records in cloud storage,” *Journal of Information Security and Applications*, vol. 47, pp. 258–266, 2019.

[44] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Succinct non-interactive zero knowledge for a von neumann architecture,” in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 781–796.

[45] I. Grishkov, R. Kromes, T. Giannetsos, and K. Liang, “Id-based self-encryption via hyperledger fabric based smart contract,” *arXiv preprint arXiv:2207.01605*, 2022.

[46] V. Shoup, “Lower bounds for discrete logarithms and related problems,” in *Advances in Cryptology—EUROCRYPT’97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings 16*, Springer, 1997, pp. 256–266.

[47] A. W. Dent, “Adapting the weaknesses of the random oracle model to the generic group model,” in *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8*, Springer, 2002, pp. 100–109.

[48] N. Koblitz and A. Menezes, “Another look at generic groups,” *Cryptology ePrint Archive*, 2006.

[49] O. Dib and K. Toumi, “Decentralized identity systems: Architecture, challenges, solutions and future directions,” *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pp. 2516–0281, 2020.

[50] B. C. Ghosh *et al.*, “Decentralized cross-network identity management for blockchain interoperation,” in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–9.

[51] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti, “Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric,” *arXiv preprint arXiv:1805.08541*, 2018.

[52] R. Cheng *et al.*, “Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2019, pp. 185–200.

[53] S. Pearson and B. Balacheff, *Trusted computing platforms: TCPA technology in context*. Prentice Hall Professional, 2003.

[54] F. McKeen *et al.*, “Innovative instructions and software model for isolated execution.,” *Hasp@ isca*, vol. 10, no. 1, 2013.

[55] T. Alves, “Trustzone: Integrated hardware and software security,” *Information Quarterly*, vol. 3, pp. 18–24, 2004.

[56] M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted execution environment: What it is, and what it is not,” in *2015 IEEE Trustcom/BigDataSE/Ispa*, IEEE, vol. 1, 2015, pp. 57–64.

[57] J. M. Rushby, “Design and verification of secure systems,” *ACM SIGOPS Operating Systems Review*, vol. 15, no. 5, pp. 12–21, 1981.

[58] U. Lee and C. Park, “Softee: Software-based trusted execution environment for user applications,” *IEEE Access*, vol. 8, pp. 121 874–121 888, 2020.

[59] J. Nowakowska-Grunt and M. Strzelczyk, “The current situation and the directions of changes in road freight transport in the european union,” *Transportation Research Procedia*, vol. 39, pp. 350–359, 2019.

[60] United Nations, *Convention on the contract for the international carriage of goods by road (cmr)*, Geneva, May 19, 1956.

[61] United Nations. “Convention on the contract for the international carriage of goods by road (cmr), of 19 may 1956,” United Nations Treaty Collection. (2023), [Online]. Available: `https://unece.org/list-agreements` (visited on 02/12/2023).

[62] J. C. Ferrer, *The cmr convention-a pillar of international carriage of goods by road*, 2006.

[63] Vrio Europe. “Digital transformation: Ecmr – a digital future for the cmr document,” Vrio Europe. (2020), [Online]. Available: `https://vrioeurope.com/en/digital-transformation-ecmr-a-digital-future-for-the-cmr-document` (visited on 02/12/2023).

[64] United Nations, *Additional protocol to the convention on the contract for the international carriage of goods by road (cmr) concerning the electronic consignment note*, Geneva, May 27, 2008.

[65] United Nations. “Additional protocol to the cmr concerning the electronic consignment note (e-cmr),” United Nations Treaty Collection. (2023), [Online]. Available: `https://unece.org/list-agreements` (visited on 02/12/2023).

[66] K. Cheu, M. Poliak, J. Tomicová, and J. Gnap, “Neutralization of cmr documents,” *Archiwum Motoryzacji*, vol. 83, no. 1, pp. 175–184, 2019.

[67] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, “Blockchain technology and its relationships to sustainable supply chain management,” *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, 2019.

[68] E. Tijan, S. Aksentijević, K. Ivanić, and M. Jardas, “Blockchain technology implementation in logistics,” *Sustainability*, vol. 11, no. 4, p. 1185, 2019.

[69] M. G. Belu *et al.*, “Application of blockchain in international trade: An overview,” *The Romanian Economic Journal*, vol. 22, no. 71, pp. 2–15, 2019.

[70] Tax and Customs Administration. “Intra community supply.” (), [Online]. Available: `https://www.belastingdienst.nl/wps/wcm/connect/bldcontenten/belastingdienst/business/vat/vat_in_the_netherlands/vat_relating_to_purchase_and_sale_of_goods/export_from_the_netherlands_to_other_eu_countries_intra-community_supply/export_from_the_netherlands_to_other_eu_countries_intra-community_supply` (visited on 02/12/2023).

[71] Council of the European Union, *Council implementing regulation (eu) 2018/1912*, Brussel, Dec. 4, 2018.

