



University of
Zurich^{UZH}

Connecting Physical Identity with Blockchain-based Self-Sovereign Identity

*Harris Sohrab Alem Yar
Zurich, Switzerland
Student ID: 21-705-124*

Supervisors: Daria Schumm, Thomas Grübl,
Prof. Dr. Burkhard Stiller
Date of Submission: January 15, 2025

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 All passages taken verbatim or in sense from published or unpublished writings are identified as such.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.

Zürich, 15.01.2025

A handwritten signature in black ink, appearing to read 'H. Meyer', is written above a horizontal line.

Signature of student

Abstract

This thesis explores the integration of physical identity with blockchain-based Self-Sovereign Identity (SSI) systems, focusing on privacy and user control. It identifies challenges of existing solutions, particularly the reliance on biometric data and its risks to user autonomy. To address these issues, this thesis proposes the system “Credchain-ZKP” using zero-knowledge proofs to verify physical identity without disclosing biometric data. Two prototypes are implemented and evaluated in terms of performance, cost, scalability, security and privacy. The results demonstrate the feasibility of securely linking physical and digital identities while adhering to SSI principles.

Zusammenfassung

Diese Bachelorarbeit untersucht die Integration physischer Identität in Blockchain-basierten Self-Sovereign Identity (SSI) Systemen mit Fokus auf Datenschutz und Datenautonomie von Nutzenden. Sie zeigt die Grenzen aktuell implementierter Lösungen auf, insbesondere die Nachteile durch die Verwendung von biometrischen Daten in Bezug auf Nutzerautonomie. Als Antwort auf diese Probleme wird das System “Credchain-ZKP” konzipiert, das Zero-Knowledge-Proofs einsetzt, um physische Identitätsmerkmale zu verifizieren, ohne sensible biometrische Daten preiszugeben. Zwei Prototypen werden implementiert und hinsichtlich Performance, Kosten, Skalierbarkeit, Sicherheit und Datenschutz bewertet. Die Ergebnisse zeigen, dass physische und digitale Identitäten unter Wahrung der SSI-Prinzipien erfolgreich verknüpft werden können.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Daria Schumm, Thomas Grübl, and Prof. Dr. Burkhard Stiller, for their unwavering support, insightful feedback, and guidance throughout my Bachelor's Thesis. Their expertise and the collaboration have been invaluable in fostering my progress during the development of this project.

I also want to thank my family, as well as Noemi Gangl, Maximilian Pefestorff and Stefan Hoffmeister, for their support, advice, and proofreading. Their help and encouragement have been vital throughout this journey.

Contents

Declaration of Independence	i
Abstract	iii
Acknowledgments	vii
1 Introduction	1
1.1 Thesis Goals	2
1.2 Thesis Outline	2
2 Fundamentals	3
2.1 Background	3
2.1.1 Identity	3
2.1.2 Proofs of Identity	4
2.1.3 Authentication	4
2.1.4 Self-Sovereign Identity (SSI)	4
2.1.5 Zero-Knowledge Proofs (ZKP)	5
2.2 Related Work	6
2.2.1 Related Work on Existing Digital and Physical Implementations . .	6
2.2.2 Related Work on Zero-Knowledge Proofs in SSI Systems	13
2.2.3 Research Gap and Problem Statement	14

3	Design	15
3.1	Design Principles and Requirements	15
3.1.1	Related Design Principles and Requirements	15
3.1.2	Additional Requirements for the Proposed Design	19
3.2	Design: Credchain-ZKP and Credchain On-Chain ZKP	22
3.2.1	Starting Point: TCID and Credchain	22
3.2.2	Integrating Biometric zk-SNARK Zero-Knowledge Proofs	22
3.2.3	Architecture and User Flows	23
3.2.4	Key Components	27
3.2.5	Addressing Non-Functional and Functional Requirements	28
4	Implementation	31
4.1	Credchain with 2FA	31
4.1.1	RegisterIdentity	33
4.1.2	VerifyDID	34
4.2	Credchain-ZKP	35
4.2.1	RegisterIdentity	37
4.2.2	GenerateProof	38
4.2.3	VerifyDID	39
4.2.4	Circom Circuit	40
4.2.5	Trusted Setup Generation	41
4.3	Credchain On-Chain ZKP	43
4.3.1	VerifyDID	45
4.3.2	Verifier.sol	46

<i>CONTENTS</i>	xi
5 Evaluation	47
5.1 System Performance	47
5.1.1 Execution Time: <i>GenerateProof</i>	48
5.1.2 Execution Time: <i>VerifyDID</i>	49
5.2 Cost	50
5.2.1 Credchain with 2FA	51
5.2.2 Credchain-ZKP	51
5.2.3 Credchain On-Chain ZKP	52
5.3 Scalability	52
5.4 Security	53
5.5 Privacy	53
6 Final Considerations	55
6.1 Summary	55
6.2 Conclusions	56
6.3 Future Work	57
Bibliography	57
Abbreviations	63
List of Figures	63
List of Tables	65
List of Listings	67
A Installation and Deployment	71
A.1 System's Point of View	71
A.2 Installation	72
A.3 Deployment and Startup	72
A.4 Data Structures	74

Chapter 1

Introduction

Digital identity solutions have become more and more common in both governmental and private sectors, promising easier interactions and less physical documents. However, many of these solutions only offer basic digital versions of existing IDs (e.g. an electronic Passport). These approaches typically fail to address challenges that Self-Sovereign Identity (SSI) solutions aim to cover, such as returning ownership and control of personal data to users.

Recent research points to a need for methods linking physical identity to digital, self-sovereign credentials in a reliable way. For instance, NIST¹ specifications explicitly exclude physical aspects [21], leaving the question of real-world identity largely unanswered. Similarly, [43] explains that understanding of how physical identity could be integrated within SSI systems would be valuable. These gaps show why such systems should not only manage digital, but also physical credentials.

While exploring existing ways of combining physical and digital identities, one challenge reoccurred for many systems: How can verifiers ensure that an individual matches the presented credentials? While some systems use sensitive biometric data to perform this match, this approach does not fulfill principles of SSI such as data minimization or user autonomy. Filling this gap, zero-knowledge proofs have emerged as a promising option, allowing user details to be verified without revealing any data. This alignment with SSI principles makes zero-knowledge proofs particularly suitable for bridging the gap between physical and digital identity.

¹United States' National Institute of Standards and Technology

1.1 Thesis Goals

This thesis intends to explore how physical identity can be securely linked with SSI while maintaining user control and privacy. In line with the tasks described in the project, the work is divided into several key goals:

- Establish background and foundations: Develop the necessary theoretical groundwork on SSI and clarify the significance of incorporating physical identity into digital identity systems.
- Review existing approaches: Conduct a comprehensive literature review to identify current methods for linking physical identity to SSI, evaluate their strengths and weaknesses, and develop a clear problem statement.
- Define a mechanism for integration: Propose a mechanism designed specifically for connecting physical identity and SSI, ensuring that it meets high standards of privacy, security, and data minimization.
- Specify requirements and design architecture: Derive both functional and non-functional requirements to guide the design of a prototype architecture. This architecture should enable physical and digital identity connectivity in a reliable and user-centric way.
- Implement a prototype: Build a simplified SSI system – comprising issuer, verifier, and identity holder – that integrates physical identity features while upholding privacy and security principles.
- Evaluate performance and security: Carry out a comprehensive assessment of the prototype, focusing on data confidentiality, system scalability, performance metrics, and cost. Compare these findings to a baseline setup that does not include physical identity, highlighting the benefits and any trade-offs.

1.2 Thesis Outline

In chapter 2, key concepts and theories related to identity and SSI are introduced. This includes definitions of identity, authentication, and zero-knowledge proofs. The chapter also reviews related work and identifies gaps this thesis aims to address. Chapter 3 outlines the proposed design. It explains the principles, requirements, and architecture needed to connect physical identity with SSI using zero-knowledge proofs. Chapter 4 describes the implementation of the prototypes. It focuses on how the design was implemented, the technologies used, and how biometric data and proofs are processed. Chapter 5 evaluates the prototypes by measuring their performance, cost, scalability, and security. It also compares the system's privacy features to existing solutions. Chapter 6 concludes the thesis, highlighting the main contributions and the system's role in securely linking physical and digital identities, and identifies potential improvements for future research.

Chapter 2

Fundamentals

This chapter introduces concepts and related work that form the foundation of this thesis. It begins with an overview of core theories and relevant definitions. The discussion then turns to an analysis of related work and the presentation of the problem statement.

2.1 Background

This section begins by defining key terms related to identity, explaining how identity can be authenticated and how this process connects to Self-Sovereign Identity systems. It then moves on to zero-knowledge proofs, providing an overview and illustrating different mechanisms.

2.1.1 Identity

At the core of every authentication process is the principle of identity verification, across varying levels of trust and specific requirements. [44] define identity as the comprehensive set of attributes defining a person uniquely. These attributes describe properties of persons like height, hair color, citizenship, or a name and have no means to be unique themselves, but can be shared by multiple persons.

To identify a person, select non-unique attributes are insufficient. An identifier, on the other hand, works as a reference to a person. Such references can be assigned (e.g. names) or be a "particular representation of an observable property" (e.g. unique attributes such as biometric data) [44], based on the context and the environment they are being used in.

2.1.2 Proofs of Identity

Identity proofing and authentication describe the process through which a person's identity is determined and verified using specific identity attributes [21]. The United States' National Institute of Standards and Technology (NIST) outlines three fundamental components of this process in their Digital Identities Guidelines [21], applicable across various contexts and environments:

1. Resolution: Attributes (e.g. height or hair color) and provided identity evidence (e.g. passport or social security number) are used to establish a reference to a person.
2. Validation: Assessment of the requested identity evidence and attributes: Are they present, valid, authentic and accurate?
3. Verification: Upon successful validation, the identity of a claimant can be confirmed.

Several sources also categorize the strength or reliability of the presented proof of identity, requiring that certain criteria must be fulfilled for varying levels of confidence. These criteria include different types of core attributes presented, identity evidence presented or usage of specific authentication and verification processes [19, 21].

2.1.3 Authentication

A widely used **authentication process** is Multi-Factor Authentication (MFA). This method is based on the categorization of attributes [20, 37]:

1. Something you **know**, such as a password or a username.
2. Something you **have**, such as an ID card, a physical badge, or a smartphone.
3. Something you **are**, referring to biometric characteristics, such as fingerprints or iris scans.

Combining two or more of these factors (MFA) results in higher levels of assurance and thus higher confidence in the identity of the claimant [14].

2.1.4 Self-Sovereign Identity (SSI)

Self-Sovereign Identity is a user-centric approach to digital identity management, allowing individuals to control, present and verify their digital identity without relying on centralized institutions. In contrast to traditional identity systems with service providers managing identities, SSI uses decentralized technologies. Users create Decentralized Identifiers (DIDs) stored on the blockchain acting as references to, in this case, human users.

Verifiable Claims, digital attestations issued by third party entities, present single attributes related to the user. For instance, a Verifiable Claim could certify the user's age, educational qualification, or membership in a specific organization. This system allows identity verification by presenting only the necessary attributes to any other party, without any central authority, enhancing autonomy, privacy and security while maintaining trust [1].

2.1.5 Zero-Knowledge Proofs (ZKP)

Zero-knowledge proofs are cryptographic protocols that allow a prover to convince a verifier of the truth of a statement without revealing any information beyond the validity of the statement itself. This ensures the confidentiality of sensitive information while enabling verification and trust [18].

Multiple types of zero-knowledge proofs exist. Most can be categorized as follows:

- **Interactive Zero-Knowledge Proofs:** Interactive ZKPs involve multiple rounds of communication between prover and verifier. The verifier issues challenges and the prover must respond correctly to each [48].
- **Non-Interactive Zero-Knowledge Proofs:** In a non-interactive ZKP, the prover and verifier do not need to communicate multiple times – the prover sends one single proof to the verifier which can be verified independently [48].
- **zk-SNARKs** (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge): zk-SNARKs are compact non-interactive proofs that are quick to verify, relying on a trusted setup phase for each proof circuit. Prover and verifier both need knowledge of the keys generated through the verified setup [48].
- **zk-STARKs** (Zero-Knowledge Scalable Transparent Arguments of Knowledge): zk-STARKs are an evolution of zk-SNARKs removing the need for a trusted setup, improving scalability [48].
- **Bulletproofs:** Bulletproofs are a type of zero-knowledge proof specifically designed for range proofs, for instance verifying whether an individual's age meets a minimum requirement [3].

2.2 Related Work

This section examines existing SSI systems and explores their details, limitations and current interactions with the physical world. This analysis is concluded with a summarizing table 2.1 and a presentation of the observed research gap and the developed problem statement.

2.2.1 Related Work on Existing Digital and Physical Implementations

In reviewing related work, a systematic approach was adopted to compare existing implementations across several dimensions. The following criteria were used:

- Purpose and Availability: The motivation behind the system's creation and its implementation status.
- Enrollment: Registration and identity verification processes, including any physical elements, if present.
- Presentation and Transmission: Presentation of identity, types of credentials used, and methods of transmission.
- Identity Assurance: Use of third-party attestations and the level of trust established.
- Usability: Ease of enrollment and presentation for users and verifiers, including any requirements such as smartphones or internet connection.
- Security and Privacy: Protection measures against fraud or identity theft.
- Connection of Physical and Digital Identity: Any findings specific to the interconnection between physical and digital identities.
- FR/NFR: Presence of specific functional or non-functional requirements.
- Specific remarks and limitations.

Six implementations have been selected for a detailed analysis based on their unique capabilities for integrating physical identity within Self-Sovereign Identity systems, followed by a summarizing table 2.1.

2.2.1.1 Worldcoin / World ID

- **Purpose and Availability:** Worldcoin and World ID were created to build a global digital identity and currency system. Unique to the system is the use of biometric data (iris scans) to verify "humanness" [45]. At the time being, over 6 million people are enrolled.
- **Enrollment:** Users can enroll by scanning their iris by an "Orb", an iris scanner, and enregister via the mobile app. The hashed iris scan is then saved in Worldcoin's database. It is not possible to authenticate by any other means to ensure that users only enroll once [16, 45].
- **Presentation and Transmission:** The World ID is presented through the Worldcoin app. After scanning a QR code of a service, the app generates zero-knowledge proofs that allow users to authenticate themselves without revealing personal info. These proofs are computed on the user's device. The transmission works over the Worldcoin network off-chain – not peer-to-peer, but over their central servers. The blockchain is only used to set the uniqueness of World IDs [45].
- **Identity Assurance:** There are no traditional third-party attestations; trust is based on the uniqueness of the biometric-derived World ID.
- **Usability:** Users must physically access an Orb for enrollment and then use a phone app. Access to an Orb is physically limited by their availability. As of September 2024, there are five Orbs available in entire Germany. Additionally, an internet connection is usually required to connect to verifiers. Verifiers can integrate an API into their services to support the World ID. [45].
- **Security and Privacy:** Using unique biometric protects against fraud. Zero-knowledge proofs ensure that users can prove their human uniqueness without releasing personal information. However, use of biometric data as well as the transmission is handled by Worldcoin's servers, raising concerns regarding privacy and data control [16, 29].
- **Specifics:** Even though World ID aims at privacy-respecting digital ID, classification of this system as truly self-sovereign is unclear, since it is lacking decentralization in many parts where a high degree of reliance on Worldcoin as a central instance is necessary [16].
- **Connection of physical and digital Identity:** World ID directly links physical biometric data (iris scans) to digital identities by creating World IDs referenced on the blockchain. Using proprietary hardware to scan the iris of users and using this proof of uniqueness forms an interesting approach. However, this sole focus on uniqueness does not hold for more advanced use cases of user identification, such as determining if a user is authorized to access a service or transiting a border.
- **FR/NFR:** World ID primarily addresses functional requirements by pursuing unique human identity verification and proof of personhood [45].

