University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Design and Evaluation of Biometric Data Management in Blockchain-Based Self-Sovereign Identities

Justin Verhoek
Zurich, Switzerland
Student ID: 18-750-661

Supervisor: Daria Schumm, Prof. Dr. Burkhard Stiller
Date of Submission: August 5, 2024

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

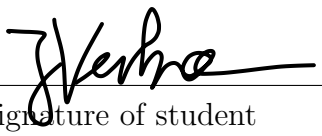
Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Generative AI was employed to create the example biometric fingerprint data (minutiae points). Additionally, a grammar correction AI program (Grammarly) was used to correct the spelling and sentence structure.

Zürich, 04.08.2024



Signature of student

Zusammenfassung

Die Nutzung digitaler Identitäten, basierend auf zentralisierten Identitätsmanagementsystemen, hat in letzter Zeit stark an Popularität gewonnen. Dies hat zu einem Anstieg der Forschung im Bereich digitaler Identitätsmanagementsysteme geführt. Ein neuer, potenzieller Ansatz ist die Nutzung von "Self-Sovereign Identity"(SSI)-Systemen, die eine dezentrale Datenverwaltung unterstützen. Aktuelle Ansätze für Identitätsmanagementsysteme basieren auf biometrischen Daten zur Authentifizierung der Nutzer. Die Integration biometrischer Daten in SSI-Systemen bereitet noch einige Probleme, insbesondere in Bezug auf Verknüpfbarkeit und Datenschutz.
Diese Arbeit schließt diese Forschungslücke, indem sie biometrische Daten privat und sicher in ein blockchain-basiertes SSI-System integriert. Das Hauptziel ist die Implementierung und Bewertung eines SSI-Systems, welches die biometrische Authentifizierung der digitalen Identitätsinhaber unterstützt, unter Berücksichtigung von Datenschutz und Sicherheit.
Das vorgeschlagene System nutzt hybride Verschlüsselung, um die Fingerabdruck-Minuzien-Daten sicher zu speichern, und wird auf einem Ethereum-basierten Framework eingesetzt. Das System basiert auf "Smart Contracts", "Decentralized Identities"(DIDs) und "Verifiable Credentials", um ein funktionierendes SSI-System aufzubauen. Der Authentifizierungsprozess der Verschlüsselung, Entschlüsselung und Vergleich der Fingerabdrücke wird "Off-Chain"durchgeführt, um die Leistung und Sicherheit zu optimieren.
Das Design und die Implementierung werden anhand von Schlüsselmerkmalen wie Leistung, Sicherheit, Datenschutz, Verknüpfbarkeit und Skalierbarkeit bewertet. Die Ergebnisse zeigen die Sicherheit und den Datenschutz des vorgeschlagenen Systems bei effizienter Leistung in Bezug auf die Authentifizierungszeit. Es wurden jedoch Herausforderungen wie hoher Gasverbrauch, der zu hohen Transaktionsgebühren führt, und Skalierbarkeitsbeschränkungen identifiziert. Die Arbeit schließt mit Empfehlungen für zukünftige Arbeiten, einschließlich der Optimierung des "Smart Contract"-Betirebs und der Erforschung alternativer Speicherlösungen zur Bewältigung der idtenifizeirten Herausforderungen.
Diese Arbeit trägt zur laufenden Entwicklung sicherer, benutzerzentrierter, dezentraler Identitätsmanagementlösungen bei, indem sie eine der ersten Implementierungen und Bewertungen eines SSI-Systems unter Verwendung biometrischer Authentifizierung bereitstellt. Dadurch wird weitere Forschung angeregt und das Potenzial der Integration biometrischer Daten in blockchain-basierte SSI-Systeme hervorgehoben.

Abstract

Digital identity usage relying on centralized identity management systems has recently seen a huge increase in popularity. This has triggered an increase in research surrounding digital identity management systems. A potential remedy is the utilization of Self-Sovereign Identity (SSI) systems, that support decentralized data management. Current identity management system approaches rely on biometric data for the authentication of users. However, there are still several challenges to overcome with the integration of biometric data into SSI systems, especially in terms of linkability and privacy concerns.
This thesis fills this research void, by integrating biometric data privately and securely into a blockchain-based SSI system. The main objective is to implement and assess an SSI system utilizing biometric authentication of the digital identity holders while taking privacy and security into account.
The proposed system utilizes hybrid encryption to securely store the fingerprint minutiae data and is deployed on an Ethereum-based framework. The system relies on smart contracts, decentralized identifiers (DIDs), and verifiable credentials to build a functioning SSI system. The authentication process of encryption, decryption, and matching of fingerprints are performed off-chain to optimize performance and security. The design and implementation are evaluated based on key aspects such as performance, security, privacy, linkability, and scalability.
The results indicate the security and privacy of the proposed system with efficient performance in terms of authentication time. However, challenges such as high gas usage, resulting in high transaction fees, and scalability limitations were identified. The work concludes with future work recommendations, including optimizing smart contract operation and exploration of alternative storage solutions to mitigate these challenges.
This thesis contributes to the ongoing development of secure, user-centric, decentralized identity management solutions, by providing one of the first implementations and evaluation of an SSI system utilizing biometric authentication. Thereby steering further research whilst highlighting the potential of integrating biometric data within blockchain-based SSI systems.

Acknowledgments

First and foremost, I would like to thank the Communication Systems Group and especially Prof. Burkhard Stiller and my supervisor Daria Schumm for the great opportunity. More concretely I want to extend my sincere appreciation to Daria Schumm for her invaluable guidance, suggestions, and assistance throughout the writing and creation process of my thesis.

Moreover, I would like to thank Amos Calamida, Robin Chan, and Sarina Grubenmann, for their support, constructive feedback, and meticulous proofreading.

Contents

Chapter 1

Introduction

1.1 Motivation

The increased use of digital identities in various government services shows the need for advancements in identity management systems. Self-Sovereign Identity (SSI) systems emerge as a promising solution, representing the transition from traditional centralized identity solutions toward decentralized self-sovereign data management. While conventional solutions, such as eId and ePassports, rely on biometric data for identity verification, the integration of biometric data into SSI introduces new challenges. An important issue identified in recent research, such as in the work of Goodell and Aste [14], is the inherent problem of linkability associated with the use of biometrics in SSI systems. Linkability expresses the interconnectedness of transactions by the same identity holder, creating a visible pattern and resulting in the formation of a permanent record. The linkability of transactions introduces a potential privacy concern, as it allows for the tracing and recording of an individual's activities over time. This prompts the research and evaluation of biometrics in SSI systems for identity verification. Addressing this issue requires careful consideration of the overall system design as well as privacy-enhancing techniques utilized within the system. Exploring solutions can help the evolution of SSI systems to potentially mitigate the challenges associated with biometrics and contribute to establishing a more resilient and privacy-centric digital identity landscape.

1.2 Thesis Goals

The primary objective of this thesis is to develop a solution utilizing biometric authentication in an SSI system, focusing on privacy and security concerns. To achieve this several intermediate goals have to be met.
Firstly, the current state of research on the use of biometrics within the SSI context will be explored through a literature review. Further, the linkability and privacy-preserving mechanism of current solutions are to be investigated. Out of that research, a problem statement to be investigated with the thesis has to be formulated. A privacy-preserving

mechanism has to be selected and the prototype architecture has to be designed. The prototype has to be implemented thereby creating a functional SSI system. Finally, the design and implementation must be evaluated based on security and privacy, scalability, performance, and costs.

1.3 Thesis Outline

This project encapsulates a literature review on the essential concept, design, and implementation of an SSI system utilizing biometrics for authentication, as well as an evaluation of that system. The aim is to develop a functional system and gain valuable insights from it to guide future research.
First, the fundamental concepts required for the project are introduced. Then the related work regarding the current state of research in SSI systems utilizing biometrics is displayed. Out of this related work, the thesis goal is formulated. For the fulfillment of the thesis goal, a system has to be implemented. In the Design chapter, the overall design of the system is explained. In the next chapter, the detailed implementation of that design is discussed. Afterward, the evaluation chapter evaluates the design and implementation. The thesis ends with final considerations regarding the work done and presents future work.

Chapter 2

Fundamentals

This chapter introduces the fundamental concepts that form the basis for understanding the research and implementation of Self-Sovereign Identity (SSI) systems utilizing biometric data. The background section describes the essential concepts whilst the related work section presents the current state of research on these concepts.

2.1 Background

2.1.1 Identity

Identity is a fundamental, complex concept studied and defined across various disciplines, including philosophy, psychology, law, sociology, and computer science [30]. In essence, identity is defined through the characteristics and attributes that uniquely distinguish one individual from another over a lifetime. These characteristics can be personal information such as name, date of birth, and social security number, but also behavioral and physical attributes such as biometric features [30, 31]. In everyday life, identity plays an important role, especially in the context of interactions with systems and authorities. Identity can facilitate interactions, and enable trust and clear communication. Traditional means of establishing identity rely on physical documents such as passports, driving licenses, and birth certificates. A bank can, for example, verify the identity of a customer by examining the authenticity of such a document. With the rise of digital technologies, it is increasingly important to extend the concept of identity to encompass digital identity [4, 31].

2.1.1.1 Digital Identity

Digital identity is the digital representation of an entity, meaning a person or organization. A digital identity includes all the information that uniquely identifies an entity and all the mechanisms used to manage, control, and authenticate the information in digital environments. A digital identity does not have to be the identity of a real person. It can digitally represent fictional people, legal people, companies, or organizations. Not every

person has a digital identity and it is possible for one person to have multiple digital identities [10, 31]. Digital identities are essential for enabling online transactions, accessing services, and interacting on online platforms. Online banking, email correspondence, and streaming services are all dependent on the use of digital identities. To use these services entities need to tie themselves to many unique digital identities, by using password/email combinations [4]. Managing all these different identities on different platforms is challenging for users as well as service providers. Users have to remember countless different password and username combinations, which can lead to security vulnerabilities if the same combination is used multiple times. Service providers face challenges in ensuring the security and privacy of that user data [4].

Digital identities have several essential components. Namely, *attributes* are pieces of information about a person's identity. These can include demographic information (e.g., age, gender, address), biometric data (e.g. fingerprints, iris scans), and behavioral patterns (e.g., browsing history, typing patterns). Some attributes are inherent to that person, while others have been assigned to that person by a third party (e.g., nationality) and could potentially be changed [10]. A key characteristic of attributes is that they are intended to classify an entity into a particular category. Two people can share the same age which is an attribute of theirs, therefore multiple entities can share the same attribute [10, 31].
Attributes that are able to point to a single digital identity record in a system are called *identifiers*. Identifiers are unique labels or codes assigned to entities in digital identity systems. As such, they are not intended to describe a person but rather used as a reference to the real-world identity. A third party often assigns these identifiers depending on the use case (e.g., the legal name of a person, social security number, or username). In other instances the identifier can be a representation of a property of an entity (e.g. fingerprints or other biometric attributes) [10, 30]. Not only can the use of an identifier make the real-world entity identifiable in a system, but the combination of attributes that are not identifiers can accomplish the same. By applying multiple attributes such as a person with green eyes, SSO starting with the number 5, 25 years old, and living in Switzerland an entity could be identifiable.

2.1.1.2 Identity Management Systems

An *identity management system* is a collection of tools, processes, and policies used to manage digital identities within an organization or across organizational boundaries [12]. Identity management systems provide several fundamental operations including *identification*, *verification*, *authentication*, and *authorization*:

1. **Identification** is the first step in the process of recognizing the user interacting with a resource. It takes place when an entity claims a digital identity. It is typically accomplished by providing a username, a unique ID, email, or some other unique identifier [26].

2. **Verification** is also known as identity proofing. It is the process of confirming the identifier is valid and is associated with an identity in the system [26].

3. **Authentication** is the process of verifying the identity of an entity before allowing access to a particular system's resources. This operation typically requires the entity to present a token, password, or biometrics before accessing the protected resource [26].

4. **Authorization** is the final operation granting access rights to an entity. This process determines what resources can be accessed by the entity [26].

The combination of these processes working in tangent ensures that only allowed entities have access to specific resources [26].

Despite the advantages offered by digital identity systems, there are many challenges. Firstly, identity formation is a continually ongoing process, where a person's identity is developed over the whole lifetime, evolving as a result of interactions with the environment surrounding the person. For that reason, every identity management system must be designed to support the flexible, resilient, and dynamic nature of human identity [30]. Further, many identity providers generate income by accumulating user data with limited transparency. This data can be used to develop user analysis systems and can be sold to advertisers. The users have no control over how the data collected from them is used and stored. Currently, most online service providers follow their own set of data management policies and practices with few restrictions which can lead to industry monopolies. These data ownership and governance issues have led to the pursuit of new identity models, processes, and regulations, to provide more control to the users and transparency over the utilization and storage of personal data by service providers [26].
In today's digital world, most data subjects are aware of the possibility of identity fraud, and still, weak passwords for online accounts are highly prevalent. This is propelled by the overwhelming amount of online services that need to be managed by a subject. Authentication through accounts from service providers such as Facebook or Google has surfaced as an option to alleviate the complexity of account management [4, 26]. This so-called OAuth solution allows users to create profiles on platforms through the use of a profile from third-party providers [18]. For example, instead of creating a new account, a user could use their Google account to create a new profile for themselves on the new platform. However, this OAuth solution for account and password management decreases the user's control over their data. Additionally, losing the account will lead to the lockout of all linked accounts. Finally, there is no visibility into the security practices of the companies managing the account and malicious actors could compromise one authentication service and gain access to all linked accounts [4, 26]. The most promising solution for the safe authentication of users is the usage of biometrics [26]. The main advantages of biometric authentication systems are that there is no need to carry tokes or remember passwords, they are harder to circumvent, and they provide a stronger link between the subject and the action [9].
Another challenge is that the users' data is generating another income stream for service providers in the digital economy which has led to the accumulation of data for single service providers and the fragmentation of identity data among different service providers. An average data subject has their data scattered on different platforms, with the privacy and security of the data being entirely dependent on individual business strategies, goals, and budget plans. The lack of universal standards and interoperability among platforms

has made it challenging for data subjects to store, obtain, remove, or share their personal data [26].
Multiple enormous data breaches in recent years have shown the weakness of current centralized data storage, used to store users' sensitive identity information. Outdated methods of authentication via physical identity documents introduce their own privacy and security issues due to their sensitivity, as they may be falsified, altered, lost, or stolen and are at the mercy of human error [26].