[72] M. Poliak, J. Tomicová, and M. Jaśkiewicz, “Identification the risks associated with the neutralization of the cmr consignment note,” *Transportation Research Procedia*, vol. 44, pp. 23–29, 2020.

[73] R. Ahmadi and B. R. Yang, “Parallel imports: Challenges from unauthorized distribution channels,” *Marketing Science*, vol. 19, no. 3, pp. 279–294, 2000.

[74] R. Kumar and R. Tripathi, “Implementation of distributed file storage and access framework using ipfs and blockchain,” in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, IEEE, 2019, pp. 246–251.

[75] J. Tomicová, M. Poliak, and N. A. Zhuravleva, “Impact of using e-cmr on neutralization of consignment note,” *Transportation Research Procedia*, vol. 55, pp. 110–117, 2021.

A

Preliminaries

In this appendix, we cover some of the basic concepts, notations, and definitions used throughout this thesis. Appendix A.1 covers mathematical notation semantics and data structures used in definitions, constructions, and proofs of our schemes. Appendix A.2 contains a collection of definitions serving as the backbone of this work.

A.1. Notation

Sets A *set* is a finite or infinite collection of elements. Examples of sets are the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ or the set of all binary strings of length n denoted as $\{0,1\}^n$. We write $x \in X$ to indicate x is an element of the set X. The number of elements a set X contains is referred to as its *cardinality* and is denoted by $|X|$. The *power set* $\mathcal{P}(X)$ of a set X is the set whose elements are the subsets of X, $\mathcal{P}(X) = \{S \subseteq X \mid S\}$. Alternatively, we can write 2^X to denote the power set of X.

Operations We write $x \leftarrow \chi$ to represent an element x being sampled from a distribution χ, and $x \xleftarrow{\$} X$ to represent an element being sampled uniformly at random from a set X. The output x of a probabilistic algorithm $\mathcal{A}$ is denoted $x \leftarrow \mathcal{A}$ and that of a deterministic algorithm $\mathcal{B}$ by $x := \mathcal{B}$

Data structures A *list* or *sequence* is an abstract data type that represents a finite number of ordered objects. The i^{th} element of a list v is written as v_i or $v[i]$ and its total number of elements is written as $\#v$. In the context of this thesis, a *string* is considered a binary sequence. If s is a string then $|s|$ denotes its bit length. A concatenation of n strings $s_1, \dots, s_n$ is written as $\langle s_1, \dots, s_n \rangle$, or $s_1 || s_2$ for $n = 2$. An *array* represents a collection of objects, where each element is accessible through its index. If A is an array then $\#A$ is its total number of cells, $A[i]$ is the value stored at index $i \in \{1, \dots, \#A\}$ and $A[i] := v$ stores the value v at index i in A. A *dictionary* (also known as map, lookup table or associative array) is a data structure that stores a collection of key-value pairs. If T is a dictionary, then $\#T$ denotes the number of pairs in T. If $(s, v) \in T$ is a pair, then the value v associated with search key s in T is written $T[s]$. $T[s] := v$ is the operation that stores v under search key s in T.

Functions Given two sets X and Y, a *function* $f \colon X \to Y$ assigns to each element $x \in X$ a unique element $f(x) \in Y$. Multivariate functions depend on several arguments, more formally, a function of n variables is a function $f \colon U \to Y$ where the domain $U \subseteq X_1 \times \cdots \times X_n$ is a set of n-tuples. While studying a multivariate function we might fix certain variables called *parameters* to highlight the "true variables". For example, to analyze a keyed hash function $h \colon X \times K \to Y$, we might fix $k \in K$ by defining the function $h_k(x) = h(x, k)$ for all $x \in X$. Furthermore, we can distinguish a function f from its value $f(x)$ by writing $f(\cdot)$. We define Func$[n, m]$ to be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^m$. A function $v \colon \mathbb{N} \to \mathbb{N}$ is called negligible in $k \in \mathbb{N}$ if for every positive polynomial $p(\cdot)$ and sufficiently large k, $v(k) < 1/p(k)$. We write $f(k) = \text{poly}(k)$ to specify the existence of a polynomial $p(\cdot)$ such that $f(k) \leq p(k)$ for sufficiently large k. Similarly, we write $f(k) = \text{negl}(k)$ to signify the existence of a negligible function $v(\cdot)$ such that $f(k) \leq v(k)$ for sufficiently large k.