2.2.1.2 Kiva Protocol

- **Purpose and Availability:** The Kiva Protocol allowed individuals to use a digital identity that combines personal details and biometrics to access financial services [44]. The project has been initially rolled out in Sierra Leone in 2018 and was discontinued by 2022 [30, 44].
- **Enrollment:** People can sign up by registering at a governmental polling station, providing biometric and personal data such as a fingerprint, their name and their birthdate [44].
- **Presentation and Transmission:** Enrolled users can then either use a digital wallet to present their identity or let a financial institution retrieve their identity based on user's biometrics [44].
- **Identity Assurance:** Governmental entities enroll users and ensure their identity at sign-up using their personal details and biometrics [44].
- **Usability:** Users do need a governmental ID to enroll. A smartphone can be used to manage their digital identity, but is not a requirement. Verifiers need proper access to the system and biometric equipment, which may limit usability for institutions not willing or able to provide that [44].
- **Security and Privacy:** The DID published only confirms that the premised holder with these personal details is a citizen of that country. Information about a loan is being published on-chain and attached to a DID; the private key is stored either on the user's smartphone or on Kiva's servers in case they don't have a smartphone [44]. This raises questions on the part of Kiva's data protection measures. The protocol itself is based on the Selective Disclosure principle. As such, a single bank will not be able to see all interactions and the full history of the user with other banks, but only transactions attached to its own sub-DID. Furthermore, the data usage is designed to be as minimal as possible – including the fact that the biometric data is not being used by the financial institute primarily, but only as a second factor in case the user does not own a smartphone [44].
- **Connection of physical and digital Identity:** In the enrollment process, users need to use their biometric data to be set up with a digital identity. At presentation, they can verify their identity digitally through an app or physically through biometric scans [44]. There is no physical transmission mechanism or mandatory physical, e.g. biometric authentication process known.
- **FR/NFR:** The Kiva Protocol mainly focuses on the functional requirements addressing digital identity creation and establishing credit history. It also addresses non-functional requirements and balances aspects like security (possibility to have private keys on your digital wallet) and availability (letting Kiva manage your private keys and authenticate using a biometric second factor) [30, 44].

2.2.1.3 Civic Pass

- **Purpose and Availability:** The Civic Pass aims to provide secure identity verification on demand and with low cost, while giving users control over their personal information [34] and enabling compliance with regulations, such as Know Your Customer protocols [36].
- **Enrollment:** Users register by scanning physical, government-issued IDs and a live facial scan comparison to it through the Civic app [6]. Upon verification, Civic generates a DID stored on the blockchain [36]. Additionally, it is possible to obtain a physical "Civic ID card" by physical verification at an "authorized provider" [5]. This enables physical use of the digital ID.
- **Presentation and Transmission:** Identity is presented via their app once users choose to share verified information and transmitted directly, off-chain, to the service provider. Different methods of communication with a service provider seem to be supported via the app: QR codes, Bluetooth and near field communication [4]. The physical ID allows offline and NFC-based verification [5]. For service providers, Civic provides APIs for easier integration [7].
- **Identity Assurance:** The compliance with Know Your Customer regulations and the use of government-issued IDs contribute to maintaining a high level of trust. However, given that Civic acts as both the attestation provider and verifier, a service provider needs to trust the processes and the API implemented by Civic.
- **Usability:** Using the app, it is possible to onboard online using a mobile phone with a camera and an internet connection. For presentation, an internet connection is required. Using the physical Civic ID card, it is possible to present ID offline, but a prior physical visit to a provider's office is necessary.
- **Security and Privacy:** Using biometric and especially in-person verification ensures high trust levels. Users' personal info and documents are only stored on their mobile devices and hashes of it are stored on the blockchain, ensuring no centralized storage of personal data [31].
- **Connection of physical and digital Identity:** Civic relies heavily on physical elements such as a governmental ID and a facial likeness scan for the initial enrollment. For the physical Civic ID card, physical presence at a verifiers office is needed. The physical Civic ID card enables users to access their digital ID offline through NFC, complementing the online use of their digital ID via QR codes, NFC, or Bluetooth, forming a bridge between the digital ID and its physical representation [5].
- **FR/NFR:** Civic addresses functional requirements (e.g. identity verification and access control) as well as non-functional requirements (e.g. accessibility and usability) through using the Civic Identity digitally and with their physical ID. [4, 5].
- **Specifics:** Due to Civic acting as the central instance in this process, this system does not match the requirement of a decentralized SSI system.

2.2.1.4 EBSI/ESSIF (EU)

- **Purpose and Availability:** The European Self-Sovereign Identity Framework (ESSIF) aimed at empowering EU citizens with a digital identity and enabling them to control and own their digital identity [17]. It was a project of the European Blockchain Service Infrastructure EBSI and was discontinued by 2022 [32]. Meanwhile, EBSI's operation and many elements of the ESSIF framework are still being continued.
- **Enrollment:** After an initial setup in a mobile application, the ESSIF relies on physical presence at enrollment for verification since only authorized institutions are able to issue a verifiable identity [13].
- **Presentation and Transmission:** ESSIF offers two possibilities of presenting identity: using biometric authentication – implemented by the mobile phone manufacturer – to access an app and share Verifiable Credentials or using an EU eID to log in, obtain a verifiable Credential and presenting it [12, 13].
- **Identity Assurance:** Only authorized, often governmental, institutions are eligible for issuing a verifiable identity [13]. Users are being verified at presentation by providing their biometrics to open the app or to log in with their EU eID [13]. Apart from that, no verification is mentioned on the service provider side.
- **Usability:** Enrollment requires users to install a compatible digital wallet, using a smartphone, internet connection, and initial physical verification. The integration of eID might have increased usability [13].
- **Security and Privacy:** Users can control their data and choose what information to display, enhancing privacy [13]. However, security measures are questionable since relying on smartphone-built authentication can be prone for manipulation, as there is no additional authentication possibility.
- **Connection of physical and digital Identity:** ESSIF connects physical identity documents to digital identities. Apart from the initial physical verification, no further specifically physical use cases are described for the SSI system. However, as mentioned, there are various efforts to further develop the learnings from this project. One example is the EU-designed specifications for digital wallets, whose advertised use cases include using it as a passport and thus international border crossings [15]. An international organization like the EU could particularly advance such applications of digital IDs and promote exchange protocols – also based on SSI.
- **FR/NFR:** ESSIF primarily targets functional requirements of cross-border digital identity verification and interoperability of the system. Additionally, it focuses on standardization by adhering and establishing EU-wide standards and regulations [15].

2.2.1.5 United Nations World Food Programme "Building Blocks":

- **Purpose and Availability:** The United Nations WFP (World Food Programme) implemented an iris scan authentication system to enable efficient food assistance distribution to refugees without relying on physical vouchers or actual money [42].
- **Enrollment:** UNHCR Jordan, the UN Refugee Agency in Jordan, registers refugees and scans their irises. They must be physically present in order to utilize proprietary iris scanners [42]. The patterns captured by the scanner create a unique, individual biometric template.
- **Presentation and Transmission:** At the point of service, e.g. a supermarket, beneficiaries can authenticate themselves by having their iris scanned. The iris scan serves as the sole authentication credential – there is no physical card, voucher, PIN or other method involved in that process [42].
- **Identity Assurance:** The WFP acts as a custodian of user's private keys. After successful biometric authentication, the keys are triggered to sign transactions on a private blockchain. Sensitive biometric and personal information stay off-chain and in a layer managed by WFP between the blockchain and the actual user [44].
- **Usability:** Beneficiaries do not need to carry documents or remember any credentials, authentication is as simple as looking into an iris scanner. The points of service need to be equipped and ensure a secure internet connection to the WFP [42].
- **Security and Privacy:** The WFP describes all protocols and procedures as secure and privacy-respecting. However, capturing biometric data and transmitting it to a server of WFP can bring risks. [44] also points out possible caveats of using biometric data: Biometric features can change, are constantly left behind everywhere by individuals (on photos or as fingerprints) and, unlike passwords, cannot be swapped in the event of an exploit. Additionally, the mandatory use of biometrics can raise ethical questions about consent, especially in vulnerable populations.
- **Connection of physical and digital Identity:** This system directly links the physical identity of an individual to their digital identity in the system and completely eliminates any other document. Furthermore, physical presence is required at enrollment as well as presentation.
- **FR/NFR:** This system addresses specific and use-case-related functional requirements, namely efficient aid distribution and identity verification. As for non-functional requirements, user-side usability of the systems without even needing any hardware increases accessibility [42].
- **Specifics:** This system described does not match SSI criteria: Users have no control over their identity data, biometric data is stored centrally and the system depends on UNHCR and WFP as central authorities.

2.2.1.6 TCID

- **Purpose and Availability:** TCID is a DID-based SSI system developed together with the Dutch government. It aims to provide a scalable digital SSI system usable for high-assurance identity verification, such as passport-grade identification [39, 40]. Their system is not ready for practical use but serves as a proof of concept for their proposed architecture.
- **Enrollment:** Users create a DID. Embedded in it is an unique string acting as a second factor that can be used for on-site verification. Possibilities range from PINs to facial recognition data [40].
- **Presentation and Transmission:** Users store their verifiable credentials in their digital wallet and share them on request, partially or completely. Bluetooth is used as an on-site communication mechanism. Communication takes place directly between user and verifier with no central authority inbetween [40].
- **Identity Assurance:** Governmental institutions and other trusted entities can verify users' identities and issue verifiable credentials. Service providers can then authenticate these credentials by verifying the digital signatures on the DIDs, along with the second authentication factor and the revocation status [40].
- **Usability:** Users need access to a smartphone and, for any biometric enrollment, to an authorized center. Verifiers need to be able to use Bluetooth for on-site communication and, if facial recognition is used, cameras. Since Bluetooth is used as a transmission protocol, no internet connection is necessary for the user at presentation.
- **Security and Privacy:** Strong encryption, digital signatures and physical factors (PIN or biometric features) help mitigate fraud and identity theft. Users can choose which attributes they want to share and all personal data is stored locally on devices.
- **Connection of physical and digital Identity:** The system integrates a second factor with a string on the DID that can be verified locally, making it suitable for high-security applications. The work also describes passport-grade ID representation in combination with facial scans with liveness detection [40]. Additionally, the ability to present credentials offline via Bluetooth increases the usability.
- **FR/NFR:** The system was built with special focus on the functional requirements of direct peer-to-peer communication without central intermediaries and integrity verification of the digital ID using a second factor as part of the DID [40].
- **Specifics:** The precautions and considerations regarding using biometric data, as outlined by [44] in subsection 2.2.1.5, are applicable in this context as well.

Criteria	Worldcoin / World ID	Kiva Protocol	Civic Pass	EBSI/ESSIF (EU)	UN WFP "Building Blocks"	TCID
Enrollment	Iris scan via "Orb" and app	Register at government station with biometrics	Scan ID and facial recognition via app; physical ID optional	Set up in app; physical verification required	Registered by UNHCR; iris scans taken	Create DID with unique string; may need on-site verification
Presentation and Transmission	app generates proofs; uses central servers	Present via wallet or biometrics at institutions	Present via app; uses QR codes, Bluetooth, NFC; physical ID possible	Use app or EU eID to share credentials	Iris scan at service points for authentication	Share credentials via Bluetooth; peer-to-peer
Identity Assurance	Trust in unique biometric ID; no third parties	Government verifies identity with biometrics	Civic verifies IDs; compliance with regulations	Authorized institutions issue identities	WFP manages keys; biometric data off-chain	Government verifies identity; providers authenticate credentials
Use of Hardware	"Orb" iris scanner; smartphone app	Biometric equipment; smartphones optional	Smartphone with camera; physical ID uses NFC	Smartphone app	Proprietary iris scanners	Smartphone with Bluetooth; possible biometric sensors
Centralization	Relies on central servers	Centralized elements; government and Kiva can manage keys if users decide not to	Civic is central authority; not fully decentralized	Involves government authorities; aims for stan- dardization	Centralized system managed by UNHCR and WFP	Decentralized; peer-to-peer communication
Connection of Physical and Digital Identity	Links iris scans to digital IDs on blockchain	Biometrics set up digital ID; verify via app or biometrics	Physical attributes (ID, facial scan); physical ID bridges digital and physical	Connects physical documents to digital IDs; includes digital wallets as passports	Links physical identity to digital; physical presence required	Second factor on DID; includes facial scans with liveness detection
Focus FR/NFR	FR: Unique human identity verification using iris scans	FR: Digital identity and credit history; NFR: balances security and availability	FR: Identity verification and access control; NFR: enhanced usability with digital and physical ID options	FR: Cross-border identity verification; NFR: EU-wide standardiza- tion	FR: Efficient aid distribution; NFR: high usability without user hardware	FR: Peer-to-peer communica- tion, integrity verification using second factor

Table 2.1: Analysis of Six Identity Management Systems

2.2.2 Related Work on Zero-Knowledge Proofs in SSI Systems

Literature on the interconnection of zero-knowledge proofs and identity management systems remains relatively limited. [48] explores foundational insights into the interconnection of zero-knowledge proofs, blockchain and Self-Sovereign Identity systems, focusing on the integration of zk-STARKs enabling selective disclosure of credentials. [41] provides a framework for zero-knowledge proofs applications in blockchain. Although not directly linked to identity systems, the work does propose zk-SNARKs as a solution for certain verifiable claims, such as proving account balances.

2.2.3 Research Gap and Problem Statement

While reviewing current solutions one challenge keeps arising: How is it possible to ensure that the person physically standing in front of a verifier truly matches the digital identity claimed? In many real-world contexts, smartphones and PIN numbers can be stolen or passed on to someone else and are subject to misuse [16, 30]. To address this, some solutions use biometric data. However, biometric data comes with disadvantages: Individuals potentially leave traces of biometric markers (e.g. photos, fingerprints) and unlike passwords, biometrics can not be replaced when compromised [16, 44]. Furthermore, some solutions use proprietary hardware to capture biometric data and process it further, being intransparent to the user and thus contradicting with self-sovereign identity principles of decentralization, user autonomy and data minimization [40].

Thus, the following problem statement explores the design of a SSI solution using biometrics for robust physical verification, yet keeping actual biometric data enclosed on the user's device and never disclosing it to any outside institution.

Problem Statement: Current high-assurance Self-Sovereign Identity systems for physical usage rely on biometric data, raising privacy concerns and conflicting with SSI principles of user control and data minimization. This thesis aims to enhance existing methods by securely integrating biometrics without disclosing actual biometric data. The goal is to develop an SSI system that enables high assurance levels in physical use cases while ensuring biometric data remains securely stored on the user's device without being disclosed to any external parties.

Chapter 3

Design

This chapter begins with the first section explaining general requirements for SSI systems and additional requirements based on the problem statement. The second section then presents the proposed design.

3.1 Design Principles and Requirements

This chapter begins with design principles and requirements formulated by other authors for general SSI systems. These various requirements and approaches can then be categorized into 13 clusters. Based on these clusters, additional requirements necessary to address the problem statement are derived and summarized in a table 3.2.

3.1.1 Related Design Principles and Requirements

[2] initiated a discussion on key design principles through a blog post outlining ten foundational concepts for Self-Sovereign Identity systems:

- **Existence:** Digital identities are extensions of real individuals.
- **Control:** Full authority over identities. Create, manage, revoke credentials.
- **Access:** Retrieve and review all identity data anytime.
- **Transparency:** Open and understandable operations, policies, and algorithms.
- **Persistence:** Identities remain valid as long as desired by the user.
- **Portability:** Transfer identity data between systems without restrictions.
- **Interoperability:** Compatibility across different platforms and services.
- **Consent:** Explicit permission required for sharing or using identity information.

- **Minimalization:** Minimize disclosure and share only necessary identity information.
- **Protection:** Prioritize and protect users' rights and freedoms.

These principles have since been refined, aggregated, and extended by numerous authors. [38] offer a broader formulation of nine general property categories based on these principles:

- **Representation:** Digital representation of any entity possible – human, legal, or technical.
- **Control:** Decision-making authority only by the controller, covering rights, management, ownership, and data updates.
- **Flexibility:** Avoid vendor lock-in; include interoperability, open standards, and support for open-source solutions.
- **Security:** Uses cryptographic tools to secure interactions, manage keys, and ensure tamper-proof operations.
- **Privacy:** Only essential data is disclosed in each interaction, using selective disclosure and consent.
- **Verifiability:** Ensures credential validity and timeliness through certificates, management, and revocability.
- **Authenticity:** Credentials are strongly bound to their initial bearers to prevent fraud.
- **Reliability:** Guidance for verifiers on which issuers to trust, with decentralization and governance mechanisms.
- **Usability:** Efficiency, a positive user experience and availability of user support.