The above-described challenges have motivated research to explore novel concepts for digital identity management. The emergence of distributed ledger technologies and blockchain has given rise to Self-Sovereign Identity systems as a possible solution for the aforementioned challenges.

2.1.2 Self-Sovereign Identity

The most predominant approaches to secure identity management focus on central providers of identities, such as national authorities or online service providers [10]. *Self-Sovereign Identity* (SSI) is a novel decentralized identity concept that enables users to fully control and manage their digital identity. The identity holder controls their data and decides under what conditions and how their data should be shared with others. The SSI model empowers the identity holders by letting them selectively disclose identity information in different contexts. This enables the owner of their digital identity to only disclose the required identity information [26]. SSI is a novel concept that has grown from the emergence of blockchain technologies and has no universal consensus [2]. Allen [2] lays the groundwork for adopting SSI by establishing desired properties in a theoretical framework. Namely, they point out ten properties which self-sovereign identities should possess:

1. Existence: Identities existence must be independent of any organization or authority
2. Control: The usage and storage of identity information must be in control of the user
3. Access: Users must always have access to their identity data
4. Transparency: The system and its used algorithms must be transparent to the user
5. Persistence: Identities sustain existence over time as long as the users want
6. Portability: Identity information should be portable to any other service/platform
7. Interoperability: Identities should be usable across any services, platforms, and international boundaries.
8. Consent: Any usage of identity information must be agreed to by the user
9. Minimalization: When data needs to be shared, the minimum amount should be shared.
10. Protection: The user's rights must always be protected

Allen [2] defines SSI in the following way. The user must be the central element in the administration of identity. That entails the interoperability of a user's identity across multiple locations, as well as the user's complete control of that digital identity. To accomplish this, SSI must be transportable, meaning it cannot be locked down to one site. An SSI system fully fulfilling the ten properties proposed by Allen [2] enables a person to have full control over their digital identities regardless of living conditions, including where the person lives, the person's citizenship or chosen service providers for social networks. SSI increases individual freedom in the digital space and counteracts the oligopoly structure of the current internet, in which the "Big Five" (Apple, Microsoft, Google, Amazon, Facebook) manage most of the digital identities [10].

2.1.2.1 Components of SSI

SSI is built upon several key building blocks. The SSI concept was established after the rise of distributed ledger technologies. The key architectural requirements helping the realization of SSI are decentralized identifiers (DID) and verifiable credentials [26]. Blockchain technology, part of the distributed ledger technologies, provides a cryptographically secure, decentralized, and distributed database of information. The blockchain enables excluding central authorities and allows transactions to take place in a peer-to-peer network without the reliance on a central authority [26]. Blockchain systems consist of an immutable ledger, which consists of cryptographically hashed transactions. A group of transactions forms a *block*. Each block contains a header referencing the hash of the previous block. This creates a chain of blocks where each block references the block before it and gives the technology the name blockchain [26]. The transparency and immutability of the blockchain provides a framework of trust and the opportunity for data storage diffused among numerous entities. Leveraging blockchain technology has several advantages for SSI systems. The decentralized nature of the blockchain allows data to be stored in a ledger and not with a central entity. The immutability of the blockchain means transactions have to be appended to the blockchain and can be verified by all the members of the network. The distributed nature of blockchain complicates influencing the availability and integrity of the data [26].
Decentralized Identifier (DID) is a new type of identifier, developed by the World Wide Web Consortium (W3C) and a key component of the SSI model [29]. It was made for a verifiable, self-sovereign digital identity that is interoperable across a range of different systems. DIDs are URLs that lead to a DID Document, with information on how to use the particular DID. In the context of SSI, a DID Document can specify a verification method used for authentication [26].
A DID by itself can only be used for authentication. DIDs become particularly useful combined with *verifiable credentials* or claims, which is another W3C standard. Verifiable credentials can be used to make any number of attestations about a DID subject [11]. These attestations can include certifications, granting DID subjects specific access rights. A verifiable credential can, for example, attest that a bank has Know-Your-Costumer (KYC) approved an individual to open a bank account, that an individual has been certified as eligible to drive or authorized to access certain programs [3]. A certifiable credential contains the DID of its subject (bank customer), and the attestation (KYC approval) and must be signed by the individual or entity making the claim (bank). Verifiable claims are

methods for trusted authorities to issue certified credentials to a specific DID. These DID verifiable credentials are under the control of the DID subject and can be used at their will independently of the authority providing the credential [30].
DIDs are made to work with many different systems and therefore do not require any distributed ledger. Given the transparency and immutability of blockchain, personal information should never be stored on the blockchain itself [26, 30]. But blockchain technology can be used to track the access of personal data, that is stored off-chain. This makes it ideal for recording attestation or claims, and for the granting and revocation of access to personal data [30].

2.1.2.2 Challenges of SSI

SSI solves several of the challenges concerning digital identity systems mentioned in 2.1.1.2. SSI offers the user control over their identity information, enabling flexible usage of digital identities and therefore supporting the continual identity formation. The user's ability to share only selected information lessens the personal data gathered by service providers. The decentralized structure of SSI systems makes the personal information stored on digital identities much more secure, and the interoperability of the system enables platform-agnostic usage.

SSI offers many advantages over centralized identity management systems and addresses many of its problems. Still, as a novel concept, it has some challenges to overcome. Namely, gaining control over the digital identities comes with additional administrative efforts for the user. A core challenge is therefore ensuring solutions to help users deal with this, ensuring SSI is not only usable but also practical and sufficiently comfortable [10]. Since SSI is supposed to be portable and support interoperability with different systems standards for data management are crucial.
In contrast to traditional identity management models, where the identity providers are primarily responsible for cryptography key management and its corresponding risks, in SSI systems, this responsibility falls on the user. Helping users with key management through the right technical support is a fundamental step towards the mass adoption of SSI systems [26].
In the context of SSI systems relying on blockchain technology, privacy concerns around identity data and issues around data correlation are important topics of discussion. SSI systems deal with personal identity information, and blockchain is immutable and transparent. Therefore, policies and practices around data management, user experience, and data exchange should be carefully defined [26].
In an SSI system, a user can have multiple digital identities that they can use for multiple different platforms. If malicious users gain control over a user's profile, all of those digital identities and connected profiles on other platforms could be in danger. Ensuring only the rightful user has access to their digital identities becomes of the utmost importance [26, 30]. For this reason, choosing the safest authentication system is essential. One of the most promising and more widely available solutions is the use of biometrics for authentication [4].

2.1.3 Biometric Data

Biometric technologies have been employed for the automatic authentication and identification of individuals at an increasing rate, because of advancements in modern technologies making them more accessible and their advantages over traditional methods [27]. Biometrics are defined by the *International Standards Committee on Biometrics* as the automated recognition of individuals based on behavioral or biological characteristics [31]. Characteristics currently used for the recognition of individuals are finger and palm friction ridges, iris, face, voice, handwriting, hand shape, and hand vein patterns. These biological characteristics are influenced by the behavior of the individuals, as such no biometric characteristic is solely behavioral or biological, but always a combination of both. [31]. Recognition is the key word in the definition, for the reason that to be recognized, something has to be known beforehand. For biometric authentication to work, the biometric has to be learned in a registration phase and stored. Biometric technology can then classify whether it has encountered these biometrics before [31].
Biometric attributes have some significant differences from other forms of authentication [31]. Traditional authentication schemes primarily rely on some form of token or some other secret knowledge possessed by the user. These have several limitations, as they identify the individuals indirectly. These approaches cannot differentiate between the intended authorized user or a person having access to the password or token, whereas biometrics directly identify the user [24, 31]. A person can easily share a PIN code with another person to grant them access, but transferring biometrics is a whole other challenge. For identity systems non-transference of authorization makes them more secure, the utilization of biometrics is therefore ideal [31]. In traditional authentication approaches, non-biometric verifiers must match exactly. A PIN is not correct unless all digits match, and a password requires the user to enter an exact match [31]. Because of variations in human biology and physiological changes over time, biometric systems need to account for some variations in the measurement of users' biometrics. To have a working, biometric-based authentication system, it is crucial to have some adjustment and robustness to the variation of biometric data capture [27, 31].

A key advantage of biometrics as described above is the distinctive identification of the user. At the same time, this is a great concern of biometric data, as they are universally distinctive enough to identify persons. Resultingly, the data is extremely sensitive. Further, individuals cannot choose their biometric data and cannot change them in the future [14]. Tokens, PINs, or passwords can be revoked or reissued by system administrators. To make systems more secure, a policy might require the change of PIN every 30 days. This is not possible with biometrics as they are fixed to the user and cannot be changed [24, 31]. Tokens, PINs, and passwords can be application-specific and one can have different ones for each application in use. This can ensure that if one digital identity associated with a password is compromised, others would still be safe. Biometric attributes are fixed to the user, one cannot have different ones for each application. The security of biometric data is therefore essential, as compromised biometric data in one system makes every other application secured with the same biometric vulnerable [24, 31]. For these reasons, biometric data is considered highly sensitive data and needs careful privacy and security considerations [14]. Therefore, the protection of biometric information is an essential component of creating an identity management system.

Over the last two decades, several different *biometric template protection* (BTP) methods have been developed [13]. To protect the privacy of individuals, the ISO/IEC International Standard 24745 [16] has defined four criteria for each biometric template protection scheme. First, the protection should not greatly degrade the performance of the biometric authenticating system (*performance preservation*). Second, it should be impossible to reconstruct the original template from the protected template (*irreversibility*). Third, if a protected template is compromised it should be possible to revoke the protected template and create a new protected template (*revocability*). Fourth, linking biometric templates across different applications or databases should not be possible (*linkability*).
Linkability of biometric data can be defined as follows: two templates are fully linkable if there is a method to detect they were extracted from the same biometric template [13]. The linkability of biometric data is a serious privacy and security threat. It enables the collection of the personal data of individuals by malicious, unauthorized groups or users. This personal data can be linked and analyzed with data from other applications, leading to a loss of control for the user. This analysis of user data can lead to the identification of the user. Biometric identification is therefore much more susceptible to the risks of identity theft and fraud [13, 23]. The problems raised by linkability show, why it is essential to address the linkability of biometric data in identity management systems.

2.1.4 Biometric Data in Self-Sovereign Identity Systems

SSI systems are based on distributed ledgers allowing participants to share control of the system. They also provide a shared public view of transactions ensuring every participant sees the same transaction history. This shared transaction history enables the decentralized safety of the systems but makes the protection of personal information shared in the transaction history essential. The information in the shared history enables malicious users to link transactions of the same participant taking these users' control over information away. The key aspect of SSI is that users can fully control and manage their digital identity. Users can choose what identity information (name, age, social security number) they want to share and with whom they want to share it [7]. This selective disclosure of information claims provides privacy preservation. If service providers can link claims to the same owner across multiple consecutive presentations or through collusion with other service providers the owner's right to privacy is destroyed [17]. Not only is linkability a privacy and security concern, but it also breaks four of the desired properties of Allen [2] namely control, consent, minimalization, and protection. As such, linkability can lead to the loss of self-sovereignty of users in an SSI system as they lose control over their information, the basic concept of SSI.

The binding of identification with a biometric characteristic that users cannot change implicitly prevents users from utilizing the system without introducing linkability [14]. The linkability of the biometric data compromises this digital identity but could endanger any digital identity registered with the same biometric information in the past or the future. Linkability of biometric data in SSI systems suffers from the same issues as normal identity systems but additionally threats to the fundamental properties upon which SSI is built. Designing an SSI system utilizing biometrics needs to integrate protection schemes against linkability.

2.1.5 Smart Contract

The term smart contract refers to the code scripts that execute certain tasks synchronously on multiple nodes of a decentralized distributed ledger (blockchain). Smart contracts are executed in a network of nodes without the need for an external authority [34]. For that reason, the code of a smart contract can facilitate a transaction or contract between two parties without reliance on a trusted third party. Smart contracts still have some limitations in terms of the lack of general-purpose libraries, limited support for debugging, and issues concerning the performance [34]. As smart contracts are deployed on blockchains they need to be executed and validated by each node making all transactions visible to the blockchain network. As a result, the transaction costs especially for those with complex computation can be high and the validation of transactions may take a long time [34].

The Ethereum blockchain has become one of the most popular blockchains for smart contracts. The Ethereum Virtual Machine (EVM) allows the execution of scripts using an international network of public nodes. Ethereum has its own internal pricing mechanism for all network transactions. Gas is the measure of how much computing power a transaction would need. Users have to pay gas fees in Ethers (ETH) for each transaction they make and if transactions run out of gas they fail [6].