Groups A *group* is a 2-tuple $(G, \cdot)$, where G is a set, $\cdot$ is a binary operation, and they satisfy the following four properties:

1. (Closure) For all $x, y \in G$, $x \cdot y \in G$.
2. (Identity) There exists an element in G, often denoted by 1 (or e), such that for all $x \in G$,
$$1 \cdot x = x \cdot 1 = x$$
3. (Inverse) For all $x \in G$, there exists an element in G called the inverse of x, often denoted by x^{-1}, such that,
$$x \cdot x^{-1} = x^{-1} \cdot x = 1$$
4. (Associativity) For all $x, y, z \in G$,
$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

A group is called an *abelian* group if it also is commutative; for all $x, y \in G$, $x \cdot y = y \cdot x$. A group is called *multiplicative*, with notation $(G, \cdot)$, if we tend to write its group operation in the same way as one does for multiplication, i.e.

$$x = y \cdot z \text{ and } x^5 = x \cdot x \cdot x \cdot x \cdot x.$$

A group is called *additive*, with notation $(G, +)$, if we tend to write its group operation in the same way as one does for addition.

$$x = y + z \text{ and } 5 \cdot x = x + x + x + x + x.$$

An abelian group is called *cyclic* if there exists an element $g \in G$ where for all $x \in G$ and $i \in \mathbb{N}$, $x = g^i$ or $x = i \cdot g$ for multiplicative or additive G respectively. If g is a generator of the cyclic group G we often write $G = \langle g \rangle$.

A subset H of G that satisfies the four group properties is called a *subgroup* of G, often denoted by $H \subseteq G$. Every group G contains two *trivial* subgroups, G itself and the group $\{1\}$ consisting of the identity element. A subgroup H of a group G that does not include the entire group G itself is called *proper*, denoted by $H \subset G$.

A.2. Definitions

Throughout the following definitions, $k \in \mathbb{N}$ will denote the security parameter and we will assume all algorithms to take it as input.

Definition A.1 (Pseudo-random Function (PRF)) *A function $f\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is pseudo-random if f_K is computable in polynomial time for any $K \in \{0,1\}^k$, and if for all polynomial size adversaries $\mathcal{A}$,*

$$\left| Pr\left[\mathcal{A}^{f_K(\cdot)} = 1 \colon K \xleftarrow{\$} \{0,1\}^k\right] - Pr\left[\mathcal{A}^{g(\cdot)} = 1 \colon g \xleftarrow{\$} \mathsf{Func}[n,m]\right] \right| \leq \mathsf{negl}(k)$$

where the probabilities are taken over the choice of K and g.

Definition A.2 (Pseudo-random Permutation (PRP)) *A function $f\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is a pseudo-random permutation if*

- *For any $K \in \{0,1\}^k$, f_K is one-to-one on $\{0,1\}^n$ to $\{0,1\}^n$.*
- *For any $K \in \{0,1\}^k$, $f_K(x)$ is computable in polynomial time for any $x \in \{0,1\}^n$.*
- *For all polynomial-size adversaries $\mathcal{A}$,*

$$\left| Pr\left[\mathcal{A}^{f_K(\cdot)} = 1 \colon K \xleftarrow{\$} \{0,1\}^k\right] - Pr\left[\mathcal{A}^{g(\cdot)} = 1 \colon g \xleftarrow{\$} \mathsf{Func}[n,n]\right] \right| \leq \mathsf{negl}(k)$$

where the probabilities are taken over the choice of K and g.

Definition A.3 (Symmetric Key Encryption (SKE)) *A symmetric encryption scheme is a tuple of three polynomial time algorithms* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

$K \leftarrow \mathsf{Gen}(1^k)$: *is a probabilistic algorithm that takes as input a security parameter* $k \in \mathbb{N}$ *and outputs a secret key* K*.*

$c \leftarrow \mathsf{Enc}(K, m)$: *is a probabilistic algorithm that takes as input a secret key* K *and a message* m *and returns a ciphertext* c*.*

$m := \mathsf{Dec}(K, c)$: *is a deterministic algorithm that takes as input a secret key* K *and a ciphertext* c *and returns a message* m*.*

A symmetric encryption scheme is correct if for all $k \in \mathbb{N}$*, for all keys* K *generated by* $\mathsf{Gen}(1^k)$*, and for all messages* $m \in \{0,1\}^*$*,*

$$\mathsf{Dec}_K(\mathsf{Enc}_K(m)) = m$$

Definition A.4 (IND-CPA Security) *Let* $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme and* $\mathcal{A}$ *be an adversary and consider the following indistinguishability experiment* $\mathbf{CPA}_{\mathsf{SKE},\mathcal{A}}(k)$*:*

1. *a key* $K \leftarrow \mathsf{Gen}(1^k)$ *is generated.*
2. $\mathcal{A}$ *is given access to oracle* $\mathsf{Enc}_K(\cdot)$*.*
3. $\mathcal{A}$ *outputs two plaintexts* m_0 *and* m_1*, where* $|m_0| = |m_1|$*.*
4. *a bit* $b \xleftarrow{\$} \{0,1\}$ *is chosen at random, and challenge ciphertext* $c \leftarrow \mathsf{Enc}_k(m_b)$ *is given to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs a bit* b' *after a polinomially bounded number of queries to* $\mathsf{Enc}_K(\cdot)$*.*
6. *if* $b = b'$*, the experiment returns 1 otherwise it returns 0.*

We say that SKE *is IND-CPA secure if for all polynomial-size adversaries* $\mathcal{A}$*,*

$$\left| Pr\left[\mathbf{CPA}_{\mathsf{SKE},\mathcal{A}}(k) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(k),$$

where the probabilities are taken over the random coins of Gen*,* Enc*, and* b*.*