[9] took another approach and composed a comprehensive list of 20 precisely formulated requirements in different categories:

- **Existence and Representation:** Independent existence of users and the creation of multiple identities.
- **Decentralization:** Avoid reliance on centralized systems.
- **Autonomy:** Full autonomy over identity data without relying on third parties.
- **Control:** Full user control over digital identities and data.
- **Privacy and Minimal Disclosure:** Use of selective disclosure and data minimization.
- **Single Source:** Users as the single source of truth, managing self-asserted and third-party claims.

- **Consent:** Users can give and withdraw consent for data usage.
- **Security:** Employ cryptographic measures for confidentiality and authentication.
- **Protection:** Safeguards entities' rights, with mechanisms for evidence and assurance.
- **Verifiability and Authenticity:** Users can prove their identity reliably and verifiers ensure data has not been tampered with.
- **Accessibility and Availability:** Unrestricted access to identity data across platforms.
- **Recoverability:** Robustness to allow identity recovery if lost or compromised.
- **Usability and User Experience:** Interfaces are intuitive and reliable, with a consistent experience.
- **Transparency:** System and algorithms are transparent to users, with open-source architecture.
- **Standardization:** Based on open standards to ensure portability and interoperability.
- **Persistence:** Identities remain as long as needed, with clear separation for modifications.
- **Portability:** Enables transfer of identity data to chosen agents.
- **Interoperability:** Usability across various domains.
- **Compatibility with Legacy Systems:** Backward compatibility to facilitate adoption.
- **Cost:** Minimal costs for identity management to encourage adoption.

[47] aim to provide a robust SSI design framework. Notably, they include functional apart from non-functional requirements, clearly differentiating between:

Non-Functional Requirements:

- **Provability:** Validate and prove credential integrity.
- **Interoperability:** Operate across platforms and domains.
- **Portability:** Transfer identity data without restrictions.
- **Pseudonymity:** Support pseudonyms for privacy.
- **Recovery:** Enable access recovery.
- **Scalability:** Handle growing users and transactions.
- **Security:** Strong cryptographic measures.
- **Usability:** User-friendly interfaces.

- **Protection:** Safeguard privacy and rights.
- **Persistence:** Identities remain as long as needed.
- **Minimization:** Limit data disclosure.
- **Existence:** Recognizes users' independent digital presence.
- **Control:** Full authority over identity data.
- **Consent:** Explicit permission for data sharing.
- **Transparency:** Open operations and policies.
- **Access:** Continuous data access.
- **Convenience:** Seamless user experience.
- **Inclusion:** Accessible to all users.
- **Trust:** Reliable and governed operations.
- **Biometrics Support:** Allows biometric authentication.
- **Support of IoT:** Compatible with IoT devices.
- **Cost:** Minimized identity management costs.

Functional Requirements:

- **Issuer Discovery:** Allow users to locate credential issuers.
- **Connection Creation:** Establish secure connections for identity interactions.
- **Credential Creation:** Generate verifiable credentials.
- **Verification with Credentials:** Allow verifiers to authenticate credentials efficiently.
- **Backup/Recovery:** Mechanisms for identity data recovery.
- **Derive/Share Credentials:** Derivation of new credentials and secure sharing.
- **Sunset Credentials:** Manage credential lifecycle, including expiration and revocation.

Depending on the use-case of an application, the expression and even presence of these requirements can vary. [22] explain that different levels of identity verification and security are required based on the specific needs and risks involved in each scenario. For example, they outline three Identity Assurance Levels ranging from minimal identity proofing to high-assurance scenarios with in-person verification.

Building upon these four publications, it is possible to categorize the different requirements into clusters. These clusters with their belonging requirements are shown in the following table 3.1.

Cluster	[2]	[38]	[9]	[47] Non-Functional Requirements	[47] Functional Requirements
Existence and Representation	Independent Existence	Representation	Existence and Representation	Existence	Credential Creation
Control and Autonomy	Control	Control	Autonomy, Single Source, Control	Control, Consent	
Access and Inclusion	Access	Flexibility	Accessibility	Access, Inclusion	Connection Creation, Issuer Discovery
Transparency	Transparency	Flexibility	Transparency	Transparency	
Privacy and Minimal Disclosure	Disclosure, Minimalization	Privacy	Privacy and Minimal Disclosure	Minimization, Pseudonymity, Consent	Derive/Share Credentials
Persistence and Portability	Persistence, Portability	Flexibility	Portability, Standards, Interoperability	Persistence, Portability, Scalability	
Interoperability and Compatibility	Interoperability	Flexibility	Interoperability, Compatibility with Legacy	Interoperability	
Consent and Delegation	Consent-based Sharing	Control	Consent	Consent	
Security and Protection	Rights Protection	Security	Protection, Security	Security, Protection, Biometrics Support	Verification with Credentials
Verifiability and Authenticity		Verifiability, Authenticity	Verifiability, Authenticity	Provability, Trust	Credential Creation, Back-up/Recovery
Reliability and Resilience		Reliability		Cost, Recovery	Backup/Recovery, Sunset Credentials
Usability and User Experience		Usability	Usability and UX	Usability, Convenience	
Adoption and Sustainability		Flexibility, Reliability		Cost, Support of IoT	

Table 3.1: Categorization of Requirements into Clusters

3.1.2 Additional Requirements for the Proposed Design

The clusters enable the formulation of additional requirements necessary to fulfill the problem statement, specific to each cluster.

Non-Functional Requirements (NFR):

- Control and Autonomy
 - **NFR1: User Control over Biometric Data:** Ensure users have full control and autonomy over their biometric data, including enrollment, storage, usage and presentation.
- Access and Inclusion
 - **NFR2: Compatibility with Standard Devices:** Ensure compatibility with standard user devices (e.g. smartphones).

- Privacy and Minimal Disclosure
 - **NFR3: No Transmission of Biometric Data:** Ensure that any biometric data remains stored on the user’s device and is never transmitted to verifiers or any external parties.
- Security and Protection
 - **NFR4: Resistance to Attacks:** Ensure that the system provides robust security measures, for example against tampering with the verification process.
- Verifiability and Authenticity
 - **NFR5: High-Assurance Verification without Disclosure:** Enable verifiers to authenticate users at high assurance levels without accessing actual biometric data.
- Reliability and Resilience & Usability and User Experience
 - **NFR6: Real-Time Processing:** Ensure real-time processing to be viable for physical, real-life authentication scenarios.

Based on these non-functional requirements it is possible to derive functional requirements, directly satisfying the respective non-functional requirements:

Functional Requirements (FR):

- User Control over Biometric Data (**NFR1**); No Transmission of Biometric Data (**NFR3**)
 - **FR1: Local Biometric Processing:** All biometric data processing should happen locally on the user’s device, so that unprocessed biometric data never leaves the device.
- High-Assurance Verification without Disclosure (**NFR5**); No Transmission of Biometric Data: (**NFR3**)
 - **FR2: Zero-Knowledge Proof Implementation:** The system should use zero-knowledge proof protocols that allow users to prove possession of biometric attributes without revealing actual biometric data.

The following table 3.2 extends the previous table 3.1 by the requirements addressing the problem statement of this work.

Cluster	[2]	[38]	[9]	[47] Non-Functional Requirements	[47] Functional Requirements	Proposed Design: Non-Functional Requirements	Proposed Design: Functional Requirements
Existence and Representation	Independent Existence	Representation	Existence and Representation	Existence	Credential Creation		
Control and Autonomy	Control	Control	Autonomy, Single Source, Control	Control, Consent		User Control over Biometric Data (NFR1)	Local Biometric Processing (FR1)
Access and Inclusion	Access	Flexibility	Accessibility	Access, Inclusion	Connection Creation, Issuer Discovery	Compatibility with Standard Devices (NFR2)	
Transparency	Transparency	Flexibility	Transparency	Transparency			
Privacy and Minimal Disclosure	Disclosure, Minimalization	Privacy	Privacy and Minimal Disclosure	Minimization, Pseudonymity, Consent	Derive/Share Credentials	No Transmission of Biometric Data (NFR3)	Local Biometric Processing (FR1), Zero-Knowledge Proof Implementation (FR2)
Persistence and Portability	Persistence, Portability	Flexibility	Portability, Standards, Interoperability	Persistence, Portability, Scalability			
Interoperability and Compatibility	Interoperability	Flexibility	Interoperability, Compatibility with Legacy	Interoperability			
Consent and Delegation	Consent-based Sharing	Control	Consent	Consent			
Security and Protection	Rights Protection	Security	Protection, Security	Security, Protection, Biometrics Support	Verification with Credentials	Resistance to Attacks (NFR4)	
Verifiability and Authenticity		Verifiability, Authenticity	Verifiability, Authenticity	Provability, Trust	Credential Creation, Back-up/Recovery	High-Assurance Verification without Disclosure (NFR5)	Zero-Knowledge Proof Implementation (FR2)
Reliability and Resilience		Reliability		Cost, Recovery	Backup/Recovery, Sunset Credentials		
Usability and User Experience		Usability	Usability and UX	Usability, Convenience		Real-Time Processing (NFR6)	
Adoption and Sustainability		Flexibility, Reliability		Cost, Support of IoT			

Table 3.2: Clustered Requirements including Proposed Design

3.2 Design: Credchain-ZKP and Credchain On-Chain ZKP

This section explores the design choices behind the two design proposals *Credchain-ZKP* and *Credchain On-Chain ZKP* and discusses them based on the non-functional and functional requirements described earlier. Following, the final design proposals are presented using user flow descriptions, selected acceptance criteria and sequence diagrams.

3.2.1 Starting Point: TCID and Credchain

To address the problem statement and the related requirements, this work considers the TCID system described in the related work. TCID, as an SSI system, aims to achieve high assurance levels in physical verification by using a PIN or a biometric template (such as a fingerprint or facial recognition) as a second authorization factor to confirm the user's ownership of the DID [40]. However, this approach requires the transmission of biometric data to the verifier, which conflicts with **NFR3**: No Transmission of Biometric Data.

The proposed design maintains the second factor principle from TCID but replaces direct biometric data transmission with a zero-knowledge proof. This satisfies **NFR1** (user retains control of data) and **NFR3** (since no unprocessed biometric leaves the device).

The second building block for this design is *Credchain*, a minimal SSI codebase providing a foundation for implementing additional features. *Credchain* is available in a public GitHub repository [10]. During implementation, the second-factor feature from TCID (referred to as *Credchain with 2FA*) and the proposed zero-knowledge proof mechanism is added to *Credchain*. Both elements together form what will be referred to as **Credchain-ZKP**. A third implementation, **Credchain On-Chain ZKP**, will additionally explore performing the verification process on the blockchain instead of performing it locally with the verifier.

3.2.2 Integrating Biometric zk-SNARK Zero-Knowledge Proofs

Credchain-ZKP uses zk-SNARK as a zero-knowledge proof since it meets the stated requirements, is available in a well-documented JavaScript library *snarkJS* [28] and present in literature. Prior research demonstrates usages of zk-SNARKs in SSI contexts as well as for isolated biometric usages, as outlined in chapter 2.2.2 and [23]. This presence in both areas reinforces the suitability of zk-SNARKs for addressing the problem statement.

While face or iris recognition would be possible, this design and implementation describes using a fingerprint template for simplicity. The same processes can be applied to other biometric input sources. Using a fingerprint also satisfies **NFR2** (compatibility with standard devices like smartphones that often use fingerprint sensors).

Using a zero-knowledge proof for biometric verification allows for privacy while maintaining high-assurance verification. It allows a user to prove ownership of a biometric input without disclosing biometrics itself. [23] proposes such a biometric identification scheme for fingerprints based on zk-SNARKs.

3.2.3 Architecture and User Flows

In this proposed design for *Credchain-ZKP*, three main phases can be described: Enrollment, zk-SNARK Proof Generation & Presentation, and Verification. Following, the user flows will be described and shown in sequence diagrams.

3.2.3.1 Enrollment

1. Local Biometric Capture

- (a) The user's device (e.g. smartphone) captures their fingerprint.
- (b) The device processes this data locally to create a biometric template, satisfying **NFR1** (User Control over Biometric Data) and **NFR3** (No Transmission of Biometric Data).

2. DID Creation

- (a) The user's device creates a DID.
- (b) The DID is a hash of
 - i. The issuer's wallet address
 - ii. A timestamp nonce (number used once, provides randomness for the hash)
 - iii. The biometric template derived previously

3. Registration on a Decentralized Network

- (a) The DID is then registered on a decentralized network. The biometric template never leaves the user's device, as required by **FR1**.

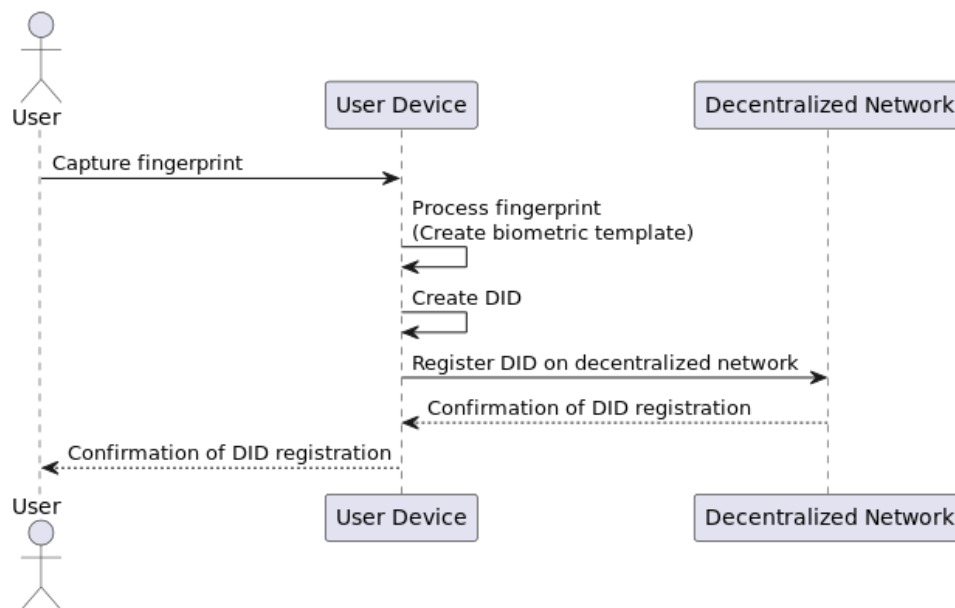


Figure 3.1: UML Sequence Diagram of Enrollment Flow

3.2.3.2 zk-SNARK Proof Generation & Presentation

1. Local ZKP Computation (**FR2**)

- (a) When users decide to present their identity, the user's device captures a fresh fingerprint scan and transforms it into a biometric template.
- (b) The biometric template is used as a private input to a zk-SNARK proof generation circuit.
- (c) Public inputs (called *publicSignals*) for this zk-SNARK proof are the issuer's wallet address, the timestamp nonce used at generation and the user's DID.

2. Transmission to the Verifier

- (a) The proof and the public inputs are then composed to one data object and shown to the user, to be transmitted to the verifier.
- (b) Effective transmission can take place locally via a wide variety of options, QR codes or direct transmission via NFC / Bluetooth would be possibilities.

As per **NFR6** (Real-Time Processing), the proof generation must be fast enough for real-life interactions. [40] mentions a maximum interaction time of 30 seconds for a system to be applicable in practice.

This proof attests that the user's fresh scan matches the biometric template attached to the biometric template without disclosing it. This aligns with **NFR3** (No Transmission of Biometric Data) and **NFR5** (High-Assurance Verification).