2.1.6 Hybrid Encryption

Data encryption is a technology that allows two parties, a sender and a receiver, to communicate while the information shared between them is transformed according to agreed-upon rules [33]. According to certain rules, the message is transferred between the plaintext and the ciphertext. The process of transforming the plaintext into ciphertext is called *encryption*. Similarly, the process of converting the ciphertext into plaintext is called *decryption*. There are various encryption algorithms that can be split into two main categories: *symmetric* encryption algorithms and *asymmetric* encryption algorithms [33].

Symmetric encryption refers to an encryption algorithm that uses the same key for both encryption and decryption. It can therefore be called private key encryption. In the symmetric encryption process, the data sender possesses the plaintext and the encryption key to generate the ciphertext. To be able to read the original text the recipient of the ciphertext has to perform the decryption with the same encryption key. The result should be the original readable plaintext. In this system both the sender and receiver necessitate the key. It requires the receiving party to be in possession of the secret key in advance [33].

Asymmetric encryption algorithms use a pair of keys, a public key and a private key. The private key is kept safe with the receiver party and cannot be shared with anyone, while the corresponding public key can be shared with anyone requesting it. A message sender can use the public key to encrypt the plaintext. The receiver can decrypt the ciphertext with the private key [33].

Both approaches have advantages and disadvantages. Symmetric encryption algorithms are very efficient and diversely usable. It requires less computational power and has

a relatively higher speed of encryption and decryption while processing. This makes symmetric encryption more suitable for large data volumes. Yet, symmetric encryption also has severe drawbacks. Due to the key being used for both encryption and decryption, the security cannot be guaranteed. Once the secret key is intercepted and known by an attacker the communication process is unsafe. Additionally, the management of secret keys can be troublesome. For each new transmission, a new key is needed and these keys need to be shared safely. This key management is a burden for users [33].
The asymmetric encryption approach is a very secure method for data encryption and decryption. The key management system is much simpler as a user only has to keep his own private key secret and safe. But compared to symmetric encryption the asymmetric encryption algorithm is comparably slow. Further, some algorithms only allow a certain length of plaintext to be encrypted. It is therefore only suitable for encrypting small amounts of data [33].

The hybrid encryption approach aims to combine the advantages of both algorithms. In the hybrid encryption approach, the message plaintext is encrypted with the secret symmetric key. The public key of the receiver is used to encrypt the symmetric key. Both the message ciphertext and the encrypted symmetric key are sent to the receiver. The receiver uses their private key to decrypt the encrypted symmetric key. With the decrypted symmetric key the receiver can decrypt the ciphertext to get the original plaintext. This hybrid scheme leverages the high speed of symmetric encryption and the high security of asymmetric encryption [33].

2.2 Related Work

2.2.1 Biometric Template Protection in SSI systems

With the emergence of decentralized technologies and the security advantages they provide, the concept of self-sovereign identity systems has gained momentum. The use of biometrics for authentication and security has risen enormously in recent years, due to the availability of biometric sensors in smartphones. Because of the sensitive nature of biometric data, protection of the data is a crucial issue. Biometric template protection refers to safeguarding individuals' biometric data, such as fingerprints or facial recognition patterns, against unauthorized access and misuse. If malicious actors gain access to the biometric template all services secured by the template are breached and because the biometric can never be changed, all future services with the same biometric template will be vulnerable. Making sure the original biometric template is secured is essential for every identity management framework based on biometrics [9, 31]. Because of the novelty of self-sovereign identity systems, there are few works on the combination of biometric template protection and SSI.

Current identification and identity-providing systems rely on centralized databases which offer a single point of compromise, which is a direct threat to users' digital identities. The Horcrux protocol proposed by Othman and Callahan [22], provides a method for the secure exchange of biometric credentials within the IEEE 2410-2017 Biometric Open Protocol

Standard (BOPS) implemented across self-sovereign identity platforms using DIDs and DID documents. This decentralized approach enhances the security of digital identities. However, the protocol was never implemented, and as such the performance, correctness, security, and privacy of the protocol have not yet been concretely investigated.
Bathen et al. [4] propose a novel approach called Self-Sovereign Biometric IDs (SelfIs), which combines concepts such as decentralization, cancelable biometrics, bloom filters, and machine learning to create a solution for using biometric data. SelfIs gives control over digital identity back to the user. The biometric template is transformed with a one-way function, which in theory is irreversible, meaning one cannot get the original biometric template once it has been transformed. If an attacker can steal the transformed biometric data he or she cannot carry out replay attacks. The transformation of the raw biometric template creates pseudo-biometrics, which enhances the security and privacy of the raw biometric template.
Hamer et al. [15] propose the Unique Self-sovereign Identity (USI), combining Cancelable Biometrics with W3C Verifiable Claims [29]. The approach provides privacy-preserving and non-linkable identification, guaranteeing no double enrolments. The system only allows one identity per person as it is intended for the services of organizations such as governments or healthcare.
Bathen et al. [4] initially proposed the use of pseudo-biometrics in blockchain, but Mishra et al. [21] raises concerns over the architectural design of the solution, as well as the use of bloom transformation, which can suffer from invertibility and linkability attacks. Because of that and the sensitive nature of biometric credentials, Mishra et al. [21] develops a novel framework to build self-sovereign and revocable pseudo-biometric identities. The approach utilizes the combination of Random Distance Method (RDM) cancelable transform, Decentralized Identifiers (DID), and BOPS server to create a secure environment for identity management. The convergence of biometric authentication with SSI systems marks a shift towards decentralized technology, granting users greater control over their digital identities. Several protocols to ensure the security and safety of the biometric data in SSI systems have been proposed, but all of them lack concrete performance and linkability analysis to guide further research in the right direction.

Table 2.1: Summary of papers discussing privacy solutions for biometrics in SSI systems

Paper Reference	Issue	Solution	Future Work
Othman and Callahan [22]	• Traditional authentication and identity-providing systems are a security risk • Identity information stored in a centralized structure	• Decentralized authentication and storage based on DIDs and BOPS. • Users have full control over their digital identities enabling SSI.	• Reference Implementation of the protocol • Analysis of performance

Continued on next page

Table 2.1 – continued from previous page

Paper Reference	Issue	Solution	Future Work
Bathen et al. [4]	• Use of biometrics has seen a huge increase, because of easier availability • Once an attacker has a biometric template replay attacks can be carried out	• SelfIs based on decentralization, cancelable biometrics, bloom filters and machine learning • Privacy-first solution for using biometrics in SSI systems	• Low accuracy with live data shows the model needs to be further developed for real use
Hamer et al. [15]	• Identifications systems allowing at most one identity exchange of verifiable documents is necessary. • The same basic information gathered for identification in every systems makes persons linkable across different systems. • Individuals are no longer in control over how their identity is used.	• Unique Self-Sovereign Identity (USI) based on Canelable Biometrics and Verifiable Claims. • Utilization of biometrics allows users to register without official documentation. • Achieve privacy preserving an non linkable identification, assuring non double enrolment.	• Improve the performance of biometric identification • Reference implementation • Security, linkability analysis
Mishra et al. [21]	• Biometrics in combination with SSI has huge potential but risks biometric templates. • Until now only Bathen et al. [4] propose use of pseudo-biometrics. The approach has drawbacks in terms of accuracy and submitting the pseudo-biometric to a third party.	• Random Distance Method and DIDs are used to create pseudo-biometric identities. • Solution Ensures safety whilst having good performance accuracy.	• No further work mentioned. • No security analysis performed

2.2.2 Biometric Template Protection in Blockchain

Biometric template protection is not only critical for SSI systems but also for other blockchain-based identity management systems. While SSI systems promise the users full control over their digital identities, blockchain-based systems in general leverage decentralized technology for data security and to some extent user sovereignty. Whilst these systems do not follow all the SSI principles to the fullest, they promote the sovereignty of the user and discuss the importance of robust safety protocols for biometric data.

Liu, Sun, and Schuckers [19] designed and built a new identity management framework that integrates a user's transformed biometric data to smart contracts through the Ethereum blockchain platform. The framework addresses the problems of traditional, centralized, biometric-based identity management systems such as data security and the users' loss of control. The approach enables data security by utilizing the distributed

public ledger, guarantees privacy by using private storage for sensitive biometric data, and facilitates self-sovereignty by using smart contracts.
The combination of blockchain and biometrics offers tremendous upside as both technologies could improve each other, while the challenge of biometric template storage and protection remains. Delgado-Mohatar et al. [9] analyzes the latency, processing time, economic cost, and biometric performance of integrating biometric data on the Ethereum blockchain platform using smart contracts. The experimental study shows that template protection can preserve the privacy of the templates and even improve biometric matching accuracy concerning unprotected templates.
Smart contracts add functionality to blockchain platforms but can suffer from double-spending attacks. Bisogni et al. [5] propose an encryption scheme using facial biometrics to authorize and sign transactions in smart contracts. This ensures malicious users remain punished and cannot continually violate the laws of the network. In the approach, the face is used as a biometric key, encoded with the Convolutional Neural Network (CNN), fused with an RSA key by using the Hybrid Information Fusion algorithm (BNIF), leading to a combined key.
Biometric authentication is the preferred authentication scheme in modern systems because of its usability. Such systems require the careful handling of biometric templates. Toutara and Spathoulas [28] propose a distributed scheme where the centralized database is replaced by a combination of Ethereum/IPFS in which the user's biometric templates are stored in transformed and encrypted form. This approach allows any third-party service to securely authenticate users without exposing the biometric template.
Acquah et al. [1] further propose an approach featuring the combination of Ethereum with the IPFS. In this method fingerprint templates are encrypted by the symmetric key algorithm: Advanced Encryption Standard (AES) and uploaded to the IPFS, while only the template hash is stored on the Ethereum network. The structure was proven to be feasible in terms of economic, performance, and security viewpoints.
Dalal et al. [8] proposes an approach for the verification of identity and educational certificates of students using biometrics and blockchain. In the proposed solution students submit a hash of their biometric and a unique phrase, this hash as well as the college certificate will be stored on the blockchain. Other college authorities can fetch a student's previous certificates using the hash of biometrics as well as the unique phrase. The project aims to help authorities with the efficient authentication of identity and past educational certificates.
Sharma, Saini, and Chaudhury [25] propose a multimodal biometric authentication scheme using decentralized fuzzy vaults relying on Blockchain technology. The combination of these technologies ensures the privacy of the user's biometric template. A security analysis of the scheme validated that the biometric template fulfills irreversibility, unlinkability, and revocability criteria.
Researchers have explored various methods to integrate biometric data into blockchain-based identity systems, addressing issues of data security, user control, and privacy. This investigation encompasses an in-depth analysis of performance, security, and privacy aspects, evaluating economic costs, processing time, and latency associated with storing biometric data on blockchain platforms using smart contracts.

Table 2.2: Summary of papers discussing applications of biometrics in the blockchain

Paper Reference	Issue	Solution	Future Work
Liu, Sun, and Schuckers [19]	• Traditional biometric-based identity management systems store biometric data on centralized databases. • Users have no control over how their data is used. • Structure a risk for the sensitive data in terms of security and privacy.	• Integrate users' transformed biometric data to smart contracts on the Ethereum blockchain. • Privacy is preserved through personal storage of biometrics on the IPFS • Flexible programming of smart contracts enables self-sovereignty	• Implementation of DIDs to enhance the framework's capabilities.
Bisogni et al. [5]	• Smart contracts need to be secure, fast and more user-friendly to be adapted in the future. • Preventing double-spending and ensuring authenticity in blockchain transactions is a challenge.	• Incorporate face biometrics and RSA encryption into Smart Contract transactions to enhance security and transparency.	• Integrate other biometric attributes. • Improve user experience during template creation.
Toutara and Spathoulas [28]	• Security and privacy vulnerabilities associated with the centralized storage of biometric templates in traditional authentication systems.	• Centralized database replaced by a combination of Ethereum/IPFS • Biometric templates are stored in transformed and homomorphically encrypted form. • Allows secure authentication without exposing actual biometric data	• Analyze the security parameters and costs of the protocol • Potentially refine the protocol
Acquah et al. [1]	• Traditional fingerprint recognition systems possess security vulnerabilities because of their centralized structure. • Centralized databases are controlled by a central authority	• Encrypted fingerprint templates are stored on the Ethereum blockchain in conjunction with the IPFS. • Templates are hashed and stored on the IPFS, only the hashes are stored on the Ethereum blockchain.	• Derive a robust hashing function for different types of blockchains.

Continued on next page

Table 2.2 – continued from previous page

Paper Reference	Issue	Solution	Future Work
Dalal et al. [8]	• Current methods of verifying student identities and educational certificates possess inefficiencies and vulnerabilities. • They rely on manual verification processes and centralized databases prone to security breaches.	• Blockchain-based systems where students register with personal data and biometric information • Hashes of certification issued by colleges, along with students' biometrics are stored on the blockchain/IPFS.	• Implementation of the verification systems using biometrics and the Ethereum blockchain. • Analysis of the implemented system.
Sharma, Saini, and Chaudhury [25]	• Protect users' biometric templates from unauthorized access and ensure the reliability of biometric authentication systems against different types of attacks • Attacks include reverse-engineer encrypted templates, link protected templates to individuals, brute force and zero-effort attacks.	• Multimodal biometric authentication scheme leveraging fuzzy vault to securely store biometric templates • Framework was analyzed in terms of security, irreversibility, unlinkability and revocability and validated in all those aspects.	• Address scalability challenges by exploring implementation on blockchain platforms such as Hyperledger. • Integrate the framework with SSI systems and digital wallets.

2.2.3 Summary of Related Work

Goodell and Aste [14] raise concerns about potential linkability associated with the use of biometric data in SSI systems, leading to security and privacy issues. Some research has addressed enhancing the security of biometric data in SSI systems, by incorporating biometric template protection. A significant gap remains in the analysis of the security and practical feasibility of these proposed systems. Research focusing on biometric template protection in blockchain has presented several approaches while also analyzing the solutions. This shows the current lack of research analyzing the security-enhancing solutions of biometric data in SSI systems. The goal of this thesis is to present a solution that reduces the linkability of biometric data in an SSI system and perform an analysis in terms of performance, scalability, security, and privacy aspects.

Chapter 3

Design

The related works have shown that biometrics are only included in SSI systems as a form of authentication. This chapter proposes the design of a system making use of biometrics for the authentication of the DID holder, by presenting the system on various levels of abstraction.
In the proposed system a user submits their biometric when creating a DID. The biometric information is encrypted and stored on the DID Doc. An issuer can add a credential to the DID. If the holder of the DID wants to present this credential to a verifier they need to authenticate themselves via biometrics. The submitted biometric is compared to the stored one. If the records match, the user will be identified as the original holder of the DID and can present the credential to the verifier.

A real-world example use case in which the system can be used is the following: A person (user/holder) creates a DID and registers it with his biometrics. The person has just finished their degree at a university. The university (issuer) issues a credential to the person by issuing the credential to the DID. While applying for jobs potential employers (verifier) want to verify the degree. The person can show proof of the degree by presenting the credential issued to their DID. To be able to present the credential the person must first authenticate themselves by submitting their biometrics. If the authentication confirms the identity, the person will be able to present the credential to the potential employer who in turn can verify the credential.

3.1 System Overview

Figure 3.1 shows an overview of the enrollment phase of the proposed system. A user registers the biometrics transforming them into biometric data, that is subsequently encrypted. With this encrypted information the DID is created and stored on the blockchain. An issuer is then able to issue a credential to the DID, which is also stored on the blockchain.

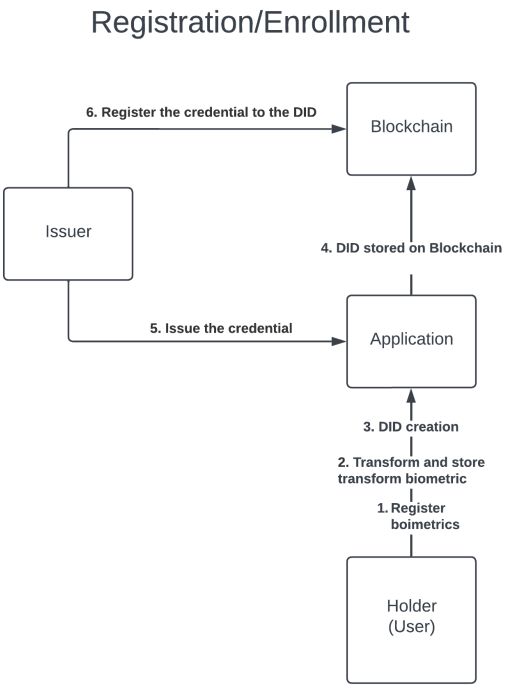


Figure 3.1: System overview of enrollment phase

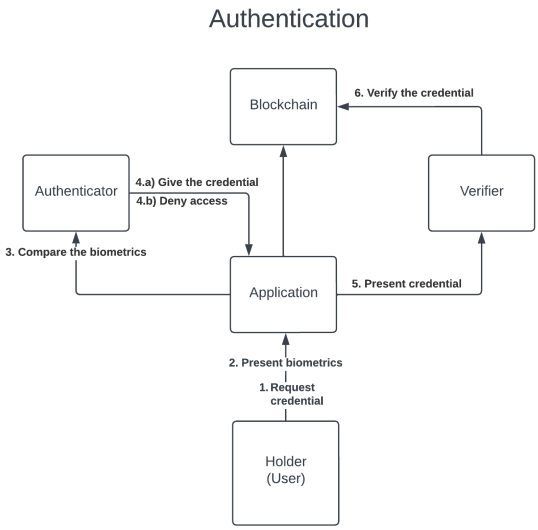


Figure 3.2: System overview of authentication phase

Figure 3.2 shows an overview of the authentication process in the proposed system. The user wants to present a credential, in order to do so the user needs to pass authentication to ensure the user is the individual who created the DID in the first place. The user therefore requests the credential and presents the biometrics. An authenticator compares the stored biometrics with the newly presented biometrics and if they match grants the user the credentials. If the biometrics do not match the user is denied access to the credentials

and cannot present them to a verifier. The user can then present the credentials to a verifier, who can then verify them.

In Figure 3.3 the detailed sequence of the enrollment phase of the proposed system is depicted. The user submits a fingerprint and the minutiae points are generated (fmp1). The fmp1 data is then encrypted with a symmetric key. Following that the encrypted fmp1 data is split into two parts share 1/2 and share 2/2. The symmetric key is encrypted with a public key of the authenticator in the system. Share 1/2 and the symmetric key are stored locally on the device. A DID is then registered to the user with the share 2/2 being saved on the DID Doc.
An issuer is able to generate a credential for any given DID. If a user earns or deserves a certain credential the issuer can create the credential and save it to the DID of the user. In the end, the user has the share 1/2, the DID ID, and the symmetric key. The enrollment phase consists of creating a DID, enabling biometric authentication, for the user, and adding credentials from an issuer to the DID.

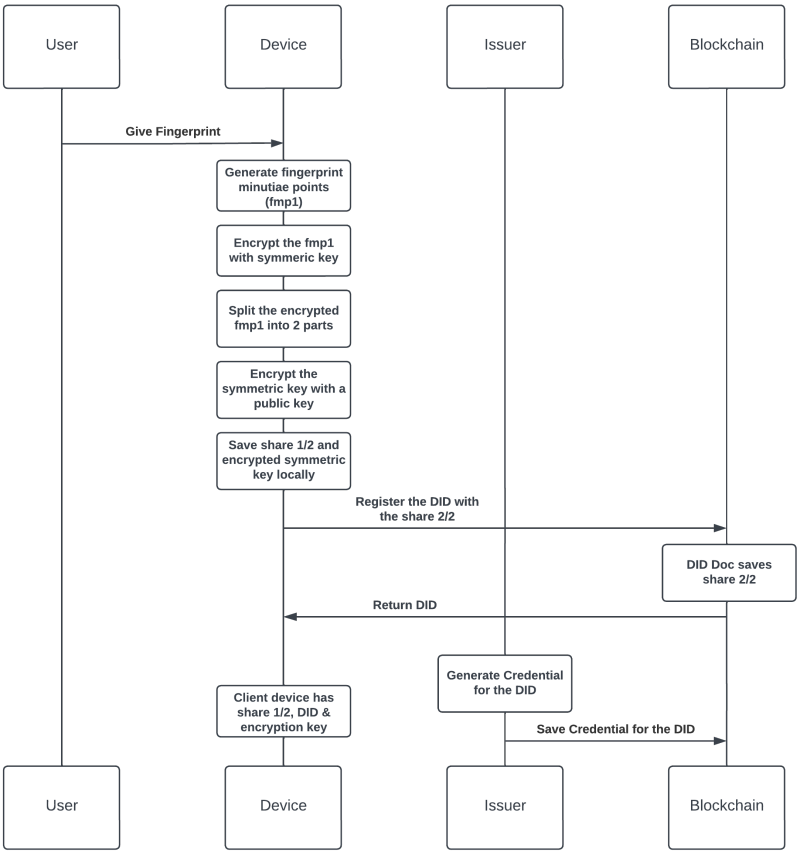


Figure 3.3: Sequence diagram of the enrollment phase of the proposed system

Figure 3.4 shows the detailed sequence of the authentication phase of the proposed system. If a user wants to present the credentials, the user first needs to receive them. A user has to retrieve the DID, the associated share 1/2, and the encrypted symmetric key. The user has to present a fingerprint from which the minutiae points are generated (fmp2). The fmp2 data is encrypted using the same symmetric key as in the enrollment phase. The user

requests the credential and submits the share 1/2, the encrypted symmetric key, and the encrypted fmp2. An authenticator receives a request for the authentication containing this information. The authenticator requests the info from the DID to receive the share 2/2. The authenticator decrypts the encrypted symmetric key with his private key. Once the authenticator has both shares 1/2 and 2/2, they are combined back together, forming the original encrypted fmp1. Afterward, the authenticator uses the symmetric key to decrypt the encrypted fmp1 and fmp2 data to obtain the decrypted fingerprint minutiae data. The two sets of minutiae points fmp1 and fmp2 are then compared and the authenticator checks whether they match. If they match the user gets the credential and is able to present it. If they do not match the user was not successfully authenticated and the access to the credential is denied.

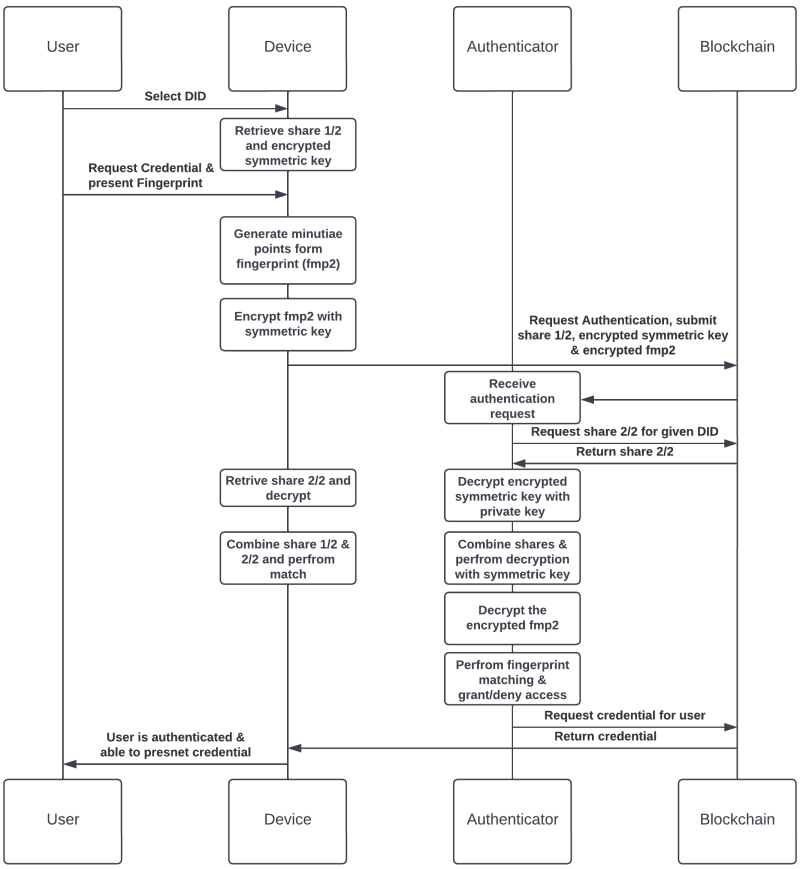


Figure 3.4: Sequence diagram of the authentication phase of the proposed system

Chapter 4

Implementation

This chapter discusses the implementation of the design of the system presented in the previous chapter 3. The implementation of the design was built on a preexisting codebase provided by CredChain Project[1]. The codebase provided the base structure for the implementation to be built upon. This code was changed and extended to implement biometric authentication in an SSI system[2]. The system's smart contracts are coded with Solidity whilst the rest of the code is written with JavaScript.
Deploying and managing the Ethereum smart contracts is done using the Hardhat development environment. Additionally, Node.js is used to handle server-side operations and integrate various functionalities within the project. To help with encryption and Ethereum smart contract interactions, the implementation leverages the web3.js libraries[3] and the Node.js crypto module[4].

Overall, the system implementation is as follows, the user creates a DID and saves part of the encrypted biometric data on that DID. The issuer can add a credential, which is created with the issuer's public key (address). If the user wants to present that credential they need to be authenticated. If the authentication is successful the user is granted the credential and is able to present it to a verifier. This verifier can check the credentials by obtaining the issuer's public key from the credential signature. If this public key matches the expected issuer's public key the credential is validated. This implementation ensures the incorporation of biometric authentication into a basic SSI system.

4.1 Smart Contracts

The smart contracts form the foundation of the SSI system. They manage DIDs, credential issuance, and authentication transactions. In this section, the essential smart contract files and their interactions are discussed.