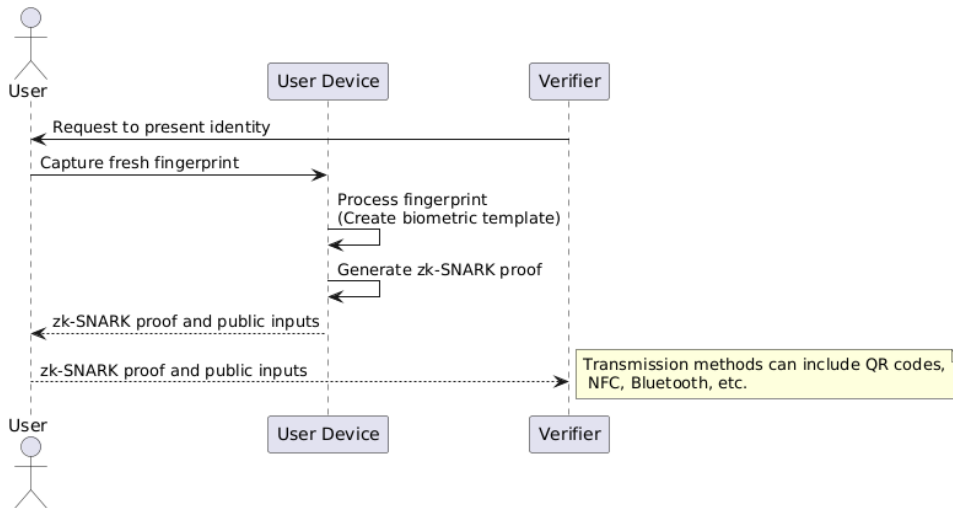


Figure 3.2: UML Sequence Diagram of zk-SNARK Proof Generation

3.2.3.3 Verification – Credchain-ZKP

1. The Verifier receives the following verification inputs:
 - (a) The Proof
 - (b) *PublicSignals* (containing issuer wallet address, timestamp nonce and a user-provided DID)
 - (c) Holder address, indicating which DID the user owns
2. Zero-Knowledge Proof Verification
 - (a) The public inputs are re-composed, ensuring that values such as the issuer and the DID come from trusted sources (on-chain or from known constant values) – no wrong or malicious inputs from the user can be used for the verification.
 - (b) In **Credchain-ZKP**: the verifier runs a verification method provided by *snarkJS* requiring three inputs:
 - i. The verification key, a public key generated during setup of the zero-knowledge circuit
 - ii. The recomposed *publicSignals*
 - iii. The proof received from the user
3. A positive result of the verification method allows the verifier to conclude that the holder indeed controls the same biometric template that was used to create the DID.

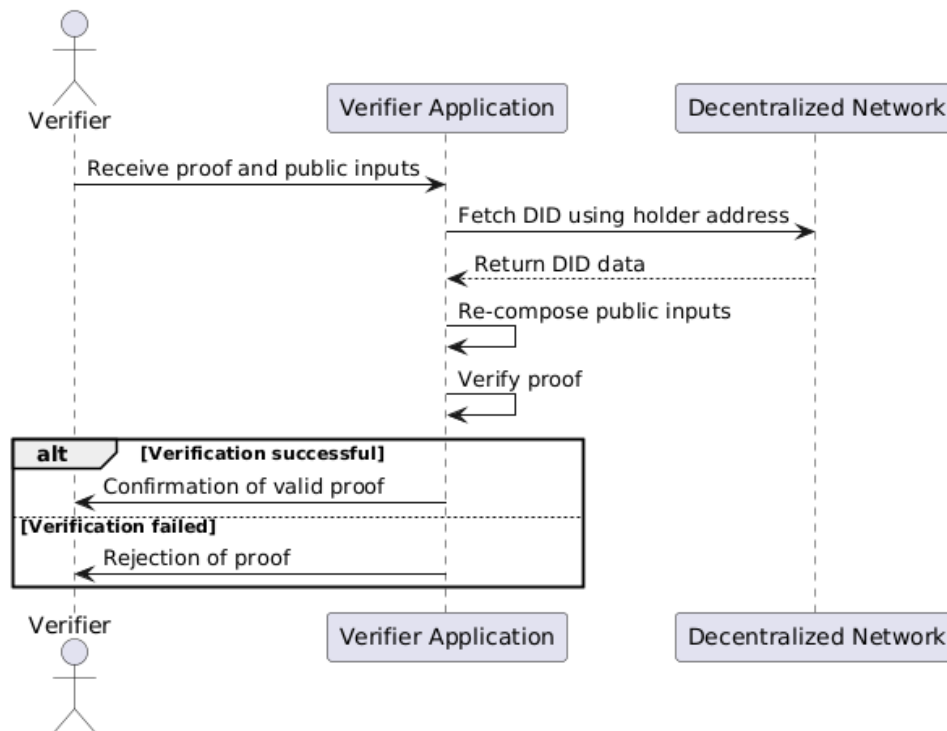


Figure 3.3: UML Sequence Diagram of Credchain-ZKP Verification Flow

3.2.3.4 Verification – Credchain On-Chain ZKP

1. The Verifier receives the following verification inputs:
 - (a) The Proof
 - (b) The *publicSignals* (containing issuer wallet address, timestamp nonce and a user-provided DID)
 - (c) Holder address, indicating which DID the user owns
2. Zero-Knowledge Proof Verification
 - (a) The public inputs are re-composed, ensuring that values such as the issuer and the DID come from trusted sources (on-chain or from known constant values) – no wrong or malicious inputs from the user can be used for the verification.
 - (b) For **Credchain On-Chain ZKP**, the verifier:
 - i. Recomposes the proof by calling a method *exportSolidityCallData* provided by the *snarkJS* library.
 - ii. Calls a method provided by a deployed Smart Contract, whose code was generated by *snarkJS* based on the circuit-specific verification key. Thus, the call to the Smart Contract requires two inputs:
 - A. The recomposed *publicSignals*
 - B. The proof received from the user
3. A positive result of the verification method allows the verifier to conclude that the holder indeed controls the same biometric template that was used to create the DID.

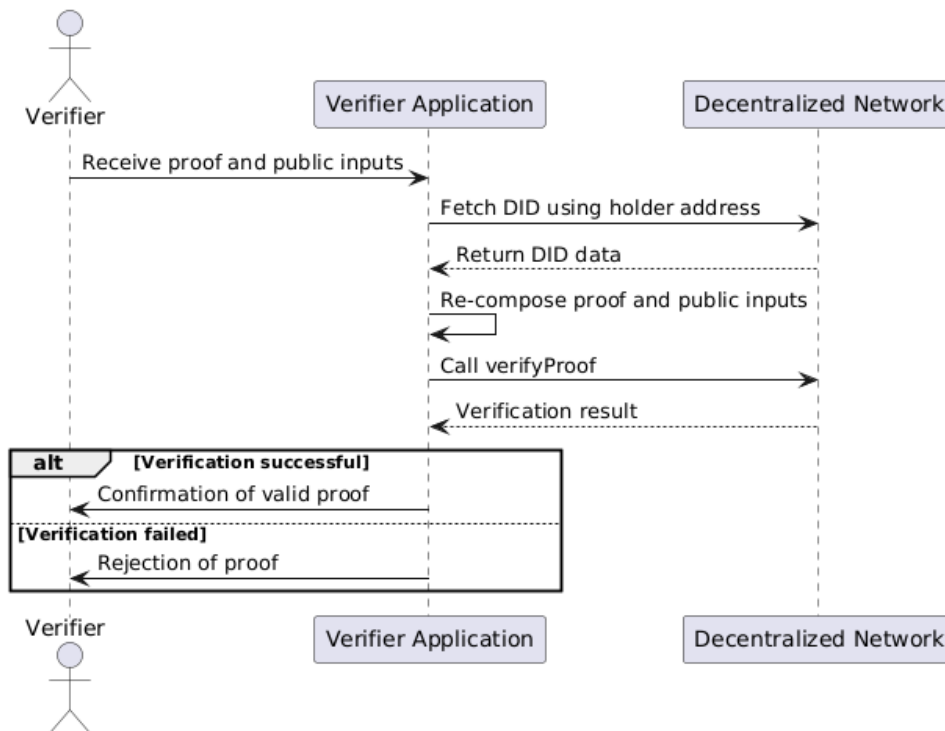


Figure 3.4: UML Sequence Diagram of Credchain On-Chain ZKP Verification Flow

3.2.3.5 Verifiable Credentials

The research gap and the problem statement focus solely to clearly determine the holder of a DID in a physical context. Verifiable credentials, as independent entities, are only linked to a DID. For this reason, they are not considered more specifically here, not in the design nor in the implementation.

3.2.4 Key Components

The following core elements are used and interacting with each other in this design:

3.2.4.1 User Device

The user's smartphone serves as the local environment for biometric capture and zero-knowledge proof generation. Its responsibilities include:

- Local Biometric Processing: Capturing the fingerprint and transforming it into a private template, during DID generation as well as during presentation and proof generation.
- DID Creation: Constructing a DID and sending it to the decentralized network.
- zk-SNARK Proof Generation: Generating the proof confirming user's biometric matches the on-chain DID using *snarkJS*.
- Proof Presentation: Sending the zero-knowledge proof and necessary *publicSignals* to the verifier on request.

3.2.4.2 Verifier

The verifier confirms the authenticity of the user's proof and ensures it matches the DID on the network.

- Retrieving DID: Fetches the DID from the network and compares user inputs with on-chain content.
- For **Credchain-ZKP**: Proof Verification: Executes a check of the zero-knowledge proof through the library *snarkJS*.

3.2.4.3 Decentralized Network

The decentralized network records DIDs and allows lookup by verifiers. Its responsibilities include:

- DID Registration: Allows for registration of the user-generated DID.
- DID Lookup: Allows a verifier to retrieve the user's DID by the user's wallet address.
- For *Credchain On-Chain ZKP*: Proof Verification: The decentralized network also contains a smart contract generated by *snarkJS* with the ability of verifying zero-knowledge proofs.

3.2.5 Addressing Non-Functional and Functional Requirements

This subsection directly shows how the formulated requirements and the design are connected.

1. **NFR1:** User Control over Biometric Data
 - (a) The biometric template is stored on the user's device and never transmitted anywhere.
2. **NFR2:** Compatibility with Standard Devices
 - (a) Enrolling and proof generation is possible on standard user's devices, requiring a smartphone with a fingerprint sensor and an internet connection at enrollment.
 - (b) Verification is also possible on standard devices. There is no need for additional or proprietary hardware. For *Credchain On-Chain ZKP*, an active internet connection is required.
3. **NFR3:** No Transmission of Biometric Data
 - (a) The biometric template is stored on the user's device and never transmitted anywhere.
4. **NFR4:** Resistance to Attacks
 - (a) Using pre-existing, stable zero-knowledge proof libraries ensures proven and reliable encryption and verification mechanisms.
 - (b) The zero-knowledge proof will fail if modified during transmission, protecting against Man-In-The-Middle Attacks.
 - (c) The Verifier compares the DID provided with the DID on-chain, ensuring no false or malicious inputs from the user's device.
5. **NFR5:** High-Assurance Verification without Disclosure

- (a) The verifier gains certainty that the user's biometric template matches the stored template without seeing biometric data.
- (b) The zero-knowledge proof circuit contains the definition of the proof. A valid proof implies that the user has knowledge of the biometric input matching the hash-based DID (hash of issuer address + timestamp + biometric template).

6. **NFR6:** Real-Time Processing

- (a) Using proven algorithms within the library *snarkJS* ensures efficient processing. Chapter 5 measures the efficiency and real-time execution time of the implementations.

7. **FR1:** Local Biometric Processing

- (a) The biometric template is stored on the user's device and never transmitted anywhere.

8. **FR2:** Zero-Knowledge Proof Implementation

- (a) Using zk-SNARK and *snarkJS* to generate and verify proofs.

Chapter 4

Implementation

This chapter explains the implementation of *Credchain with 2FA*, *Credchain-ZKP* and *Credchain On-Chain ZKP*, focusing on elements added to *Credchain*. *Credchain with 2FA* serves as a baseline with integration of two-factor authentication using direct biometric input. *Credchain-ZKP* adds the zero-knowledge proof, and *Credchain On-Chain ZKP* moves verification on-chain. The intention behind three implementations as separate projects is to enable measurement of the overhead added by either zero-knowledge proof or verification on-chain. The goal is to implement all three solutions as similarly as possible to facilitate their comparison.

4.1 Credchain with 2FA

Per design, this implementation is based on a second factor as seen in TCID [40] supporting physical authentication. The implementation is available in a public GitHub repository [24].

The main implementation is located in *frontend/src/App.js*. The application logic as well as the user interface (UI) is divided into two parts: *registerIdentity* and *verifyDID*.

The following figures 4.1 and 4.2 show the UI of *Credchain with 2FA*, before and after successful verification.

User

Generate DID

Log Output:

Verifier

Holder Address:

Issuer Address:

Timestamp:

Biometric Template:

Verify DID

Verification result:
Execution time: ms

Figure 4.1: *Credchain with 2FA*: Before Verification

User

Generate DID

Log Output:

Holder is: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8

Issuer is: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC

Date is: 578

Biometric template Mock is:
a24f98a8b2c2ffcf6d7777e73ebe756f7e944316056ef5afbe347a3437d760761c3b6b
b70b6a43ae09a7a56b623b67d251d9d8f62ac5df73275e5e140afa4afbc3cdd8517b5a
bd600ac9421a11b39780cec000b82b23ae1af9f71262baf3fedeac24a7f3b7c7c5e81d
2bb46002c4a2cfee775b1c650b4d3b365fbb3ecd9727c3d26188604c03a12ac6f1552d
2342f9356b9fbec6c6c9bde85d900e243b92a1445da3401ce42a5db8168a75953ae44c
3256b1ef73509fc1d264bbb9fd37fb8af730f8600b576bcbf1f1cfd766d4ee8dcf1bfb
46ade5474c053d4f9298105c7740a4906532640c00c3b17987ec129d2ffe6b6fb34aea
851eb8e01d956e1af78e0

DID is:
1345868014228099280424887318130162279888252630897428753191188267622136
3551398

Verifier

Holder Address:

Issuer Address:

Timestamp:

Biometric Template:

Verify DID

Verification result: Verification successful. Template and DID match.
Execution time: 5.80 ms

Figure 4.2: *Credchain with 2FA*: After Verification

4.1.1 RegisterIdentity

Clicking a Button labeled "generate DID" on the UI calls the method *registerIdentity*. *RegisterIdentity* prepares all inputs as outlined:

- Issuer (*issuerToDecimalString*): The issuer's wallet address is converted into a decimal number format *BigInt* and then formatted as a string. This is required because the *snarkJS* library only accepts inputs in this specific format for its public Input objects.
- Timestamp Nonce (*nowDecimalString*): A timestamp nonce is used to add randomness in the generated hash.
- Mock of Biometric Data (*biometricMockString*): The focus of the implementation is using and evaluating the zero-knowledge proof. Thus, the implementation uses a 512 character hexadecimal string mimicking a fingerprint template, without retrieving and using actual biometric data.

These three inputs are being hashed and saved on a smart contract *DIDRegistry*. The smart contract remains unchanged in this implementation. Last, the generated data is displayed to the user.

Following is the main code of *registerIdentity*, partly abbreviated for readability:

```

1  let holder = "0x70997970C51812dc3A010C7d01b50e0d17dc79C8";
2  let issuer = "0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC";
3  let issuerToDecimalString = BigInt(issuer.toLowerCase()).toString(10);
4  let nowDecimalString = new Date().getMilliseconds().toString();
5
6  let biometricMockString = "a24f98a8b2c2ffcf6d7777e73ebe756f7 [...]"
7  let biometricDecimalString = BigInt("0x" + biometricMockString).
  toString(10);
8
9  const ubaasDID = poseidon3([biometricDecimalString,
  issuerToDecimalString, nowDecimalString], 1).toString();
10
11 await didContract.methods.register(holder, ubaasDID).send({from:
  holder});
12 const did = await didContract.methods.getInfo(holder).call();
13
14 const didData = `Holder is: ${holder}\n\nIssuer is: ${issuer}\n\nDate
  is: ${nowDecimalString}\n\nBiometric template Mock is: ${
  biometricMockString}\n\nDID is: ${did[0]}`;
15 setDidOutput(didData);

```

Listing 4.1: Code of *RegisterIdentity*

4.1.2 VerifyDID

This method is called after inputting a holder's wallet address, the issuer's wallet address, the timestamp nonce used at generation and a biometric template. The verification process hashes the biometric template, the timestamp nonce and the issuer's wallet address with the same hash algorithm used at generation. This recomputedDID is compared to the actual DID linked to the holder's wallet address. Time markers encapsulate the verification process to measure its execution time.