[1] `https://github.com/schummd/credchain`
[2] `https://github.com/jverho/SSI-Biometric-Authentication`
[3] `https://web3js.readthedocs.io/en/v1.2.11/index.html`
[4] `https://nodejs.org/api/crypto.html`

4.1.1 DID Registry

The Listing 4.1 shows the DID Registry smart contract written with Solidity that handles the DIDs and their basic functionalities. After the smart contract is deployed, the DID Registry allows the creation of DIDs. This is done with the `register()` function that is shown in Listing 4.1. The function requires the Ethereum address of the user, the DID method, and the encrypted biometric information as inputs. It then adds all the information to a DID Doc, which in this simplified system is a struct saved on the same smart contract. The register function ensures that each Ethereum address can only have one DID. Once the register function is used the DID is created for the address the user specified. The `getInfo()` function can be used to get the stored additional info. It is used by the Credential Registry contract, discussed in 4.1.3 when the user requests authentication.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.9;
3
4  contract DID {
5
6      struct DDO {                                // DID Document
7          address id;                             // ethereum address of
               identity owner
8          address owner;                          // original identity owner
9
10         string did;                             // did method
11
12         string additionalInfo;                  // field to store
               information for authentication
13     }
14
15     mapping (address => DDO) private identity;  // DID has a single DDO
16     mapping (address => bool) registered;
17
18     modifier onlyOwner (address _id) { require(msg.sender == identity[
           _id].owner); _; }
19
20     function register(address _id, string memory _did, string memory
           _additionalInfo) public {
21         require(registered[_id] == false, "the DID document already
               exists");
22         registered[_id] = true;
23         identity[_id].id = _id;
24         identity[_id].owner = msg.sender;
25         identity[_id].did = _did;
26         identity[_id].additionalInfo = _additionalInfo; // Store the
               additional information
27     }
28
29     function getInfo(address _id) public view returns(string memory) {
30         return  identity[_id].additionalInfo;
31     }
32 }
```

Listing 4.1: DID Registry smart contract

4.1.2 Issuer Registry

The Issuer Registry contract shown in Listing 4.2 is used to maintain a registry of issuers. These issuers give DID holders specific credentials. Issuers can be added and deleted to the registry with the appropriate functions. The `checkIssuer()` function allows checking whether the address submitted as an input is the address of an issuer in the registry.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.9;
3
4  contract IssuerRegistry {
5
6      mapping(address => bool) public registry;
7
8      function addIssuer(address _address) public {
9          registry[_address] = true;
10     }
11
12     function deleteIssuer(address _address) public {
13         delete registry[_address];
14     }
15
16     function checkIssuer(address _address) external view returns(bool) {
17         if (registry[_address]) { return true; }
18         return false;
19     }
20 }
```

Listing 4.2: Issuer Registry smart contract

4.1.3 Credential Registry

The Credential Registry is the smart contract that controls the credentials that can be added to DIDs. The credential is a verifiable claim that a verifier can verify by checking the signature. The code in Listing 4.3 shows the Credential Registry contract. It requires the import of the DID Registry contract because the `getInfo()` function from the DID Registry contract is used, in the function `requestAuthentication()` (cf. Listing 4.4). The DID Registry is saved as a state variable of type 'DID' representing a reference to an instance of the 'DID' contract. The Credential Registry can use this reference to interact with the 'DID' contract and call functions from it.

The two events `CredentialIssued` and `AuthenticationRequest` are declared when the contract is deployed. The events are used and emitted in the `requestAuthentication()` (Listing 4.4) and `handleAuthenticationResult()` (cf. Listing 4.5) functions.

The struct Credential is the part of the code that saves the credential information ensuring each credential has a unique ID. The `addCredential()` function in Listing 4.3 enables adding credential information to a given DID. The function only saves all the information for the credential. The credential itself is created in the JavaScript file `credential.js` with the function `generateCredential()` (Listing 4.6), which is discussed in 4.2.1.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.9;
3
4  import "./DIDRegistry.sol";
5
6  contract Credentials {
7      DID private didRegistry;
8
9      event CredentialIssued(bool result, address indexed user, string
           issuer, string holder, string credHash, string signature);
10     event AuthenticationRequest(address indexed user, string _credId,
           string submittedInfo, string storedInfo, string localInfo, string
            key);
11
12     constructor(address _didRegistry) {
13         didRegistry = DID(_didRegistry);
14     }
15
16     struct Credential {             // verifiable claim
17         string id;
18         string issuer;
19         string holder;
20         string credHash;
21         string signature;           // issuer signature of the
               credential
22         bool uploaded;              // true: the ID is used and
               information uploaded
23     }
24
25     mapping (string => Credential) private credential;
26
27     function addCredential(string memory _id, string memory _issuer,
           string memory _holder, string memory _credHash, string memory
           _signature) public {
28         require (credential[_id].uploaded == false, "credential already
               exists");
29         credential[_id].id = _id;
30         credential[_id].issuer = _issuer;
31         credential[_id].holder = _holder;
32         credential[_id].credHash = _credHash;
33         credential[_id].signature = _signature;
34         credential[_id].uploaded = true;
35     }
36 }
```

Listing 4.3: Part of the Credential Registry smart contract

The `requestCredential()` function in Listing 4.4 is called by a user wanting to authenticate themselves. It takes the `userID` and the `credentialID` as inputs, as well as the `submittedInfo` which is the encrypted authentication biometric (encrypted fmp2), the localInfo (share 1/2), and the encrypted symmetric key. The explanation of what is encrypted and how can be found in 4.2.2. To get the encrypted biometric information (share 2/2) that is stored on the DID Doc the `getInfo()` function is used. All this information is used and given as inputs to the `AuthenticationRequest` event that is emitted. This

event is caught by the authenticator which uses the information passed in the event to carry out the authentication. The authenticator is discussed in 4.2.4.

```
1 function requestCredential(address _id, string memory _credId, string
    memory _submittedInfo, string memory localInfo, string memory key)
    public {
2         string memory storedAdditionalInfo = didRegistry.getInfo(msg.
              sender);
3         emit AuthenticationRequest(_id, _credId, _submittedInfo,
              storedAdditionalInfo, localInfo, key);
4     }
```

Listing 4.4: Request credential funciton from the Credential Registry smart contract

The `handleAuthenticationResult()` function is called by the authenticator to emit the result of the authentication. If the authentication was successful and the fingerprints matched, the function will emit the `CredentialIssued` event, with the result `true`, the credential information containing the issuer address, the holder address, the credential hash and the signature of the credential, that is created in the `generateCredential()` function (Listing 4.6). If the fingerprints do not match the authenticator emits the event with the result `false` and no additional information.

```
1 function handleAuthenticationResult(address _id, string memory _credId,
    bool _result) public {
2         if (_result) {
3             emit CredentialIssued(_result, _id, credential[_credId].
                  issuer , credential[_credId].holder, credential[_credId].
                  credHash, credential[_credId].signature);
4         }
5         else{
6             emit CredentialIssued(false, _id, "-", "-", "-", "-");
7         }
8     }
```

Listing 4.5: Handle authentication result function from the Credential Registry smart contract

4.2 JavaScript Modules

The JavaScript modules facilitate the creation and verification of credentials, manage the hybrid encryption and decryption processes, and handle the authentication events. This section discusses the key JavaScript files, their functionalities, and interactions within the system.

4.2.1 Credential

The `credential.js` file handles the creation and verification of the credentials. To generate a credential the `generateCredential()` function, shown in Listing 4.6, is used.

This function relies on the web3.js ethereum library. The `credentialID` is created by utilizing the `web3.utils.sha3()` function which creates a hash of the input string using the Secure Hash Algorithm 3 (SHA-3). The credential consists of several pieces of information: `credentialID`, `holderAccount`, `issuerAccount`, `creation date`, and the `claim`. The `web3.utils.sha3()` is used again, this time to create a hash of the credential. This hash is taken together with the private key of the issuer as inputs in the `web3.eth.accounts.sign()` function to create a signature of the credential. The function uses the private key of the issuer to create the signature. The `generateCredential()` function returns the credential, credential hash, and the signature in string format. With this function, credentials are created for a DID account. The return values are used in the `addCredential()` function, (cf. Listing 4.3) to add the credential to the registry.

```
1  async function generateCredential(holderInfo, holderAccount,
       issuerAccount, issuerPrivateKey) {
2
3      let now = new Date();
4      // holder info should be different to make sure hash is different
           for each credential
5      let credentialID = web3.utils.sha3(issuerAccount+now+holderInfo);
6
7      // create the credential
8      var credential = {
9          "id": credentialID,
10         "holder": holderAccount,
11         "issuer": issuerAccount,
12         "created": now.toLocaleDateString(),
13         "claim": holderInfo,
14     };
15
16     // create the has and signature, convert signature to string for
           storage
17     let credentialHash = web3.utils.sha3(JSON.stringify(credential));
18     let sig = await web3.eth.accounts.sign(credentialHash,
           issuerPrivateKey);
19     let signature = JSON.stringify(sig);
20
21     return [ credential, credentialHash, signature ];
22 }
```

Listing 4.6: Generate credential function from the credential.js file

In a system with verifiers and issuers, the public address acting as the public key of the issuer would be known. Any verifier could therefore acquire the address of the issuer. The `verifyCredential()` function (cf. Listing 4.7) from the `credential.js` file handles the verification of a credential by verifying the signature of the credential. To achieve this the address of the issuer is recovered from the credential hash and the signature with the `web3.eth.accounts.recover()` function. This function returns the corresponding public key of the private key with which the signature was created. This public key is the address of the issuer and can be compared to the address of the issuer that is publicly known. If they match the credential is verified, if they do not match the credential is refuted.

```
async function verifySignature(credHash, signature,
    expectedIssuerAddress) {
    // Recover the signer's address from the signature and the credHash
    let recoveredAddress = web3.eth.accounts.recover(credHash, signature
        );

    // Compare the recovered address with the expected issuer address
    if (recoveredAddress.toLowerCase() === expectedIssuerAddress.
        toLowerCase()) {
        console.log('Signature is valid and credential is verified.');
    } else {
        console.log('Signature is invalid or does not match the expected
            issuer address.');
    }
}
```

Listing 4.7: Verification of the credential function from the credential.js file

4.2.2 Encryption

The `encryption.js` file handles all the encryption and decryption processes within the system. The system relies on a hybrid approach to encryption, meaning the biometric data is encrypted with a symmetric key, and the symmetric key itself is then encrypted with an asymmetric key pair. One part of the code shown in Listing 4.8 is responsible for symmetric encryption and decryption of the biometric information. The other part of the file as seen in Listing 4.9 handles the encryption and decryption of the symmetric key. Both parts rely on the crypto library to carry out the encryption and decryption.

The `generateSymmetricKey()` function (Listing 4.8) creates the symmetric key that is used to encrypt and decrypt the biometric information. The function returns a string 256-bit key in hex string format that is created with the `crypto.randomBytes()` function. This random key is specified to be a 256-bit key because Advanced Encryption Standard 256 (AES-256) is employed. The `enrcypt()` function (Listing 4.8) takes data as input in string format and the symmetric key. The Initialization Vector (IV) is created to ensure that the same data encrypted with the same key does not produce the same encrypted ciphertext. The `crypto.createCipheriv()` function creates the cipher object with which the data can be encrypted. Within the function the encryption algorithm is specified to be 'aes-256-cbc', the key is converted into a buffer and the IV is added. The cipher is then used to encrypt the data creating the ciphertext. The IV is then prepended to the ciphertext to ensure the IV can easily be used in the decryption process. Including the IV in the result does not make the system less secure. It is simply used to ensure that the same input data with the same key does not lead to the same encrypted ciphertext. Adding the IV to the ciphertext is a standard practice to simplify the decryption process.

Decryption of the data is accomplished with the `decrypt()` function shown in Listing 4.8. It takes the encrypted data and the symmetric key as inputs. The IV is split from the encrypted data. A decipher object is created with the `crypto.createDecipheriv()` function. The decipher object is defined to use the same algorithm 'aes-256-cbc' used

in the encryption process. To create the decipher object the function requires the same symmetric key and IV. With the decipher object the encrypted data can be decrypted. The `decrypt()` function returns the decrypted data in string format.

```
1  // Function to generate a symmetric key
2  function generateSymmetricKey() {
3      return crypto.randomBytes(32).toString('hex'); // 256-bit key in hex
           string format
4  }
5
6  // Function to encrypt data with symmetric key
7  function encrypt(data, secretKey) {
8      const iv = crypto.randomBytes(16);
9      const cipher = crypto.createCipheriv('aes-256-cbc', Buffer.from(
           secretKey, 'hex'), iv);
10     let encrypted = cipher.update(data);
11     encrypted = Buffer.concat([encrypted, cipher.final()]);
12     return iv.toString('hex') + ':' + encrypted.toString('hex');
13 }
14
15 // Function to decrypt data with symmetric key
16 function decrypt(data, secretKey) {
17     let parts = data.split(':');
18     let iv = Buffer.from(parts.shift(), 'hex');
19     let encryptedText = Buffer.from(parts.join(':'), 'hex');
20     let decipher = crypto.createDecipheriv('aes-256-cbc', Buffer.from(
           secretKey, 'hex'), iv);
21     let decrypted = decipher.update(encryptedText);
22     decrypted = Buffer.concat([decrypted, decipher.final()]);
23     return decrypted.toString();
24 }
```

Listing 4.8: Functions for the symmetric AES encryption/decryption of the biometric data

The code in Listing 4.9 creates public/private RSA key pair and uses them for encryption and decryption. The public and private keys are generated with the `crypto.generateKeyPairSync()` and specified to be RSA keys. The keys are saved to PEM files. The keys are only generated once, as the symmetric key should be encrypted with the same public key so that the authenticator can decrypt the symmetric key with the same corresponding private key.

The `encryptSymmetricKeyWithPublicKey()` function in Listing 4.9 takes the symmetric key as the input. The `crypto.publicEncrypt()` function is used to encrypt the symmetric key. It uses the public key previously specified to encrypt the symmetric key. The padding and oaepHash are specifically defined to ensure they match the corresponding padding and oaepHash in the decryption function. In the end, the function returns the encrypted symmetric key in string format.

The `decryptSymmetricKeyWithPrivateKey()` in Listing 4.9 function takes the encrypted symmetric key as the input. The `crypto.privateDecrypt()` function handles the decryption of the encrypted symmetric key. It is specified to use the private key corresponding to the public key, as well as the same padding and oaepHash as in the encryption function.

The `decryptSymmetricKeyWithPrivateKey()` returns the decrypted symmetric key in string format.