Following is the main code of verifyDID, partly abbreviated for readability:

```

1      const chainDIDContract = await didContract.methods.getInfo(
holderAddress).call();
2      const chainDID = chainDIDContract[0];
3
4      const biometricDecimalString = BigInt("0x" + mockString).toString
(10);
5      const issuerToDecimalString = BigInt(issuerInput.toLowerCase()).
toString(10);
6
7      const recomputedDID = poseidon3([biometricDecimalString,
issuerToDecimalString, timestampInput], 1).toString();
8
9      // Comparison
10     if (recomputedDID === chainDID) {
11         setVerificationResult("Verification successful. Template and DID
match.");
12     } else {
13         setVerificationResult("Verification failed. Invalid inputs.");
14     }

```

Listing 4.2: Code of *VerifyDID* in *Credchain with 2FA*

4.2 Credchain-ZKP

Credchain-ZKP is based on the implementation *Credchain with 2FA* and aims at additionally integrating the zero-knowledge proof. The following sections explain the five main components of the implementation. The implementation is available in a public GitHub repository [26]. The following figures 4.3, 4.4 and 4.5 show the UI of *Credchain ZKP*, before, after successful and after failed verification.

User

Generate DID

Proof Generation

Holder Address: Enter Holder Address input

Issuer: Enter Issuer input

Timestamp: Enter Timestamp input

Biometric Input: Enter biometric mock input

Generate Proof

Generated Proof + Public Signals:

Verifier

Holder Address: Enter Holder Address input

Paste Proof Input including Public Signals:

Verify Proof

Verification result:
Execution time: ms

Figure 4.3: *Credchain-ZKP*: Before Verification

User

Generate DID

Holder is: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8

Issuer is: 0x3C44CdD86a900fa2b585dd299e03d12FA4293BC

Date is: 92

Biometric template Mock is:

a24f98a8b2c2ffc6d777e73ebe756f7e944316056ef5afbe347a3437d760761c3b6b
b70b6a43ae99a7a56b623b67d251d9d8f62ac5d73275e5e140afaf4a3cdd8517b5a
bd600ac9421a11b39780cec000b82b23ae1af9f71262baf3fedec24a7f3b7c7c5e81d
2bb46002c4a2cfee775b1c650b4d3b365fbb3ecd9727c3d26188604c03a12ac6f1552d
2342f9356b9fbec6cbc9bde85d900e243b92a1445da3401ce42a5db8168a75953ae44c
3256b1ef73509f1d264bbb9fd37fbaf730f8600b576bcbf1f1cfd766d4ee8dcf1fbf
46ade5474c053d4f9298105c7740a4906532640c0c3b17987ec129d2ffe6b6fb34aea
851eb8e601d956e1af78e0

DID is:
1712741872730316655117878884264489017164909591344202087666697695990813
0178171

Verifier

Holder Address: 0x70997970C51812dc3A01

Paste Proof Input including Public Signals:

```
{
  "proof": {
    "pi_a": [
      "22636244486907627189072474014247339136261691633067855334136167023453",
      "69881515",
      "14599429104711902528923609925203121843474900775728072803677",
      "890086068611454733",
      "1"
    ],
    "pi_b": [
      "2008903586102714476818608468924376131266180598669916814680281306854",
      "2955978953",
      "183125079773797800940119662459417815434197181806144106687",
      "5072112153472671679"
    ],
    "pi_c": [
      "29428801376773301176604741644883285913440367155695434198385647558593",
      "55747925",
      "97915170512952265700023121378834612813245450111664864106680",
      "13404157770385366"
    ],
    "pi_d": [
      "50212321008286368668211634382529332766278477939614237884183018605760",
      "76193837",
      "80339686388686424375019828673919762554902126073010571896875",
      "68321412438794665",
      "1"
    ],
    "protocol": "groth16",
    "curve": "bn128",
    "publicSignals": [
      "344073830386746567427978432078835137280280269756",
      "92",
      "17127418727303166551178788842644890171649095913442020876666976959908130178171"
    ]
  }
}
```

Verify Proof

Verification result: Proof is valid.

Execution time: 27.40 ms

Proof Generation

Holder Address: 0x70997970C51812dc3A01

Issuer: 0x3C44CdD86a900fa2b585dd299e03d12FA4293BC

Timestamp: 92

Biometric Input: a24f98a8b2c2ffc6d777e73ebe756f7e944316056ef5afbe347a3437d760761c3b6b

Generate Proof

Generated Proof + Public Signals:

```
{
  "proof": {
    "pi_a": [
      "22636244486907627189072474014247339136261691633067855334136167023453",
      "69881515",
      "14599429104711902528923609925203121843474900775728072803677",
      "890086068611454733",
      "1"
    ],
    "pi_b": [
      "2008903586102714476818608468924376131266180598669916814680281306854",
      "2955978953",
      "183125079773797800940119662459417815434197181806144106687",
      "5072112153472671679"
    ],
    "pi_c": [
      "29428801376773301176604741644883285913440367155695434198385647558593",
      "55747925",
      "97915170512952265700023121378834612813245450111664864106680",
      "13404157770385366"
    ],
    "pi_d": [
      "50212321008286368668211634382529332766278477939614237884183018605760",
      "76193837",
      "80339686388686424375019828673919762554902126073010571896875",
      "68321412438794665",
      "1"
    ],
    "protocol": "groth16",
    "curve": "bn128",
    "publicSignals": [
      "344073830386746567427978432078835137280280269756",
      "92",
      "17127418727303166551178788842644890171649095913442020876666976959908130178171"
    ]
  }
}
```

Generation time: 258.20 ms

Figure 4.4: Credchain-ZKP: After Successful Verification

User

Holder is: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
 Issuer is: 0x3C44CdDd86a900fa2b585dd299e03d12FA4293BC
 Date is: 578
 Biometric template Mock is:
 a24f98a8b2c2ffcf6d7777e73ebe756f7e944316056ef5afbe347a3437d760761c3b6b
 b70b6a43ae09a7a56b623b67d251d9d8f62ac5df73275e5e140afa4afbc3cdd8517b5a
 bd600ac9421a11b39780cec000b82b23ae1af9f71262baf3fedea24a7f3b7c7c5e81d
 2bb46002c4a2cfee775b1c650b4d3b365fbb3ecd9727c3d26188604c03a12ac6f1552d
 2342f9356b9fbec6bc9bde85d900e243b92a1445da3401ce42a5db8168a75953ae44c
 3256b1ef73509fc1d264bbb9fd37fb8af730f8600b576bcbf1f1cfd766d4ee8dcf1bfb
 46ade5474c053d4f9298105c7740a4906532640c00c3b17987ec129d2ffe6b6fb34aea
 851eb8e601d956e1af78e0
 DID is:
 1345868014228099280424887318130162279888252630897428753191188267622136
 3551398

Proof Generation

Holder Address:

Issuer:

Timestamp:

Biometric Input:

Generated Proof + Public Signals:


```
{
  "proof": {
    "pi_a": [
      "91734341591673673665022404995497727574798034455893077679266717234862",
      "26167041",
      "11608628624736114767926111505576156724729585884215335240377",
      "064716688084771918",
      "1",
      "pi_b": [
        "1160603854140850719404427187075161003760592248236200311560242030981",
        "3250437887",
        "664270931487592836187436224478767405147354335233598019495",
        "0127046856651507688"
      ],
      "14419305478731802274730704954128777330671813078256678198384080350215",
      "283377238",
      "1606501827741231909464321504092750023755790491924265895218",
      "0483850793260780267",
      "1",
      "0",
      "pi_c": [
        "19426718530157555759317642300485684799935906898460783410730767088023",
        "97318563",
        "XXX61796297008124698150926593959559986175169651771152026737",
        "3",
        "1",
        "protocol": "groth16",
        "curve": "bn128",
        "publicSignals": [
          "344073830386746567427978432078835137280280269756",
          "578",
          "13458680142",
          "280992804248873181301622798882526308974287531911882676221363551398"
        ]
      ]
    }
  }
}
```

Generation time: 332.70 ms

Verifier

Holder Address:

Paste Proof Input including Public Signals:


```
{
  "proof": {
    "pi_a": [
      "91734341591673673665022404995497727574798034455893077679266717234862",
      "26167041",
      "11608628624736114767926111505576156724729585884215335240377",
      "064716688084771918",
      "1",
      "pi_b": [
        "1160603854140850719404427187075161003760592248236200311560242030981",
        "3250437887",
        "664270931487592836187436224478767405147354335233598019495",
        "0127046856651507688"
      ],
      "14419305478731802274730704954128777330671813078256678198384080350215",
      "283377238",
      "1606501827741231909464321504092750023755790491924265895218",
      "0483850793260780267",
      "1",
      "0",
      "pi_c": [
        "19426718530157555759317642300485684799935906898460783410730767088023",
        "97318563",
        "XXX61796297008124698150926593959559986175169651771152026737",
        "3",
        "1",
        "protocol": "groth16",
        "curve": "bn128",
        "publicSignals": [
          "344073830386746567427978432078835137280280269756",
          "578",
          "13458680142",
          "280992804248873181301622798882526308974287531911882676221363551398"
        ]
      ]
    }
  }
}
```

Verification result: Invalid proof.
 Execution time: 12.40 ms

Figure 4.5: Credchain-ZKP: After Failed Verification

4.2.1 RegisterIdentity

This method remains unchanged to *Credchain with 2FA* and is therefore not explained further.

4.2.2 GenerateProof

GenerateProof is called after providing four key inputs: The holder's wallet address, their biometric template, the issuer's wallet address, and a timestamp nonce. Internally, the method retrieves the user's DID from the blockchain and constructs a zero-knowledge proof demonstrating that the provided biometric input aligns with the DID. The zero-knowledge proof takes three essential inputs:

- *circuitInput*: Includes the biometric template, the issuer's wallet address, timestamp nonce used during DID generation and the DID. These values are required by the designed circuit.
- *circuitWasm*: The WebAssembly file that executes the proof generation logic, generated based on the designed circuit
- *circuitFinalZKey*: The final private key generated during the trusted setup.

The proof is then generated using *snarkjs.groth16.fullProve*, a method provided by the *snarkJS* library. This method combines the local inputs with the circuit artifacts to produce the proof and the *publicSignals*, a small set of visible outputs, necessary to validate the proof. The output of the *fullProve* method are the generated proof and the *publicSignals*, both objects then need to be transmitted to a verifier. Once the proof is generated, the method calculates its execution time.

Following is the main code of *generateProof*, partly abbreviated for readability:

```

1      const chainDIDContract = await didContract.methods.getInfo(
generateHolderAddress).call();
2      const chainDID = chainDIDContract[0];
3      const circuitInput = {
4          biometricTemplate: BigInt("0x" + generateBiometricInput).toString
(10),
5          issuer: BigInt(generateIssuer.toLowerCase()).toString(10),
6          now: generateTimestamp,
7          DID: chainDID
8      };
9      const circuitWasmBuffer = await fetch("/zkFiles/circuit.wasm").then(
res => res.arrayBuffer());
10     const circuitWasm = new Uint8Array(circuitWasmBuffer);
11     const circuitFinalZkeyBuffer = await fetch("/zkFiles/circuit_final.
zkey").then(res => res.arrayBuffer());
12     const circuitFinalZkey = new Uint8Array(circuitFinalZkeyBuffer);
13
14     const {proof, publicSignals} = await snarkjs.groth16.fullProve(
15         circuitInput,
16         circuitWasm,
17         circuitFinalZkey
18     );
19     const proofData = {proof, publicSignals};
20     setGeneratedProof(JSON.stringify(proofData));

```

Listing 4.3: Code of *GenerateProof*

4.2.3 VerifyDID

Different to *Credchain with 2FA*'s `verifyDID`, this implementation's `verifyDID` requires the proof and *publicSignals* as one JSON object and, as a second input, the holder's wallet address.

The verification process recomposes *publicSignals* to prevent any user manipulation. Then, a call to the method `snarkjs.groth16.verify` follows to perform the validation based on the verification key (the public key to the zero-knowledge proof circuit), the recomposed *publicSignals* and the proof.

Finally, the method calculates its execution date and outputs it together with the result of the proof validation.

Following is the main code of *verifyDID*, partly abbreviated for readability:

```

1      const parsedProof = JSON.parse(proofInput);
2      const {proof, publicSignals} = parsedProof;
3
4      // compose data to recompose Public Signals
5      const chainDIDContract = await didContract.methods.getInfo(
verifierHolderAddress).call();
6      const chainDID = chainDIDContract[0];
7
8      let issuer = "0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC";
9      let issuerToDecimalString = BigInt(issuer.toLowerCase()).toString
(10);
10
11     const recomposedPublicSignals =
12     [
13         publicSignals[0],
14         issuerToDecimalString,
15         publicSignals[2],
16         chainDID
17     ];
18
19     if (JSON.stringify(recomposedPublicSignals) !== JSON.stringify(
publicSignals)) {
20         throw new Error("publicInputs mismatch! Proof invalid!");
21     }
22
23     const isValid = await snarkjs.groth16.verify(verificationKey,
recomposedPublicSignals, proof);
24     if (isValid) {
25         setVerificationResult("Proof is valid.");
26     } else {
27         setVerificationResult("Invalid proof.");
28     }

```

Listing 4.4: Code of *VerifyDID* in *Credchain-ZKP*

4.2.4 Circom Circuit

The library used for generation and verification of the zero-knowledge proof, *snarkJS* [28], relies on Circom 2 [27] to define its zero-knowledge proof circuits.

The circuit specifies precisely what information can be retrieved from a valid proof generated by *snarkJS*.

Similar to the *verifyDID* method in *Credchain with 2FA*, the circuit is designed to perform the following steps:

1. Take as inputs: Biometric template, issuer's wallet address, the timestamp nonce.
2. Hash the biometric template, the issuer's wallet address, the timestamp nonce.
3. Ensure that the hash is equal with the DID input.