```
// Generate an RSA key pair (only once, and store them in respective
    files)
const { publicKey, privateKey } = crypto.generateKeyPairSync('rsa', {
    modulusLength: 2048,
    publicKeyEncoding: {
        type: 'spki',
        format: 'pem'
    },
    privateKeyEncoding: {
        type: 'pkcs8',
        format: 'pem'
    }
});
const PUBLIC_KEY = fs.readFileSync("utilities/rsa_public_key.pem");
const PRIVATE_KEY = fs.readFileSync("utilities/rsa_private_key.pem");

// Function to encrypt a symmetric key with the public key
function encryptSymmetricKeyWithPublicKey(symmetricKey) {
    const buffer = Buffer.from(symmetricKey, 'hex');
    const encryptedKey = crypto.publicEncrypt(
        {
            key: PUBLIC_KEY,
            padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
            oaepHash: 'sha256'
        },
        buffer
    );
    return encryptedKey.toString('base64');
}

// Function to decrypt a symmetric key with the private key
function decryptSymmetricKeyWithPrivateKey(encryptedKey) {
    const buffer = Buffer.from(encryptedKey, 'base64');
    const decryptedKey = crypto.privateDecrypt(
        {
            key: PRIVATE_KEY,
            padding: crypto.constants.RSA_PKCS1_OAEP_PADDING,
            oaepHash: 'sha256'
        },
        buffer
    );
    return decryptedKey.toString('hex');
}
```

Listing 4.9: Functions for the asymmetric encryption/decryption with the RSA public/private keypair

4.2.3 Matcher

The `matcher.js` file shown in Listing 4.10 handles the matching of fingerprints. The fingerprints are matched by comparing the minutiae points of the fingerprints. In order

for two minutiae points to be considered a match three conditions must be verified.

1. **Type**: The minutiae points must be of the same type. When minutiae points are extracted from fingerprints, each point is given a certain type. These types must be a match.

2. **Orientation**: The orientation of the minutiae points must match. Each minutiae point has an orientation in degrees when extracted from a fingerprint.

3. **Distance**: The distance between the two points is compared as the final condition. To achieve this, the Euclidean distance between the minutiae points is calculated and compared.

The `euclideanDistance()` function is used to calculate the distance between two points. The Euclidean Distance is the shortest distance between two points meaning the distance of the line segment between the two points. The distance is calculated using the formula shown in Equation 4.1, the euclideanDistance() function implements this formula.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{4.1}$$

The `matchFingerprints()` function in Listing 4.10 handles the comparison and matching of fingerprints. The two fingerprints to be compared are taken as the input. The threshold variable is 5 by default but can be changed. Each of the three conditions described above is verified for each minutiae point. If at any point in the matching a condition is not met the next point is taken to be compared.

1. **Type**: First the type of the two points is compared. This is checked by the condition `storedPoint.type === newPoint.type`. Checking the type of the point is done first because it requires the least computational power, making the matching process more efficient and faster.

2. **Orientation**: The next condition to be verified is the orientation. For the orientation to match the difference in orientation between the two points must be within 10 degrees. This is achieved with the condition `Math.abs(storedPoint.orientation - newPoint.orientation) <= 10`. This condition is checked before the distance because checking the orientation condition requires significantly less computational effort than the distance comparison.

3. **Distance**: The final condition to be checked is the distance between the two points. To achieve this, the Euclidean distance between the minutiae points is calculated with the `euclideanDistance()` helper function and compared. For the verification of the distance condition, the distance between the two points must not exceed the distance defined as the threshold (5 by default). This is accomplished with `euclideanDistance(storedPoint, newPoint) <= threshold`. This condition is checked last because it requires the helper function to calculate the distance and afterward compare it to the threshold requiring the most computational power of

the three conditions. Examining this condition last ensures that the task requiring the most computational power is only done when the two previous conditions are satisfied.

If for two points all conditions are met the variable `matches` is increased by one. This way the variable keeps count of all the successful matches. In the end, a threshold in terms of how many minutiae points should match is defined. In this case, the threshold is 70% meaning that at least 70% of the points should be a match. With the condition `matches >= matchThreshold` it is checked if the amount of matches is enough for the threshold. Depending on this result the `matchFingerprints()` returns `true` for a match and `false` for a mismatch.

```
1  // Helper function to calculate Euclidean distance between two points
2  function euclideanDistance(point1, point2) {
3      return Math.sqrt(Math.pow(point1.x - point2.x, 2) + Math.pow(point1.
           y - point2.y, 2));
4  }
5
6  function matchFingerprints(stored, newFingerprint, threshold = 5) {
7      let matches = 0;
8
9      for (let i = 0; i < stored.minutiae_points.length; i++) {
10         for (let j = 0; j < newFingerprint.minutiae_points.length; j++)
               {
11             const storedPoint = stored.minutiae_points[i];
12             const newPoint = newFingerprint.minutiae_points[j];
13
14             if (storedPoint.type === newPoint.type && Math.abs(
                   storedPoint.orientation - newPoint.orientation) <= 10) {
15                 const distance = euclideanDistance(storedPoint, newPoint
                       );
16                 if (distance <= threshold) {
17                     matches++;
18                     break;  // Move to the next stored point after a
                           match
19                 }
20             }
21         }
22     }
23
24     // Define a threshold for a match (currently 70% of points have to
           match)
25     const matchThreshold = 0.7 * stored.minutiae_points.length;
26     return matches >= matchThreshold;
27 }
```

Listing 4.10: Functions from the matcher.js file, that handle the matching of fingerprints

4.2.4 Authenticator

The `authenticator.js` file shown in Listing 4.11 creates a Node.js WebSocket that listens to specific events emitted by a smart contract on the Ethereum blockchain using Web3.js.

The task of the authenticator is to receive an authentication request event and carry out the authentication. This authenticator utilizes previously discussed functions from the `encryption.js` file discussed in 4.2.2 and the `matcher.js` file discussed in 4.2.3.

In the file, a new Web3 instance using a WebSocket provider to connect to the local Ethereum node running at 'ws://127.0.0.1:8545' is created. As the authenticator should listen to events released by the Credential Registry smart contract, the contract's address is given to the event listener. The `startListener()` function is an asynchronous function that sets up the WebSocket listener.
The `CredentialRegistry.events.AuthenticationRequest()` function listens for the `AuthenticationRequest` event emitted by the Credential Registry. If the listener catches such an event the event handler function is executed. The handler function receives and processes the event data. The `AuthenticationRequest` event defined in the Credential Registry contract (cf. Listing 4.4) sends out the user address, the credential ID, the stored encrypted info, the locally saved info and the new submitted encrypted info and the encrypted symmetric key. The event handler receives all this information and uses it for the authentication. First, the encrypted symmetric key is decrypted with the `decryptSymmetricKeyWithPrivateKey()` function from the `encryption.js` file (cf. Listing 4.9). In the next step, the original encrypted biometric that was split up is concatenated. Afterward, the symmetric key is used to decrypt the two encrypted fingerprint minutiae datasets. This is done using the `decrypt()` function from the `encryption.js` file (cf. Listing 4.8). The decrypted biometric data is used as inputs in the `matchFingerprints()` function from the `matcher.js` file, shown in Listing 4.10. The return of this function will either be `true` or `false`, which represents the result of the authentication.
The event handling ends with sending out the result of the authentication. This is achieved by calling the `handleAuthenticationResult()` function from the Credential Registry contract (cf. Listing 4.5). The function takes the user address, the credential ID and the authentication result as inputs. In this way, the process of requesting a credential, performing authentication, and sending out the authentication result is managed using smart contracts. This ensures that the whole process is saved on the blockchain. This has several advantages as tamper-proof records are created providing transparency in all authentication transactions.

```
1  const Web3 = require('web3');
2  const web3 = new Web3('ws://127.0.0.1:8545'); // Use IPv4 explicitly
3  const contractABI = require('../artifacts/contracts/CredentialRegistry.
       sol/Credentials.json').abi;
4  const contractAddress = '0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0'; //
       Credential Registry address
5  const CredentialRegistry = new web3.eth.Contract(contractABI,
       contractAddress);
6  const { matchFingerprints } = require('./matcher');
7  const { decrypt, decryptSymmetricKeyWithPrivateKey } = require('./
       encryption');
8
9  async function startListener() {
10     try {
11         const accounts = await web3.eth.getAccounts();
12         const credentialRegistryAccount = accounts[0];
13
14         // Listen for the AuthenticationRequest event
```

```
15          CredentialRegistry.events.AuthenticationRequest({
16              fromBlock: 0
17          }, async function(error, event) {
18              if (error) {
19                  console.error(error);
20                  return;
21              }
22
23              // getting the return values from the event
24              const { user, _credId, submittedInfo, storedInfo, localInfo,
                    key } = event.returnValues;
25
26              // decrypting the symmetric key
27              const decryptedKey = decryptSymmetricKeyWithPrivateKey(key);
28
29              // putting the encrypted string back together and decrypting
                    the encrypted information
30              const combinedInfo = localInfo + storedInfo;
31              const decryptedCombinedInfo = decrypt(combinedInfo,
                    decryptedKey);
32              const decryptedSubmittedInfo = decrypt(submittedInfo,
                    decryptedKey);
33
34              // Parse the JSON strings into JavaScript objects
35              const submittedFingerprint = JSON.parse(
                    decryptedSubmittedInfo);
36              const storedFingerprint = JSON.parse(decryptedCombinedInfo);
37
38              // Perform the matching off-chain
39              const result = matchFingerprints(storedFingerprint,
                    submittedFingerprint);
40
41              // Sending the authentication result back
42              CredentialRegistry.methods.handleAuthenticationResult(user,
                    _credId, result)
43                  .send({ from: credentialRegistryAccount })
44                  .on('receipt', function(receipt) {
45                      console.log(`Matching Result for ${user}: ${result
                            }`);
46                  })
47                  .on('error', console.error);
48          });
49
50          console.log('Authenticator started, waiting for events...');
51      } catch (error) {
52          console.error('Error starting listener:', error);
53      }
54  }
55  startListener();
```

Listing 4.11: Authenticator file

4.2.5 Client Listener

The `clientListener.js` file shown in Listing 4.12 is similar to the `authenticator.js` file in the sense that both listen to events being emitted by the Credential Registry smart contract. The WebSocket and event listener setup is the same for both files. The difference is the client listener is tasked with receiving the credential. Because of that the the event that is being listened to is the `CredentialIssued` event. The listener contains a filter in the form of a specific user address. As the file simulates the user's client device and is only interested in credentials issued to that user. The event handler function receives the data from the event emitted namely the user address, issuer address, holder address, the credential hash, and the signature. The credential can be checked by verifying the signature of the credential. This is accomplished with the `verifySignature()` function from the `credential.js` file (Listing 4.7).

```
1  const Web3 = require('web3');
2  const web3 = new Web3('ws://127.0.0.1:8545'); // Use IPv4 explicitly
3  const contractABI = require('../artifacts/contracts/CredentialRegistry.
       sol/Credentials.json').abi;
4  const contractAddress = '0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0';
5  const Credentials = new web3.eth.Contract(contractABI, contractAddress);
6  const { verifySignature } = require('./credential');
7
8  async function listenForCredentials(userAddress) {
9      // Listen for the credential issuance event
10     Credentials.events.CredentialIssued({
11         filter: { user: userAddress },
12         fromBlock: 'latest'
13     }, function(error, event) {
14         if (error) {
15             console.error(error);
16             return;
17         }
18
19         // getting the return values from the event
20         const { result, user, issuer, holder, credHash, signature } =
               event.returnValues;
21
22         // if the result is true receive the credential and verify it
23         if (result) {
24             let sig = JSON.parse(signature);
25             console.log(`Credential for ${user}:`);
26             console.log(`Issuer: ${issuer}`);
27             console.log(`Holder: ${holder}`);
28             console.log(`Credential Hash: ${credHash}`);
29             console.log(`Signature: ${JSON.stringify(sig)}`);
30             // credential is received by the client device and could be
                   given to verifier
31
32             // verifier verifies the credential
33             verifySignature(credHash, sig.signature, issuer);
34         }
35         else{
36             console.log("Authentication did not work! Biometrics did not
                   match!")
```

```
37          }
38      });
39      console.log("Client Receiver started and waiting for events...");
40  }
41
42  listenForCredentials('0x70997970C51812dc3A010C7d01b50e0d17dc79C8'); //
        user address
```

Listing 4.12: Client listener file

Chapter 5

Evaluation

In this chapter the implementation of the system presented in the previous chapter 4 is evaluated. The system is assessed regarding various key metrics such as performance, security, privacy, and scalability. Sections 5.1–5.4 individually focus on these different aspects to provide a comprehensive analysis of the system's strengths and areas for improvement. Additionally, the system is compared to the solution of Acquah et al. [1] in terms of these key metrics.

5.1 Performance

Proper logging was added to the respective deploy authentication files to evaluate the system performance regarding the time and cost of performing authentication. The performance evaluation was conducted with two sets of artificial example fingerprints containing 30 and 80 minutiae points. This was done because typical data from fingerprints ranges between 10-200 minutiae points, with good quality data requiring at least 40 to 100 minutiae [20, 32]. The two datasets were selected to evaluate the influence of different sizes of fingerprint information submitted to the system.

5.1.1 Gas usage

The gas usage for each transaction is stable, the gas cost prices in terms of ETH, CHF, and USD are highly volatile and can change rapidly. These prices are calculated at the time of writing (July 2024) where 1 ETH = 2846,38 USD = 2558,30 CHF. The first step in the process analyzed is the registering of a DID with the `register()` function from the DID Registry smart contract. As seen in Table 5.1 the gas usage is very high with a value of 1'406'862 for 30 minutiae points and 3'426'131 for 80 minutiae points. This gas usage currently translates to a price of 5,63 USD and 13,71 USD respectively.

The gas usage and gas cost of adding a credential to a DID with the `addCredential()` function can be seen in Table 5.2. The gas usage value is 731'856, meaning adding a

register()	30 minutiae points	80 minutiae points
Gas usage	1406862	3426131
Cost in ETH	$1,98 \times 10^{-3}$	$4,82 \times 10^{-3}$
Cost in CHF	5,06	12,31
Cost in USD	5,63	13,71

Table 5.1: Gas usage and costs for registration of a DID

credential would currently cost 2,93 USD. This gas usage is not influenced by the different sizes of fingerprint information sets as they are not used in this transaction.

addCredential()	
Gas usage	731856
Cost in ETH	$1,03 \times 10^{-3}$
Cost in CHF	2,63
Cost in USD	2,93

Table 5.2: Gas usage and costs for adding a credential

Table 5.3 shows the gas usage and cost of performing the authentication of a user with 30 minutiae points, while Table 5.4 shows the result of using 80 minutiae points. This includes the `authenticationRequest()` and `handleAuthenticatioResult()` functions. Only the `authenticationRequest()` function is influenced by the different sizes of fingerprint information. The authentication process uses gas in the amount of 438'841 for fingerprints with 30 minutiae points, which corresponds to a value of 1,56 USD. Performing the authentication with fingerprint data containing 80 minutiae points uses gas in the amount of 1'155'398 corresponding to a current price of 4,46 USD.

	authenticationRequest()	handleAuthenticationResult()	Total
Gas usage	94330	344511	438841
Cost in ETH	$1,21 \times 10^{-4}$	$4,28 \times 10^{-4}$	$5,49 \times 10^{-4}$
Cost in CHF	0,31	1,10	1,41
Cost in USD	0,34	1,22	1,56

Table 5.3: Gas usage and costs for authentication transactions with 30 minutiae points

	authenticationRequest()	handleAuthenticationResult()	Total
Gas usage	810887	344511	1155398
Cost in ETH	$1,14 \times 10^{-3}$	$4,28 \times 10^{-4}$	$1,57 \times 10^{-3}$
Cost in CHF	2,91	1,10	4,01
Cost in USD	3,24	1,22	4,46

Table 5.4: Gas usage and costs for authentication transactions with 80 minutiae points

The gas and cost analysis shows that the current system is extremely expensive. All these costs presumably have to be paid for by the user, the identity holder. The creation of a DID and adding a credential to that DID would cost the user, depending on the size of biometric data, 8,56 USD or 16,64 USD. To add to that the user would have to pay

1,56 USD or 4,46 USD to authenticate themselves and receive the credential. These are extremely high fees that the users would have to pay.
Furthermore, the analysis shows that the operations that do not occur often, the creation of DID and the adding of credentials, are the more expensive ones. The authentication of the user which presumably would happen much more often in the system is the less expensive one. As a result, the correct part of the system's functionalities are the more and less expansive ones. Still, the system as a whole is way too expensive and in its current form would never be adopted in a real-world scenario.

5.1.2 Time

To evaluate how much time the authentication process takes the time between sending out the authentication request and receiving the authentication result is measured. The time duration of the authentication is again analyzed using artificial fingerprint data containing 30 and 80 minutiae points respectively.

Table 5.5 shows the single measurements of the authentication time as well as the calculated average in milliseconds. It shows that the whole authentication process takes about 187,0 milliseconds on average.

measurements	authentication time [ms]
1	169,00
2	229,00
3	171,00
4	162,00
5	228,00
6	169,00
7	148,00
8	167,00
9	179,00
10	248,00
Average	187,00

Table 5.5: Time duration of the authentication process using 30 minutiae points

Table 5.6 shows the measurements and calculated average of the authentication time utilizing datasets of 80 minutiae points. The measurements show that the authentication takes 355,5 milliseconds on average.

The measurements of the authentication time showed that the smaller the size of fingerprint data files the faster the authentication process is. This comes from a minimal increase in time needed for the off-chain comparison of two fingerprints. But mainly stems from the fact that each step in the authentication process, such as sending and receiving, takes more time when more data is involved. The time analysis in general shows that the system performs efficiently for both 30 minutiae and 80 minutiae points, ensuring quick authentication times.

measurements	authentication time [ms]
1	352,00
2	352,00
3	337,00
4	355,00
5	344,00
6	355,00
7	367,00
8	384,00
9	364,00
10	343,00
Average	355,30

Table 5.6: Time duration of the authentication process using 80 minutiae points

This suggests the system is well-optimized for handling different sizes of biometric data. A user would never have to wait more than half a second to receive the authentication result, which is a satisfactory result. The fast authentication time enhances the user experience by minimizing the wait time and ensuring smooth operation.

5.2 Security

The security of the systems relies on the use of smart contracts, cryptographic techniques, and a well-structured authentication process. This section evaluates the security of the system by discussing the various parts of the system, responsible for creating a safe system. To achieve this different aspects of the system such as smart contracts, data encryption, the authentication process, and the overall security of the system are assessed.

5.2.1 Smart Contracts for Security

The system utilizes Ethereum smart contracts to manage the DIDs, credential issuance as well as the transactions needed for the authentication. Smart contracts increase the security of the system by providing a transparent and resistant mechanism for transaction processes. The key security advantages of using smart contracts are:

1. **Transparency**: All transactions are recorded on the blockchain, creating a transparent record that can be seen and examined by any participant in the network. Anyone can look for inconsistencies in the process of requesting credentials and performing the authentication. This makes faking one part of the process useless because the whole process is recorded on the blockchain.

2. **Immutability**: Once a smart contract is deployed it cannot be changed or altered. This ensures the way the different participants of the system communicate with each

other is fixed. This results in the logic of the whole system remaining consistent and secure.

3. **Decentralization**: Using the decentralized Ethereum blockchain network eliminates the single-point failure problem. There is no use in attacking a single node or participant in the system. This makes the system much more secure and resilient to attacks.

4. **Automation**: The automated execution of contracts according to predefined rules, reduces the risk of human error thereby enhancing the system's reliability.

5.2.2 Hybrid Encryption of Biometric Data

The system utilizes a hybrid encryption approach to secure biometric data. The combination of symmetric and asymmetric encryption ensures the security of sensitive biometric data. A detailed description of the system's encryption implementation is provided in section 4.2.2.

The symmetric encryption of the biometric data is done using the Advanced Encryption Standard (AES) algorithm. Specifically, the AES-256 algorithm is employed, as the key used for the encryption is 256 bits. In the process, a randomly generated Initialization Vector (IV) is used to ensure the ciphertext is always unique, even for identical plaintext input. This makes the system more secure against certain types of attacks such as replay attacks. For every DID in the system a new key is created to encrypt the biometric data, further increasing the security. The AES-256 encryption is uncrackable with current technologies. AES-256 ensures the safe and efficient encrypting of the biometric data.

To enable asymmetric encryption a RSA (Rivest-Shamir-Adleman) key pair is created. This key pair is generated once, the authenticator in the system has the private key and any user wanting to authenticate themselves can use the respective public key. The symmetric key is encrypted using RSA-2048 with the authenticator's public key. RSA-2048 encryption is uncrackable with current technologies, ensuring that only the authenticator with their private key can perform the decryption.

The encryption and decryption are performed off-chain, locally on the user's device. This is done for performance reasons concerning the computational power of smart contracts. Additionally, anyone could see how the encryption/decryption is done if it would be performed by a smart contract, keeping the logic off-chain protects the process further.

The hybrid encryption approach ensures that the encrypted biometric data can only be decrypted by the authenticator. As long as the private key of the authenticator remains safe the encrypted biometric data is completely safe as the combination of both encryption are currently not decryptable without the corresponding keys.

5.2.3 Authentication Process

Biometric authentication has several advantages compared to traditional authentication schemes as discussed in section 2.1.3. The implementation of biometric authentication in the system leverages all these advantages making the system more secure. In the specific use case the biometric authentication guarantees that only the original holder of the DID can present credentials to potential verifiers. This is done through several key components of the system. Hybrid encryption guarantees the security of the encrypted biometric data. The biometric data is only stored in encrypted form never risking the original biometric template. The matching of biometrics is performed off-chain ensuring raw biometric data is never exposed during the authentication process. Secure storage of the authenticator's private key ensures only the authenticator is able to decrypt the biometric data.

5.3 Privacy

Privacy is a critical aspect of the system, especially because of the sensitive nature of biometric data. This section aims to discuss the system's privacy features.

To increase privacy the system adheres to the rules of data minimization only storing the essential information of users required to make the system functional. To that effect, only the minutiae points of fingerprints are encrypted and saved to perform fingerprint matching. The minimizing of sensitive data stored reduces the risk of privacy breaches. User consent is a fundamental aspect of any SSI system. As such the system implemented requires explicit user consent for any action or sharing of data. The user remains in complete control over their personal data. The process of creating a DID and registering biometric data is only done at the user's request. The authentication process can only be initiated by a user ensuring their consent to use biometric data for authentication. The reliance on user consent increases the protection of the users' privacy.

5.3.1 Linkability

With linkability, data can be linked to a single user, potentially revealing the identity or activities of the user. This is a serious privacy concern and for that reason, the system needs to be evaluated in terms of linkability.

The utilization of DIDs ensures that the users control their digital identities without their actual identities. Users decide which credentials are presented to verifiers, maintaining complete control over their personal information. The selective disclosure of identity information ensures only necessary information is shared, minimizing the linkability risk. The biometrics is encrypted with the AES CBC encryption approach. Each encryption is done with a different random IV ensuring that even the same plaintext gets different ciphertext output. This reduces the linkability of the encrypted biometrics. Even if the user registers multiple DIDs with the same fingerprint data the encryption process guarantees different ciphertexts. The linkability of biometric data definition is provided

in subsection 2.1.3 and mentions that two biometric templates are linkable if there is a way to detect that they came from the same original template. The encryption approach utilized guarantees that according to that definition, the encrypted biometrics are not linkable.

As discussed in section 5.2, the use of smart contracts increases the security of the system. At the same time, they introduce potential privacy concerns, especially regarding linkability. The public ledger records all transactions and operations making them publicly available. This transparency while beneficial in terms of security can lead to potential privacy concerns if transaction patterns are analyzed. The analysis of transaction patterns can introduce even more linkability concerns depending on the inappropriate usage of the system by the user. If a user has a single DID to store all of his credentials, every transaction with these credentials links back to the same DID. The usage of the system in such a way would severely increase linkability. To ensure safety from linkability the user would have to be educated on this issue or the system could be adapted to only allow a limited number of credentials added to a single DID.

5.4 Scalability

Evaluation of the current system in terms of scalability is an essential aspect of the system, as it aims to handle a large number of users and transactions over time. This section assesses the scalability of the system to gather knowledge on what future improvements could be made.

While the blockchain provides security and transparency it introduces some scalability challenges. The gas usage evaluation showed the system's limitation in relying on the current blockchain. The main issue is the high gas fees. With the current prices of the contract transactions, any user is deterred from using the system. For the system to garner any users the pricing problem has to be fixed. Another potential issue with the reliance on the blockchain is the limitation of transaction throughput. The Ethereum network can only handle a certain amount of transactions per second. Depending on the number of users in the system this could be a future limitation in terms of scalability.
The system employs off-chain processing to ensure performance efficiency. The performance in terms of time is positive and currently does not seem like a limiting factor for scalability. However, testing with more users in the system would have to be performed to confirm this.

5.5 Comparison with Existing Solution

In this section, the proposed system is compared to the system proposed by Acquah et al. [1], described in subsection 2.2.2. Although it is not a system featuring authentication of users via biometrics but only the secure storage of biometrics. Other implementations of biometric authentication based on smart contracts are rare and the ones that do exist do not feature any evaluation that would allow the comparison. Therefore the system

proposed by Acquah et al. [1] was chosen, as the comparison of the two systems can still lead to some valuable insights.

In the work of Acquah et al. [1], the hash of the fingerprint minutiae data is saved on the blockchain whilst the data itself is stored on the IPFS. How much gas is used and the cost associated with this is shown in Table 5.7, with the values taken from Acquah et al. [1]. The cost in ETH, CHF, and USD are shown according to the conversation rates at the time of writing (July 2024). Saving the biometric in this system has a gas usage of 147'982 which corresponds to a cost of 0,59 USD. As seen in Table 5.1 registering a DID and storing the encrypted biometrics requires gas usage from 1'406'862 to 3'426'131 which corresponds to 5,63 USD and 13,71 USD. This shows the clear economic advantage of only having to store the hash on the blockchain, as significantly less data has to be stored. The disadvantage on the performance side comes in the form of time, as the system of Acquah et al. [1] takes longer to store and get the data.