The issuer's wallet address, timestamp nonce and DID serve as public inputs, while the biometric template remains a private input.

```

1 pragma circom 2.1.9;
2
3 include "./circomFiles/poseidon.circom";
4 include "./circomFiles/comparators.circom";
5
6 template DIDValidation() {
7
8     signal input biometricTemplate;
9     signal input issuer;
10    signal input now;
11    signal input DID;
12
13    component hash = Poseidon(3);
14    hash.inputs[0] <== biometricTemplate;
15    hash.inputs[1] <== issuer;
16    hash.inputs[2] <== now;
17
18    component eq = IsEqual();
19    eq.in[0] <== hash.out;
20    eq.in[1] <== DID;
21
22    // eq.out must be 1 (hash == DID)
23    eq.out === 1;
24 }
25
26 component main {public [issuer, now, DID]} = DIDValidation();

```

Listing 4.5: Code of Circom Circuit

4.2.5 Trusted Setup Generation

Generating the trusted setup is a manual process required to generate foundational cryptographic material for *Groth16*. *Groth16* is the zero-knowledge proof scheme used by *snarkJS* to generate cryptographic proofs based on the circuit. It involves a series of command-line commands and file outputs. Below is an outline of the key commands, their outputs and their purpose. The setup documentation is available online as well [27].

```

1 > circom2 circuit.circom --r1cs --wasm --sym -o
2 template instances: 73
3 non-linear constraints: 261
4 linear constraints: 0
5 public inputs: 3
6 private inputs: 1
7 public outputs: 0
8 wires: 266
9 labels: 945
10 Written successfully: ./circuit.r1cs
11 Written successfully: ./circuit.sym
12 Written successfully: ./circuit\_js/circuit.wasm
13 Everything went okay

```

Listing 4.6: Command-Line Interaction: Convert Circom Circuit into Multiple Artifacts

This step converts the Circom circuit source file (circuit.circom) into:

- Circuit constraints in a rank-1 constraint system file (circuit.r1cs)
- A compiled WebAssembly file for generating witnesses (circuit.wasm)
- Symbolic debug data (circuit.sym)

```

1 > snarkjs powersoftau new bn128 12 pot12_0000.ptau
2 [INFO] snarkjs: First Contribution Hash:
3 9e63a5f6 2b96538d aed2372 481920d1
4 a40b9195 9ea38ef9 f5f6a303 3b886516
5 0710d067 c09d0961 5f928ea5 17bcd49
6 ad75abd2 c8340b40 0e3b18e9 68b4ffef

```

Listing 4.7: Command-Line Interaction: Initialize a Powers of Tau File

This command initializes a Powers of Tau file for a specific curve. These Powers of Tau numbers ensure we have enough cryptographic randomness for the eventual *Groth16* scheme construction.

```

1
2 > \snarkjs powersoftau contribute pot12_0000.ptau pot12_0001.ptau --name
  ="First Contribution"
3 Enter a random text. (Entropy): regenschirm
4 [INFO] snarkJS: Contribution Response Hash imported:
5 b8fcd703 732e6546 4f5f04c9 b43c9bc7
6 f2c505ee cb3d3a5b 6e85458f c1a2b595
7 30dbbe34 a5f5105c e883d770 f298b7e3

```

```

8  3cd5c609 c43dcc05 af52171a e59666d8
9  [INFO] snarkJS: Next Challenge Hash:
10 b1f8a57f 4dd40aa9 1dd887d0 8cbb775d
11 dc54160e f3456e28 2168b550 397024f4
12 6216e3fe ffb1f734 7b6daa8b 5b4d7557
13 dd9fd459 e5969bc4 2ff99504 d0000a1c

```

Listing 4.8: Command-Line Interaction: Contribute to Random Entropy

This step introduces random entropy to the Powers of Tau generated previously by typing any random text.

```

1 > snarkjs powersoftau prepare phase2 pot12_0001.ptau pot12_final.ptau

```

Listing 4.9: Command-Line Interaction: Fix Random Entropy into Final File

This step fixes all prior randomness into a final file *pot12_final.ptau* required for *Groth16*'s second phase of the trusted setup.

```

1 > snarkjs groth16 setup circuit.r1cs pot12_final.ptau circuit_final.zkey
2 [INFO] snarkjs: Reading r1cs
3 [INFO] snarkjs: Reading tauG1
4 [INFO] snarkjs: Reading tauG2
5 [INFO] snarkjs: Reading alphatauG1
6 [INFO] snarkjs: Reading betatauG1
7 [INFO] snarkjs: Circuit hash:
8 795dc153 b45d0c7a a5aceb9b e08fc14a
9 2d1c912d f4c41727 cbea117 e837dece
10 5635aa93 864a9266 a2ddbcca 7b6d7013
11 b36b0db3 dd31070a 3b92b84c ec6b808d

```

Listing 4.10: Command-Line Interaction: Produce Proving and Verification Key

Next is the actual *Groth16* setup. The circuit definition (*circuit.r1cs*) and the final Powers of Tau data (*pot12_final.ptau*) are used to produce our proving and verification key in one packaged file (*circuit_final.zkey*).

```

1 > snarkjs zkey export verificationkey circuit_final.zkey
  verification_key.json
2 [INFO] snarkjs: EXPORT VERIFICATION KEY STARTED
3 [INFO] snarkjs: > Detected protocol: groth16
4 [INFO] snarkjs: EXPORT VERIFICATION KEY FINISHED

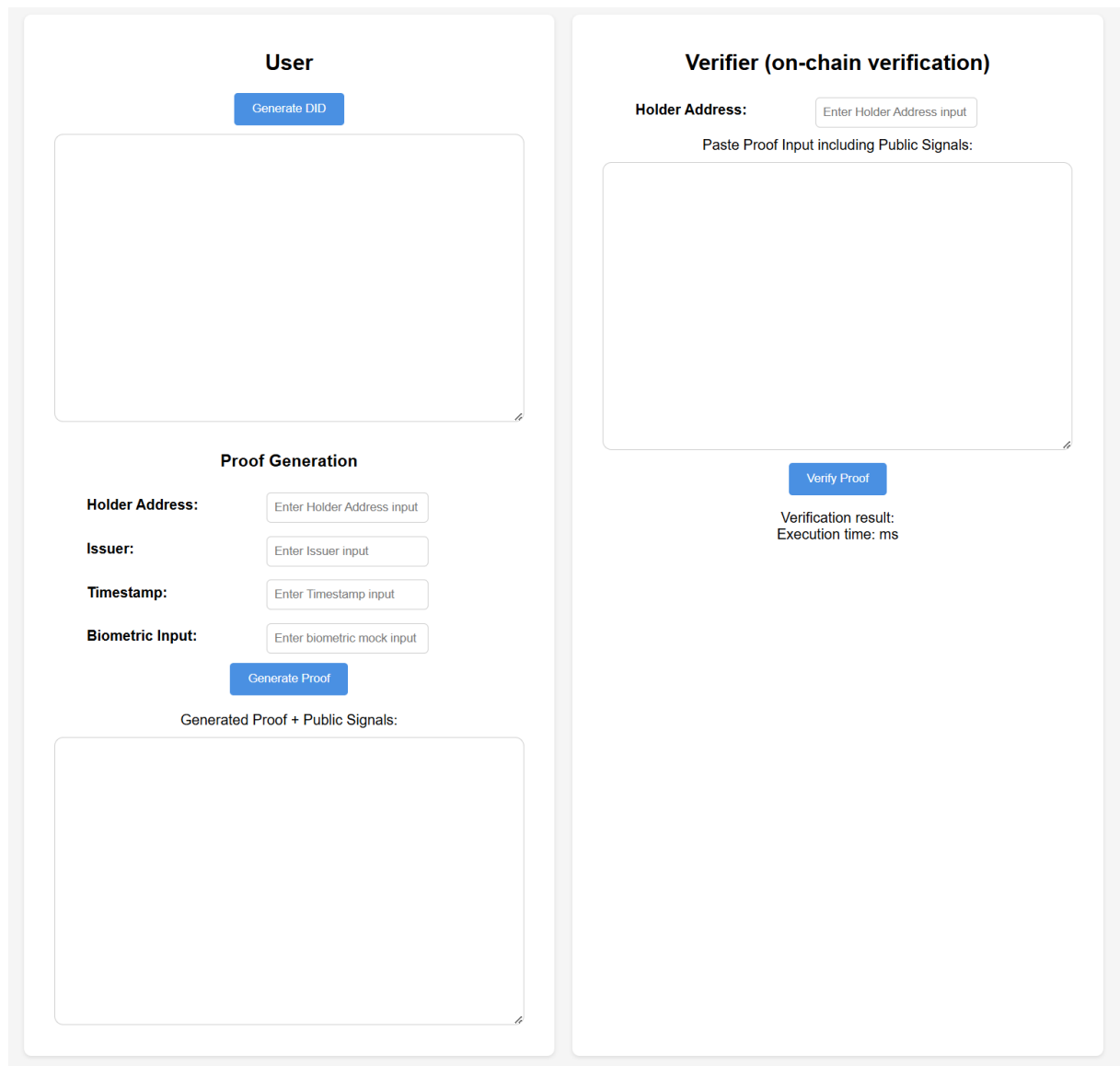
```

Listing 4.11: Command-Line Interaction: Separate Verification Key

Finally, the verification key is separated out as *verification_key.json*. Verifiers will be able to load this JSON object during the *snarkjs.groth16.verify* step to validate any proofs generated for that specific circuit.

4.3 Credchain On-Chain ZKP

This implementation is based on *Credchain-ZKP*, but moves the verification process from the verifier to the blockchain using a smart contract. The source code of this implementation is available in a public GitHub repository [25]. Figures 4.6, 4.7 and 4.8 show the UI of *Credchain On-Chain ZKP*, before, after successful and after failed verification.



The image displays a two-panel user interface for the Credchain On-Chain ZKP system. The left panel, titled "User", contains a "Generate DID" button at the top, followed by a large empty text area. Below this is a "Proof Generation" section with four input fields: "Holder Address:" (with placeholder "Enter Holder Address input"), "Issuer:" (with placeholder "Enter Issuer input"), "Timestamp:" (with placeholder "Enter Timestamp input"), and "Biometric Input:" (with placeholder "Enter biometric mock input"). A "Generate Proof" button is positioned below these fields. At the bottom of the panel is a label "Generated Proof + Public Signals:" followed by another large empty text area. The right panel, titled "Verifier (on-chain verification)", features a "Holder Address:" label with an "Enter Holder Address input" field. Below this is a label "Paste Proof Input including Public Signals:" followed by a large empty text area. At the bottom of the panel is a "Verify Proof" button, and below that, the text "Verification result:" and "Execution time: ms".

Figure 4.6: *Credchain On-Chain ZKP*: Before Verification

User

Generate DID

Holder is: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8

Issuer is: 0x3C44CdDd86a900fa2b585dd299e03d12FA4293BC

Date is: 667

Biometric template Mock is:

a24f98a8b2c2ffcf6d777e73ebe756f7e944316056ef5afbe347a3437d760761c3b6b
b70b6a43ae09a7a56b623b67d251d9d8f62ac5df73275e5e140afa4afbc3cdd8517b5a
bd600ac9421a11b39780ec000b2b23ae1af9f71262baf3fedec24a7f3b7c7c5e81d
2bb46082c4a2cfee775b1c650b4d3b365fbb3ec9727c3d26188604c03a12ac6f1552d
2342f9356b0fbec6cbc9bde85d900e243b92a1445da3401ce42a5db8168a75953ae44c
3256b1ef73509fc1d264bb9fd37fb8af730f8600b576bcbf1f1cfd766d4ee8dcf1bfb
46ade5474c053d4f9298105c7740a4906532640c00c3b17987ec129d2ffe6b6fb34aea
851eb8e601d956e1af778e0

DID is:
6926146931401628285335895194775807652138317461677832876013561909823373
258835

Proof Generation

Holder Address:

0x70997970C51812dc3A01

Issuer:

0x3C44CdDd86a900fa2b5f

Timestamp:

667

Biometric Input:

a24f98a8b2c2ffcf6d7777e7

Generate Proof

Generated Proof + Public Signals:

```
{
  "proof": {
    "pi_a": [
      "11472440258096960976799926381728385301276067263547157252936996620150",
      "612013259",
      "1302066962883823792856036133567179944096057469186228450361",
      "0370542762682733785",
      "1",
      "pi_b": [
        "7866312760886124981604065482436627618772182278308510669725214134825",
        "612662311",
        "5159077180537477170414766645353475888148823235435327594056",
        "884025104088119211",
        "53307695457304486299958814150801933386798561145239004140674109831123",
        "68368709",
        "71889138889532129356908439464073016988743104333964641798599",
        "22108281951036766",
        "1",
        "0",
        "pi_c": [
          "18656304609731106061640873201538595826006240045006073369460887759637",
          "707426656",
          "284688576733262322210969971607692483904246358819603731868",
          "739917979097675998",
          "1",
          "protocol": "groth16",
          "curve": "bn128",
          "public Signals": [
            "344073830386746567427978432078835137280280269756",
            "667",
            "69261469314",
            "01628285335895194775807652138317461677832876013561909823373258835"
          ]
        ]
      }
    ]
  }
}
```

Generation time: 379.80 ms

Verifier (on-chain verification)

Holder Address:

0x70997970C51812dc3A01

Paste Proof Input including Public Signals:

```
{
  "proof": {
    "pi_a": [
      "11472440258096960976799926381728385301276067263547157252936996620150",
      "612013259",
      "1302066962883823792856036133567179944096057469186228450361",
      "0370542762682733785",
      "1",
      "pi_b": [
        "7866312760886124981604065482436627618772182278308510669725214134825",
        "612662311",
        "5159077180537477170414766645353475888148823235435327594056",
        "884025104088119211",
        "53307695457304486299958814150801933386798561145239004140674109831123",
        "68368709",
        "71889138889532129356908439464073016988743104333964641798599",
        "22108281951036766",
        "1",
        "0",
        "pi_c": [
          "18656304609731106061640873201538595826006240045006073369460887759637",
          "707426656",
          "284688576733262322210969971607692483904246358819603731868",
          "739917979097675998",
          "1",
          "protocol": "groth16",
          "curve": "bn128",
          "public Signals": [
            "344073830386746567427978432078835137280280269756",
            "667",
            "69261469314",
            "01628285335895194775807652138317461677832876013561909823373258835"
          ]
        ]
      }
    ]
  }
}
```

Verify Proof

Verification result: Proof is valid.

Execution time: 28.50 ms

Figure 4.7: *Credchain On-Chain ZKP*: After Successful Verification

User

Generate DID

Holder is: 0x70997970C51812dc3A018C7d01b50e0d17dc79C8
Issuer is: 0x3C44CdD86a900fa2b585dd299e03d12FA4293BC
Date is: 883
Biometric template Mock is:
a24f98a8b2c2ffc6d7777e73e756f7e944316056ef5afbe347a3437d760761c3b6b
b70b6a43ae09a7a56b623b67d251d9d8f62ac5d7f3275e5e140afa4afbc3cdd8517b5a
bd600ac9421a11b39780cec000b82b23ae1af9f71262baf3fedec24a7f3b7c7c5e81d
2bb46002c4a2cfee775b1c650bd4d3b365fbb3ec9d727c3d26188604c03a12ac6f1552d
2342f9356b9fbec6bc9bde85d900e243b92a1445da3401ce42a5db8168a75953ae44c
3256b1ef73509fcd1d264bb9f3d37fb8af730f8600b576bcbf1f1cfd766d4ee8dcf1bfb
46ade5474c053d4f9298405c7740a4906532640c00c3b17987ec129d2ffe6b6fb34aea
851eb8e6010956e1af78e0
DID is:
1870731169192228672497471205840108306407210675694171982422962278500277
2133581

Proof Generation

Holder Address: 0x70997970C51812dc3A01
Issuer: 0x3C44CdD86a900fa2b585dd299e03d12FA4293BC
Timestamp: 883
Biometric Input: a24f98a8b2c2ffc6d7777e73e756f7e944316056ef5afbe347a3437d760761c3b6b
Generate Proof

Generated Proof + Public Signals:

```
{
  "proof": {
    "pi_a": [
      "83225514700166107582573096313981332802492726700817005524599958036503",
      "42406486",
      "21382524868481430559923576248992628540579857624635056590897",
      "932566933766890747",
      "1"
    ],
    "pi_b": [
      "1834054104922236557665186839664376329469477535220935457093209870704",
      "5115741483",
      "390312143032597447558415399685601599954638888830608871440",
      "6177630021302401338"
    ],
    "pi_c": [
      "18590905563661032633660301011639645210600310158191928213516371139779",
      "045200029",
      "x931782470897559001966000164519844039152112879310815695875",
      "8585673136381061381",
      "1",
      "0"
    ],
    "pi_d": [
      "13484608260294201945125619238557937077190398558850182395611039424123",
      "634901551",
      "1594235462284319761225028040277253375861439755762438929099",
      "1951346677075170166",
      "1"
    ],
    "protocol": "groth16",
    "curve": "bn128",
    "publ": "cSignals"
  }
}
```

Generation time: 354.30 ms

Verifier (on-chain verification)

Holder Address: 0x70997970C51812dc3A01

Paste Proof Input including Public Signals:

```
{
  "proof": {
    "pi_a": [
      "83225514700166107582573096313981332802492726700817005524599958036503",
      "42406486",
      "21382524868481430559923576248992628540579857624635056590897",
      "932566933766890747",
      "1"
    ],
    "pi_b": [
      "1834054104922236557665186839664376329469477535220935457093209870704",
      "5115741483",
      "390312143032597447558415399685601599954638888830608871440",
      "6177630021302401338"
    ],
    "pi_c": [
      "18590905563661032633660301011639645210600310158191928213516371139779",
      "045200029",
      "x931782470897559001966000164519844039152112879310815695875",
      "8585673136381061381",
      "1",
      "0"
    ],
    "pi_d": [
      "13484608260294201945125619238557937077190398558850182395611039424123",
      "634901551",
      "1594235462284319761225028040277253375861439755762438929099",
      "1951346677075170166",
      "1"
    ],
    "protocol": "groth16",
    "curve": "bn128",
    "publ": "cSignals"
  }
}
```

Verify Proof

Verification result: Verification failed. Check log.
Execution time: ms

Figure 4.8: Credchain On-Chain ZKP: After Failed Verification

4.3.1 VerifyDID

The method `verifyDID` differs from *Credchain-ZKP*. Instead of making a verification call to the *snarkJS* library, it calls a method on a smart contract. To accomplish this, the proof elements are converted to hexadecimal format, split, and then passed to the function call in separate parts. The hexadecimal conversion is performed using the method `snarkjs.groth16.exportSolidityCallData` from the *snarkJS* library.

The following code excerpt highlights the modifications made to the `verifyDID` method compared to *Credchain-ZKP*.

```

1
2    // transform proof and publicInput to Contract input format
3    const callData = await snarkjs.groth16.exportSolidityCallData(proof,
    recomposedPublicSignals);
4
5    const callDataArray = JSON.parse('[${callData}]');
6    const a = callDataArray[0];
7    const b = callDataArray[1];
8    const c = callDataArray[2];
9    const callDataInput = callDataArray[3];
10
11    // on-chain verification
12    let verifier = "0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC"; //
Account 3
13    const isValid = await verifierContract.methods.verifyProof(a, b, c,
    callDataInput).send({from: verifier});

```

Listing 4.12: Code of *VerifyDID* in *Credchain on-Chain ZKP*

4.3.2 Verifier.sol

This file is generated by the *snarkJS* library based on the circuit artifacts previously created during the trusted setup. It includes a single method, *verifyProof*, which accepts four inputs:

- pA
- pB
- pC
- *pubSignals*

Components pA , pB and pC are parts of a zk-SNARK proof and must be provided in a specific format. *PubSignals* expects the public inputs required for the verification. The verification is then performed using numerous constraint constants and calculations enforcing these constraints.

Generation of this file is possible using the following command:

```

1 > snarkjs zkey export solidityverifier circuit_final.zkey Verifier.sol
2 [INFO] snarkJS: EXPORT VERIFICATION KEY STARTED
3 [INFO] snarkJS: > Detected protocol: groth16
4 [INFO] snarkJS: EXPORT VERIFICATION KEY FINISHED

```

Listing 4.13: Command-Line Interaction: Generation of *Verifier.sol*

Chapter 5

Evaluation

This chapter evaluates the three implementations *Credchain with 2FA*, *Credchain-ZKP* and *Credchain OnChain ZKP* through experimental tests and theoretical analysis. The evaluation scenarios focus on proof of ownership of a DID in physical contexts, such as when an individual user visits a physical store and attempts verification. This focus aligns with non-functional requirements **NFR2** (Compatibility with Standard Devices) and **NFR6** (Real-Time Processing).

5.1 System Performance

Measured performance refers to the execution time as perceived and measured by the React user interface for the respective methods responsible for zero-knowledge proof generation and verification. In line with **NFR2** (Compatibility with Standard Devices), three devices are used for the experiments, representing realistic scenarios in which such devices could be used by either user (Smartphones) or verifier (Windows Notebook).

- OnePlus 7T, OxygenOS (Android) 12
- Apple iPhone 14, iOS 18.2
- Lenovo X1 Yoga G4, i7 8565U, Windows 11

For all tests, a locally hosted HardHat Node was started and accessed through the React application. Tests were either run locally on the Windows Notebook mentioned above or on the mobile devices, with a HardHat Node made available in the local network.

After each of the five test runs per device and implementation, both the Node and the React development server were restarted and the device's cache cleared to prevent any interference and ensure comparable results.

5.1.1 Execution Time: *GenerateProof*

The following Figure 5.1 presents the execution times of the generateProof method by device and implementation, the tables 5.1, 5.2 and 5.3 contain the associated data.

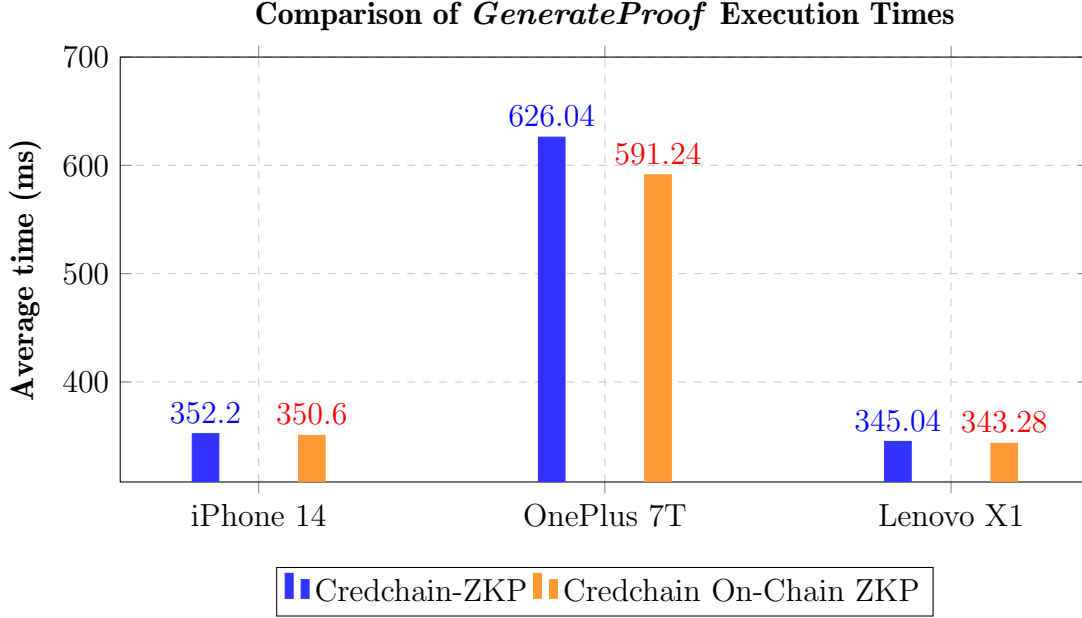


Figure 5.1: Comparison of Average GenerateProof Execution Times Across Devices.

Note: *Credchain with 2FA* has no zero-knowledge proof generation, while the generateProof method is identical for the other two implementations. For this reason it is possible to calculate a combined average across all 10 runs per device, resulting in an execution time of 351.9ms for the iPhone 14, 608.64ms for the OnePlus 7T and 344.16ms for the Lenovo X1 Yoga.

The generation of the zero-knowledge proof is the most time- and computation-intensive phase. Device performance appears to play a role, with the older OnePlus 7T notably lagging behind. However, all devices comfortably meet the evaluation scenario of real-time usage with user devices, completing the generation of a zero-knowledge proof in under one second. This satisfies both **NFR2** (Compatibility with Standard Devices) and **NFR6** (Real-Time Processing).

Test Run Nr.	1	2	3	4	5	average
<i>Credchain-ZKP</i>	352.00ms	349.00ms	357.00ms	357.00ms	351.00ms	352.20ms
<i>Credchain</i>	379.00ms	355.00ms	348.00ms	338.00ms	333.00ms	350.60ms
<i>On-Chain ZKP</i>						

Table 5.1: Execution Time of *GenerateProof*: iPhone 14

Test Run Nr.	1	2	3	4	5	average
<i>Credchain-ZKP</i>	680.00ms	604.30ms	610.40ms	608.50ms	627.00ms	626.04ms
<i>Credchain</i>	591.10ms	607.60ms	602.60ms	557.70ms	597.20ms	591.24ms
<i>On-Chain ZKP</i>						

Table 5.2: Execution Time of *GenerateProof*: OnePlus 7T

Test Run Nr.	1	2	3	4	5	average
<i>Credchain-ZKP</i>	332.70ms	338.50ms	356.80ms	350.50ms	346.70ms	345.04ms
<i>Credchain</i>	340.40ms	349.10ms	341.40ms	341.90ms	343.60ms	343.28ms
<i>On-Chain ZKP</i>						

Table 5.3: Execution Time of *GenerateProof*: Lenovo X1 Yoga G4 / i7 8565U

5.1.2 Execution Time: *VerifyDID*

The following Figure 5.2 presents the execution times of the *verifyDID* method by device and implementation, the tables 5.4, 5.5 and 5.6 contain the associated data.

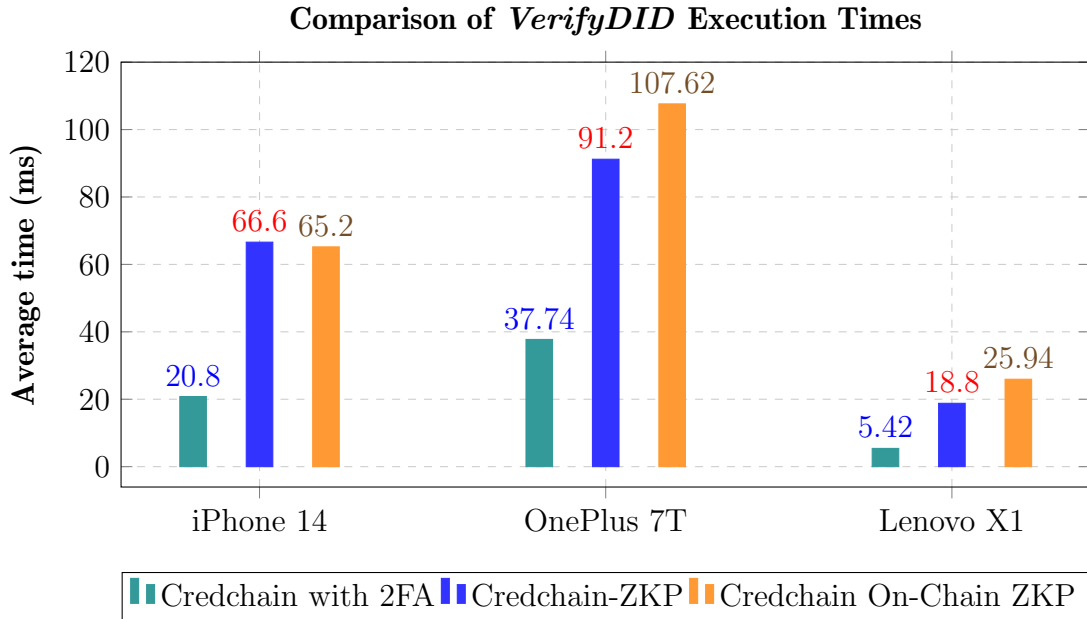


Figure 5.2: Comparison of Average VerifyDID Execution Times Across Devices.

The verification of the zero-knowledge proof in *Credchain-ZKP* adds between 14.38ms and 53.46ms of execution time compared to *Credchain with 2FA*, depending on the device. The difference measured between *Credchain-ZKP* and *Credchain On-Chain ZKP* is negligible, with a maximum of 16.42ms.

For *Credchain On-Chain ZKP*, it is important to note that these tests were done on a local HardHat Node. In a real-world scenario on the Ethereum MainNet, on-chain verification would take longer because of network delays and block confirmation times.

On chain verification could take anywhere from a few seconds to minutes. As of 5th of January 2025, the average block time on the Ethereum MainNet is around 12 seconds [46]. However, the total execution time for a transaction involves more than just the block time, such as transaction propagation across the network and time until inclusion in a block.

It can be concluded that verification with *Credchain-ZKP* meets both **NFR2** (Compatibility with Standard Devices) and **NFR6** (Real-Time Processing). In contrast, while the on-chain verification with *Credchain On-Chain ZKP* satisfies **NFR2**, as processing is independent of the user's device, it does not fulfill **NFR6**, since processing times of up to several minutes cannot be considered real-time.

Test Run Nr.	1	2	3	4	5	average
<i>Credchain 2FA</i>	18.00ms	30.00ms	16.00ms	17.00ms	23.00ms	20.80ms
<i>Credchain-ZKP</i>	64.00ms	65.00ms	67.00ms	72.00ms	65.00ms	66.60ms
<i>Credchain</i>	60.00ms	63.00ms	70.00ms	65.00ms	68.00ms	65.20ms
<i>On-Chain ZKP</i>						

Table 5.4: Execution Time of *VerifyDID*: iPhone 14

Test Run Nr.	1	2	3	4	5	average
<i>Credchain 2FA</i>	40.70ms	36.40ms	37.70ms	39.10ms	34.80ms	37.74ms
<i>Credchain-ZKP</i>	96.10ms	103.70ms	85.80ms	81.50ms	88.90ms	91.20ms
<i>Credchain</i>	92.70ms	109.10ms	124.80ms	95.70ms	115.80ms	107.62ms
<i>On-Chain ZKP</i>						

Table 5.5: Execution Time of *VerifyDID*: OnePlus 7T

Test Run Nr.	1	2	3	4	5	average
<i>Credchain 2FA</i>	5.80ms	5.00ms	4.20ms	5.10ms	7.70ms	5.42ms
<i>Credchain-ZKP</i>	12.40ms	21.60ms	19.90ms	19.70ms	20.40ms	18.80ms
<i>Credchain</i>	25.40ms	23.80ms	33.20ms	24.10ms	23.20ms	25.94ms
<i>On-Chain ZKP</i>						

Table 5.6: Execution Time of *VerifyDID*: Lenovo X1 Yoga G4 / i7 8565U

5.2 Cost

The cost of blockchain transactions is determined by gas fees, which represent the computational resources required to process and validate the transaction. The Hardhat Node outputs logs containing "Gas used" for any deployment or computation done on-chain. Average gas price per unit recorded as of 5th of January 2025 is 8 Gwei [11], however, it can fluctuate heavily [33]. 8 Gwei equate to 0.000000008 ETH. With a Price of 3,632.82 USD per ETH, as of 5th of January 2025 07:00 GMT+1 [8], a gas unit consumed translates to cost of 0.00002906256 USD per gas unit.

The following subsections will calculate the cost associated with the deployment of the contract as well as transaction cost, using the above value per gas unit.

These calculations, summarized in Table 5.7, indicate that zero-knowledge proof verification on-chain introduces fix-cost of USD 12.36 at deployment and per-call cost of USD 6.64 for each verification in comparison to local verification.

Implementation	Deployment (Gas)	Deployment (USD)	Interactions (Gas)	Interactions (USD)
<i>Credchain 2FA</i>	8,078,173	234.77	189,728	5.51
<i>Credchain-ZKP</i>	8,078,173	234.77	189,716	5.51
<i>Credchain</i>	8,928,977	259.50	418,326	12.51
<i>On-Chain ZKP</i>				
additional cost	425,390	12.36	228,610	6.64
<i>On-Chain ZKP</i>				

Table 5.7: Summary of Gas Units and Cost for the Associated Smart Contracts

5.2.1 Credchain with 2FA

Deploying the contracts required a total of 8,078,173 gas units, resulting in deployment cost of USD 234.77. A single call to register a DID consumed 189,728 gas units, equivalent to USD 5.51.

Since the contracts have not been modified from the original *Credchain* implementation, no additional cost occurs for *Credchain with 2FA* in that aspect. Only the cost of the call to the DIDRegistry is specific to this implementation since the structure of the DID has been modified.

5.2.2 Credchain-ZKP

Deploying the contracts required a total of 8,078,173 gas units, resulting in deployment cost of USD 234.77. A single call to register a DID consumed 189,716 gas units, equivalent to USD 5.51.

Since the contract and registration code are identical, the resulting cost are identical to *Credchain with 2FA*. The registration call used slightly fewer gas units, likely due to differences in the size of the DID components. However, when rounded to USD, the cost is effectively the same.

Credchain-ZKP incurs no additional gas costs compared to *Credchain with 2FA* since both proof generation and verification occur locally.

5.2.3 Credchain On-Chain ZKP

For *Credchain On-Chain ZKP*, deploying the contracts required a total of 8,503,587 gas units, resulting in deployment cost of USD 247.14. This includes the additional deployment of the *Groth16Verifier* contract, which used 425,390 gas units, adding USD 12.36 to the total cost.