sendTemplate()	
Gas usage	147982
Cost in ETH	$2,08 \times 10^{-4}$
Cost in CHF	0,53
Cost in USD	0,59

Table 5.7: Gas usage and cost for storing biometric template with the system from Acquah et al. [1]

In terms of security, both systems offer the same advantages as traditional fingerprint storage systems. Both systems rely on decentralized structures and smart contracts which provide advantages in terms of transparency as well as immutability. It can be argued that the system proposed by Acquah et al. [1] is even more secure as only the hash of the fingerprint data is stored on the blockchain which can be argued is more secure than encrypting the fingerprint data. Analyzing the privacy of the system is more difficult as the system of Acquah et al. [1] offers no functionalities of identity management systems other than the storage of private information. It can be said that this single functionality preserves the privacy of its users as no information reveals the identity of the users in any way. The same can be said for the system implemented in this work as only storing the encrypted biometrics preserves privacy. Difficulties can arise with the additional functionalities of credentials and authentication as discussed in section 5.3. But to compare the privacy-preserving mechanisms of these functionalities the system of Acquah et al. [1] would have to be extended to be considered an identity management system. The scalability issue is lowered by the reduction of gas usage and the associated costs. But the increased complexity of the system of Acquah et al. [1] could lead to more transaction strain resulting in increased scalability issues.

The system of Acquah et al. [1] offers no other functionality than a distributed storage solution for biometrics, but still, the comparisons with the system implemented and evaluated in this work provide insights into what the system already does well and what could be improved in the future. Only storing the hash of the biometric data on the blockchain leads to significantly less gas usage, cost reduction, and scalability improvements, but could come with more time required for operations. Privacy and security of the user's

biometric data are guaranteed in both systems, whilst only saving the hash of biometrics on the blockchain could provide more safety.
A biometric authentication system based solely on saving the hash of fingerprint data does not work. Measuring the same fingerprint leads to slightly different data every time, meaning the hash of the biometric data would be completely different every time and could not be compared for authentication purposes. Still reducing the gas usage and cost of the system solutions based on saving the hash of biometrics on the blockchain with access to the biometrics stored in a decentralized manner seems the most promising option. Further work has to be done to explore this option and integrate such a system with an SSI system utilizing biometrics for authentication.

5.6 Evaluation Summary

The evaluation of the performance of the system shows that the system incurs high gas fees, making it expensive for users. The authentication time efficiency is proficient with average times of 187 milliseconds for 30 minutiae points and 355 milliseconds for 80 minutiae points. The system is well-optimized for quick authentication of users. Using smart contracts increases security by providing transparency, immutability, decentralization, and automation. The hybrid encryption approach using AES-256 for symmetric encryption and RSA-2048 for asymmetric encryption ensures the robust protection of biometric data. Data minimalization and user consent ensure reducing privacy risks. The employment of DIDs and selective disclosure minimize the risk of linkability. The encryption method ensures the unlinkability of the encrypted biometric data. The smart contract's transparency can introduce linkability concerns if transaction patterns are analyzed. Efficient off-chain processing reduces the blockchain load and improves performance. The high gas fees are the biggest challenge in system scalability.

The comparison with the solution by Acquah et al. [1] of securing biometrics on a distributed storage system shows improvements in decentralized storage compared to traditional centralized storage solutions. Additionally, the upside of developing a solution saving the hash of biometrics on the blockchain and the encrypted biometric data itself on a distributed storage system in combination with an SSI system is shown. This could improve performance in terms of cost and scalability, while the execution time could suffer.

The system promises strong security and efficient performance but still faces challenges in terms of gas cost and scalability. Addressing these issues, while ensuring the privacy of the user will be critical for adopting the system into real-world use.

Chapter 6

Final Considerations

6.1 Discussion

The thesis goal of implementing a system using biometrics to authenticate users in an SSI system and evaluating the system was accomplished, thereby creating the first work featuring the implementation and evaluation of such a system. The background research and literature review on the adjacent topics of SSI systems and biometrics inspired the design of the system. The initial idea was to design a system managing biometric data in an SSI system, no thought was put into why biometric data would be in that system. The literature review showed the only reason to include biometric data would be to authenticate users based on biometrics. Now the project not only included the secure management of biometric data in an SSI system but also the authentication of users based on that biometric data. The implementation of such a system proved to be more challenging than anticipated. First was the problem of finding an appropriate codebase to build upon and based on that learning about Smart Contracts and how to code them. How to handle the biometric data with smart contracts in order to achieve the authentication was another challenge, until the structure with performing the encryption/decryption as well as matching the fingerprints off-chain was conceived. These implementation challenges with the authentication took more time than anticipated and were part of the reason why no second privacy-preserving mechanism was implemented. Another challenge was finding another mechanism to secure the biometrics while still allowing authentication to work. Before hybrid encryption was considered, the first idea was to hash biometric data to secure it. An example of this that did not work out was using Zero-Knowledge Proof. The problem with these hashing approaches is they do not work with authentication based on biometric data. Measurements of the same fingerprint lead to slightly different results which leads to the hashes being completely different making the comparison impossible. As there was only one privacy-preserving mechanism to be able to compare the results found the comparison featured another system previously presented in the related work section. This system only featured secure storage of biometrics and no other identity management or authentication functionalities. This system was chosen because no paper could be found with a biometric authentication system and evaluation of that system to make the comparison of the systems possible. Other than not being able to implement

a second system the schedule could be followed with only small adjustments necessary. Those adjustments included more time needed for the development of the design and implementation, but because there was enough time detriment for the final review and corrections, the work schedule could be shifted by shortening the time allotted for final touches.

6.2 Conclusions

Developing an SSI system featuring biometric authentication is a novel idea with potentially huge upsides for users in terms of data ownership, security, and privacy, but it is currently not feasible, because the gas usage and the correlated transaction costs of smart contracts are too high. Consequently, more research and improvements are required before real-world adoption is possible. The use of biometrics for authentication in SSI systems is a complex subject matter. Storing the biometrics in encrypted form makes them secure, but it is not enough for implementations based on blockchain as gas usage is still too high with the amount of data sent in the transactions. A system based solely on this approach will not work unless the gas usage fees of smart contracts are significantly decreased in the future.
One possible solution would be the hashing of the biometric data, to decrease the amount of data sent. But this makes biometric matching to authenticate users impossible. To make storing the hashed biometrics a possible solution it would therefore have to be combined with storing encrypted biometrics in a distributed storage system, making retrieval of the biometrics for authentication possible. This proposal is based on the results of the evaluation. An implementation of such a system would have to be evaluated in terms of gas usage and authentication time to examine the feasibility.
If such a solution and other solutions prove not to be suitable, further improvements on smart contracts may be necessary before the adoption of an SSI system utilizing biometric authentication becomes adaptable for real-world usage. Furthermore, the discussion could then be held if biometric authentication with SSI systems and smart contracts are necessary, or if other solutions for authentication in an SSI system would be better suited.

6.3 Future Work

To extend this work itself implementing a system that gathers biometric data from real fingerprints and analyzing the system with the data gathered from those is the natural next step. This analysis would help determine the time and cost the authentication process takes with real biometric fingerprint data. The analysis would further allow the adjustment to threshold values inside of the matching function in Listing 4.10 further optimizing the matching process.
Additionally, advancing the system to require less gas usage for the execution of the contract transactions. This is partly achieved as further research and development is made with smart contracts in general. On top of that research into the possibility of a system

featuring the storage of hashed biometrics on the blockchain, while the encrypted biometrics are saved on a distributed system, retrievable for biometric matching, could be explored.
Otherwise, further privacy and security-preserving mechanisms are to be designed, implemented, and evaluated, for comparison against each other and previous solutions, to gain valuable insights and propel the evolution of SSI systems.

Bibliography

[1] Moses Arhinful Acquah et al. *Securing fingerprint template using blockchain and distributed storage system*. 2020.

[2] Christopher Allen. *The Path to Self-Sovereign Identity*. URL: `https://www.lifewithalacrity.com/article/the-path-to-self-soverereign-identity/`.

[3] Mehmet Aydar, Serkan Ayvaz, and Salih Cemil Cetin. "Towards a Blockchain based digital identity verification, record attestation and record sharing system". In: *arXiv preprint arXiv:1906.09791*. 2019.

[4] Luis Bathen et al. "Selfis: Self-sovereign biometric ids". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[5] Carmen Bisogni et al. "ECB2: A novel encryption scheme using face biometrics for signing blockchain transactions". In: *Journal of Information Security and Applications*. Vol. 59. 2021, p. 102814.

[6] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform". In: vol. 3. 37. 2014, pp. 2–1.

[7] Spela Cucko et al. "Towards the classification of self-sovereign identity properties". In: *Ieee Access*. Vol. 10. 2022, pp. 88306–88329.

[8] Jignasha Dalal et al. "Verification of identity and educational certificates of students using biometric and blockchain". In: *Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST)*. 2020.

[9] Oscar Delgado-Mohatar et al. *Blockchain meets biometrics: Concepts, application to template protection, and trends*. 2020.

[10] Uwe Der, Stefan Jähnichen, and Jan Sürmeli. "Self-sovereign identity − opportunities and challenges for the digital revolution". In: *arXiv preprint arXiv:1712.01767*. 2017.

[11] Paul Dunphy and Fabien A.P. Petitcolas. "A First Look at Identity Management Schemes on the Blockchain". In: *IEEE Security & Privacy*. Vol. 16. 4. 2018, pp. 20–29.

[12] Md Sadek Ferdous et al. "Security usability of petname systems". In: *Identity and Privacy in the Internet Age: 14th Nordic Conference on Secure IT Systems, NordSec 2009, Oslo, Norway, 14-16 October 2009. Proceedings 14*. Springer. 2009, pp. 44–59.

[13] Marta Gomez-Barrero et al. "General Framework to Evaluate Unlinkability in Biometric Template Protection Systems". In: *IEEE Transactions on Information Forensics and Security*. Vol. 13. 6. 2018, pp. 1406–1420.

[14] Geoff Goodell and Tomaso Aste. "A decentralized digital identity architecture". In: *Frontiers in Blockchain*. Vol. 2. 2019, p. 17.

[15] Tom Hamer et al. *Private digital identity on blockchain*. 2019.
[16] *ISO/IEC 24745:2022*. ISO Standard. International Organization for Standardization, 2022.
[17] Evan Krul et al. "SoK: Trusting Self-Sovereign Identity". In: *arXiv preprint arXiv:2404.06729*. 2024.
[18] E. H. Lahav. *The OAuth 1.0 Protocol*. URL: `https://datatracker.ietf.org/doc/html/rfc5849`.
[19] Yaoqing Liu, Guchuan Sun, and Stephanie Schuckers. "Enabling secure and privacy preserving identity management via smart contract". In: *2019 IEEE conference on communications and network security (CNS)*. IEEE. 2019, pp. 1–8.
[20] Nuno Martins, José Silvestre Silva, and Alexandre Bernardino. "Fingerprint Recognition in Forensic Scenarios". In: vol. 24. 2. 2024, p. 664.
[21] Prince Mishra et al. "Pseudo-Biometric Identity Framework: Achieving Self-Sovereignity for Biometrics on Blockchain". In: *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2021, pp. 945–951.
[22] Asem Othman and John Callahan. "The Horcrux Protocol: A Method for Decentralized Biometric-based Self-sovereign Identity". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–7.
[23] Hatef Otroshi Shahreza, Yanina Y. Shkel, and Sebastien Marcel. *Measuring Linkability of Protected Biometric Templates Using Maximal Leakage*. 2023.
[24] Nalini K Ratha et al. "Generating cancelable fingerprint templates". In: *IEEE Transactions on pattern analysis and machine intelligence*. Vol. 29. 4. 2007, pp. 561–572.
[25] Shreyansh Sharma, Anil Saini, and Santanu Chaudhury. "Multimodal biometric user authentication using improved decentralized fuzzy vault scheme based on Blockchain network". In: *Journal of Information Security and Applications*. Vol. 82. 2024, p. 103740.
[26] Reza Soltani, Uyen Trang Nguyen, and Aijun An. "A survey of self-sovereign identity ecosystem". In: *Security and Communication Networks*. Vol. 2021. 2021, pp. 1–26.
[27] Yagiz Sutcu et al. "What is biometric information and how to measure it?" In: *2013 IEEE international conference on technologies for homeland security (HST)*. IEEE. 2013, pp. 67–72.
[28] Foteini Toutara and Georgios Spathoulas. "A distributed biometric authentication scheme based on blockchain". In: *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE. 2020, pp. 470–475.
[29] W3C. *Decentralized Identifiers (DIDs)*. URL: `https://www.w3.org/2017/vc/WG/`.
[30] Fennie Wang and Primavera De Filippi. "Self-Sovereign Identity in a Globalized World: Credentials-Based Identity Systems as a Driver for Economic Inclusion". In: *Frontiers in Blockchain*. Vol. 2. 2020.
[31] James L. Wayman. "Biometrics in Identity Management Systems". In: *IEEE Security & Privacy*. Vol. 6. 2. 2008, pp. 30–37.
[32] Naser Zaeri. "Minutiae-based fingerprint extraction and recognition". In: vol. 10. 2011, p. 17527.
[33] Qixin Zhang. "An overview and analysis of hybrid encryption: the combination of symmetric encryption and asymmetric encryption". In: *2021 2nd international conference on computing and data science (CDS)*. IEEE. 2021, pp. 616–622.
[34] Weiqin Zou et al. "Smart contract development: Challenges and opportunities". In: *IEEE transactions on software engineering*. Vol. 47. 10. 2019, pp. 2084–2106.

Abbreviations

AES	Advanced Encryption Standard
BOPS	Biometric Open Protocol Standard
BTP	Biometric Template Protection
CBC	Cipher Block Chaining
CHF	Swiss Franc
DID	Decentralized Identifier
ETH	Ether (Cryptocurrency)
EVM	Ethereum Virtual Machine
fmp1	Fingerprint Minutiae Points 1
fmp2	Fingerprint Minutiae Points 2
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IPFS	InterPlanetary File System
ISO	International Organization for Standardization
IV	Initialization Vector
KYC	now-Your-Costumer
OAuth	Open Authorization
OAEPHash	Optimal Asymmetric Encryption Padding Hash
PEM	Privacy-Enhanced Mail
PIN	Personal Identification Number
RDM	Random Distance Methods
RSA	Rivest-Shamir-Adleman
SHA-3	Secure Hash Algorithm 3
USD	United States Dollar
W3C	World Wide Web Consortium

List of Figures

List of Tables

Listings

Appendix A

Installation Guidelines

The code can be found in the following GitHub repository:
`https://github.com/jverho/SSI-Biometric-Authentication`
There you can find a detailed installation and operation guide in the README.md file.