The registration of a DID consumed 189,716 gas units, equivalent to USD 5.51 per call, identical to *Credchain-ZKP*. A single call to the *Groth16Verifier* used 228,610 gas units, translating to cost of USD 6.64.

5.3 Scalability

In this subsection, scalability is considered as the ability to handle an increasing number of users and transactions, while highlighting limiting factors for possible growth under potential users and verifiers.

For *Credchain-ZKP*, both proof generation and verification happen locally. Since there is no specific hardware needed and everything is processed in real time, even on older mobile phones such as the OnePlus 7T released in 2019 [35], there are no central nor decentral processing time constraints compared to the base system *Credchain with 2FA*. Local processing avoids bottlenecks that might occur in decentralized networks. This setup meets the requirements **NFR2**: Compatibility with Standard Devices and **NFR6**: Real-Time Processing. Cost-wise, there are no increases in comparison to the base system.

For *Credchain On-Chain ZKP*, the scalability of the verification process is tied to the underlying blockchain. Factors such as transaction throughput, block confirmation times and network congestion can directly impact verification times. Especially with high demand in the network, possible delays are likely to increase. Additionally, gas fees associated with on-chain calls can fluctuate as well [33]. Elevated gas fees and non-real-time processing could discourage potential users and verifiers.

One unresolved aspect is the systems' reliance on a stable biometric template. [23] has proven that zk-SNARKs are fundamentally compatible with fingerprints. However, how this pattern's performance in real-world scenarios remains an open question and should be explored in future work.

5.4 Security

Both implementations leverage zero-knowledge Proofs using *snarkJS* and the *Groth16* algorithm, ensuring data confidentiality and high trust for users as well as verifiers. The Circom circuit defines the cryptographic constraints and renders them transparent. This also ensures that proofs can not be altered or intercepted, since any malformation would lead to an invalid proof. Verification on the chain as in *Credchain On-Chain ZKP* enhances security by performing verification directly on the blockchain, making the verification process transparent, immutable and auditable for users and verifiers, reducing the risk of hidden manipulation.

A notable limitation in the current design is the lack of a mechanism to guarantee a freshly generated proof. Verifiers cannot be certain if a proof was generated just-in-time or previously, raising the possibility of replay attacks if an old proof is being reused in a new context. This gap could be addressed by exploring different zero-knowledge proof protocols or challenge-response mechanisms.

The results of this analysis aligns with the requirement **NFR4**: Resistance to Attacks, but also indicates room for future improvements.

5.5 Privacy

The zero-knowledge proof employed in *Credchain-ZKP* and *Credchain On-Chain ZKP* allows users to prove their identity without revealing any underlying biometric data. This ensures that sensitive biometric information remains confidential and stored locally, never being transmitted, aligned with **NFR3**: No Transmission of Biometric Data.

Furthermore, the cryptographic guarantees also fulfill **NFR5**: High-Assurance Verification without Disclosure, as users can demonstrate possession of valid biometric credentials with the same level of trust as if they would present it physically.

Both implementations also follow **NFR1**: User Control over Biometric Data, since no modifications are made how user data is captured, stored or shared compared to the base SSI systems. The user retains full autonomy over their data – zero-knowledge proofs simply introduce an additional layer for verification between actual biometric data, encapsulating it away from the verifier.

Through this analysis, it becomes clear that *Credchain-ZKP* works effectively with local verification for the purposes of the task description, while moving verification on-chain introduces associated costs and delays. These factors make on-chain verification less practical for the discussed evaluation scenarios. On-chain verification may be beneficial in scenarios where the verification process also triggers an action on the blockchain, and where additional transparency to the users is needed, thereby justifying the additional costs and delays.

Chapter 6

Final Considerations

In this chapter, this thesis will be summarized, final conclusions will be drawn and future work is proposed based on the limitations and insights.

6.1 Summary

This thesis began with the key objective of investigating the feasibility of connecting physical identity with Self-Sovereign Identity. After researching the fundamentals of SSI and its integration with physical identity, I reviewed current and existing solutions that link physical identity to digital systems.

A common issue in these solutions is verifying that the person presenting themselves is the rightful owner of a DID. These systems then rely on a second factor, a PIN or biometric data, to solve this issue. While using biometric verification, they often gather biometric data directly at the verifier, compromising user privacy and control.

To address this, I started from a minimal SSI codebase provided and implemented *Credchain-ZKP*: utilizing the two-factor authentication approach, but integrated with zero-knowledge proofs using the *snarkJS* library, allowing verification of biometric data without disclosing it. Additionally, in parallel, I implemented *Credchain On-Chain ZKP* allowing on-chain verification through a smart contract.

6.2 Conclusions

This thesis aims to improve existing mechanisms supporting usage of physical identity in the context of a digital system by securely integrating biometric verification without revealing the actual biometric information. Building upon the research gap identified in related studies, the primary goal was to develop a Self-Sovereign Identity system that ensures high assurance levels in physical use cases while keeping biometric data securely stored on the user’s device, preventing any disclosure to external parties.

In the chapter 5 Evaluation, the developed implementations are compared against the baseline project, *Credchain with 2FA*. The *Credchain-ZKP* implementation successfully demonstrated the integration of zero-knowledge proofs to prove ownership of biometric data without exposing the data itself. This approach enhances the verifiability of digitally represented physical identities while respecting users’ privacy and protecting their biometric information. Additionally, *Credchain On-Chain ZKP* explored outsourcing the verification process to the blockchain.

Overall, the challenge of integrating biometric data into SSI systems in a secure and privacy-preserving manner, and thus the overlying goal of improving digital and physical identity, has been successfully addressed. The implementations provided insights into the trade-offs between local and on-chain verification. On-chain verification increased transparency for users but comes with significantly higher costs and possible waiting times, while local processing did serve the evaluation goals outlined in chapter 5. By ensuring that biometric data remains under the user’s control and is not transmitted externally, the proposed SSI system designs aligns with the principles of user autonomy and data minimization.

However, chapter 5.4 outlines a notable limitation in regards of verifiers trust and security concerns: the current design cannot ensure proofs are freshly generated, making it potentially vulnerable to replay attacks. This could be mitigated by adopting alternative zero-knowledge proof protocols or challenge-response mechanisms.

Both implementations were particularly successful due to the choice of *snarkJS*, which significantly simplified the integration of zero-knowledge proofs for users, verifiers, and smart contracts. At the same time, working with *snarkJS* required careful and precise understanding of parsing objects in the code and creation of a circuit in Circom. This aspect of the project demanded substantial conceptual work and planning, constituting a majority of the development efforts, in order to enable a reliable, efficient and useful implementation and meeting the key objectives.

A notable modification during the creation of this thesis was the addition of on-chain verification. This addition provided an alternative perspective and enabled the verification process to be outsourced from the verifier’s device to the blockchain.

A deviation from the initial work schedule was mainly caused by an insufficient foundation of related work at the beginning. Extending and pivoting the review and analysis of relevant literature was deemed to be both necessary and beneficial for identifying and addressing the research gap.

6.3 Future Work

Given the outcomes and limitations of this thesis, following future work is proposed:

- Enhancing on-chain verification: Deploying the verifier on a blockchain test network would provide more accurate data regarding gas costs and realistic transaction times, offering better understanding of the implications of on-chain verification.
- Developing an application: *Credchain-ZKP* and *Credchain On-Chain ZKP* function as proof-of-concepts. Embedding within a smartphone application and a dedicated verifier would strengthen the claims made through this thesis. This would also enable capturing actual biometric data and evaluate its modalities, limits and the user experience while interacting with the system.
- Exploring zero-knowledge proofs: As outlined in 5.4 Security, the implementations could be enhanced by a mechanism ensuring freshness of the proof. To address this, exploring alternative zero-knowledge proof systems or implementing a challenge-response mechanism could prove effective.

Bibliography

- [1] Alexander Mühle et al. “A survey on essential components of a self-sovereign identity”. In: *Computer Science Review* 30 (2018), pp. 80–86.
- [2] Christoph Allen. *The Path to Self-Sovereign Identity*. Apr. 2016. URL: <https://www.lifewithalacrity.com/article/the-path-to-self-sovereign-identity/> (visited on Sept. 7, 2024).
- [3] Benedikt Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *Cryptology ePrint Archive* (2017).
- [4] Scott Brady. *Technical Review of Civic’s Secure Identity Platform*. Feb. 2018. URL: <https://www.scottbrady91.com/blockchain-identity/technical-review-of-civics-secure-identity-platform> (visited on Sept. 29, 2024).
- [5] Civic Technologies, Inc. *Civic Introduces Physical Identity Card to Combat AI-Driven Identity Fraud - Civic Technologies, Inc.* Mar. 2024. URL: <https://www.civic.com/blog/introducing-civic-id/> (visited on Sept. 29, 2024).
- [6] Civic Technologies, Inc. *Civic Pass | trust, control and safety for digital identity*. 2024. URL: <https://www.civic.com/> (visited on Sept. 29, 2024).
- [7] Civic Technologies, Inc. *Quickstart | Civic Docs*. Dec. 2024. URL: <https://docs.civic.com/> (visited on Sept. 29, 2024).
- [8] CoinGecko. *Ethereum Price: ETH Live Price Chart, Market Cap & News Today | CoinGecko*. Jan. 2025. URL: <https://www.coingecko.com/en/coins/ethereum> (visited on Jan. 7, 2025).
- [9] Spela Cucko et al. “Towards the Classification of Self-Sovereign Identity Properties”. In: *IEEE Access* 10 (2022), pp. 88306–88329.
- [10] Daria Schumm. *schummd/credchain*. Nov. 2022. URL: <https://github.com/schummd/credchain> (visited on Dec. 31, 2024).
- [11] Etherscan.io. *Ethereum Gas Tracker | Etherscan*. Jan. 2025. URL: <https://etherscan.io/gastracker> (visited on Jan. 6, 2025).
- [12] European Blockchain Services Infrastructure. *About ESSIF framework – ESSIF Playground*. 2024. URL: <https://docs.essif.sk/en/about-essif-framework/> (visited on Oct. 8, 2024).
- [13] European Blockchain Services Infrastructure, ed. *ESSIF Playground – ESSIF Playground*. 2024. URL: <https://docs.essif.sk/en/home-english/> (visited on Oct. 8, 2024).
- [14] European Commission. *eIDAS Levels of Assurance*. 2014. URL: <https://ec.europa.eu/digital-building-blocks/sites/display/DIGITAL/eIDAS+Levels+of+Assurance> (visited on Aug. 25, 2024).

- [15] European Union, Directorate-General for Communication, ed. *The many use cases of the EU Digital Identity Wallet - EU Digital Identity Wallet*. 2024. URL: <https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/The+many+use+cases+of+the+EU+Digital+Identity+Wallet> (visited on Oct. 8, 2024).
- [16] Edd Gent. “A Cryptocurrency for the Masses or a Universal ID?: Worldcoin Aims to Scan all the World’s Eyeballs”. In: *IEEE Spectrum* 60.1 (2023), pp. 42–57.
- [17] Alexandra Giannopoulou. “Digital Identity Infrastructures: a Critical Approach of Self-Sovereign Identity”. In: *Digital society : ethics, socio-legal and governance of digital technology* 2.2 (2023).
- [18] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208.
- [19] GOV.UK. *How to prove and verify someone’s identity*. 2024. URL: <https://www.gov.uk/government/publications/identity-proofing-and-verification-of-an-individual/how-to-prove-and-verify-someones-identity> (visited on Aug. 24, 2024).
- [20] Paul Grassi et al. *Digital identity guidelines: authentication and lifecycle management*. Tech. rep. 800-63B. National Institute of Standards and Technology, 2017.
- [21] Paul Grassi et al. *Digital identity guidelines: enrollment and identity proofing*. Tech. rep. 800-63A. National Institute of Standards and Technology, 2017.
- [22] Paul Grassi et al. *Digital identity guidelines: revision 3*. Tech. rep. 800-63-3. National Institute of Standards and Technology, 2017.
- [23] Chunjie Guo, Lin You, and Gengran Hu. “A Novel Biometric Identification Scheme Based on Zero-Knowledge Succinct Noninteractive Argument of Knowledge”. In: *Security and Communication Networks* 2022 (2022), pp. 1–13.
- [24] Harris Alem Yar. *so-ri/credchain-2FA*. Jan. 2025. URL: <https://github.com/so-ri/credchain-2FA> (visited on Jan. 2, 2025).
- [25] Harris Alem Yar. *so-ri/credchain-onchain-zkp*. Jan. 2025. URL: <https://github.com/so-ri/credchain-onchain-zkp> (visited on Jan. 7, 2025).
- [26] Harris Alem Yar. *so-ri/credchain-zkp*. Jan. 2025. URL: <https://github.com/so-ri/credchain-zkp> (visited on Jan. 2, 2025).
- [27] iden3. *Circom 2 Documentation*. Nov. 2024. URL: <https://docs.circom.io/> (visited on Jan. 2, 2025).
- [28] iden3. *iden3/snarkjs: zkSNARK implementation in JavaScript & WASM*. Oct. 2024. URL: <https://github.com/iden3/snarkjs> (visited on Dec. 31, 2024).
- [29] Joel Khalili. *I Looked Into Sam Altman’s Orb and All I Got Was This Lousy Crypto*. July 2023. URL: <https://www.wired.com/story/sam-altman-orb-worldcoin-tools-for-humanity/> (visited on Sept. 23, 2024).
- [30] Kiva. *Kiva announces the sunset of Kiva Protocol | Kiva*. May 2022. URL: <https://www.kiva.org/blog/sunset-kiva-protocol> (visited on Oct. 8, 2024).
- [31] Michael Kuperberg. “Blockchain-Based Identity Management: A Survey From the Enterprise and Ecosystem Perspective”. In: *IEEE Transactions on Engineering Management* 67.4 (2020), pp. 1008–1027.
- [32] eSSIF Lab. *home - eSSIF-Lab*. Sept. 2024. URL: <https://essif-lab.eu/> (visited on Sept. 29, 2024).

- [33] Bernhard K. Meister and Henry C. W. Price. “Gas fees on the Ethereum blockchain: from foundations to derivative valuations”. In: *Frontiers in Blockchain* 7 (2024).
- [34] OKX Group. *What is Civic (CVC)?* Apr. 2024. URL: <https://www.okx.com/de/learn/what-is-civic> (visited on Sept. 23, 2024).
- [35] *OnePlus 7T - Full phone specifications*. Sept. 2019. URL: https://www.gsmarena.com/oneplus_7t-9816.php (visited on Jan. 12, 2025).
- [36] Roberto A. Pava-Díaz, Jesús Gil-Ruiz, and Danilo A. López-Sarmiento. “Self-sovereign identity on the blockchain: contextual analysis and quantification of SSI principles implementation”. In: *Frontiers in Blockchain* 7 (2024).
- [37] Pearson Education. *Understanding the Three Factors of Authentication | Understanding the Three Factors of Authentication | Pearson IT Certification*. June 2011. URL: <https://www.pearsonitcertification.com/articles/article.aspx?p=1718488> (visited on Aug. 25, 2024).
- [38] Johannes Sedlmeir et al. “Transition Pathways towards Design Principles of Self-Sovereign Identity”. In: *ICIS 2022 Proceedings* (2022).
- [39] Quinten Stokkink. “Systems for Digital Self-Sovereignty”. PhD thesis. Delft University of Technology, 2024.
- [40] Quinten Stokkink et al. “A Truly Self-Sovereign Identity System”. In: *Proceedings of the IEEE 46th*, pp. 1–8.
- [41] Xiaoqiang Sun et al. “A Survey on Zero-Knowledge Proof in Blockchain”. In: *IEEE Network* 35.4 (2021), pp. 198–205.
- [42] The Reach Project. *JORDAN_CaseStudy_ReachProject2017*. 2017. URL: https://reachalliance.org/wp-content/uploads/2021/04/JORDAN_CaseStudy_ReachProject2017.pdf (visited on Oct. 5, 2024).
- [43] Roman-Valentyn Tkachuk et al. “A Survey on Blockchain-Based Telecommunication Services Marketplaces”. In: *IEEE Transactions on Network and Service Management* 19.1 (2022), pp. 228–255.
- [44] 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* 2 (2020).
- [45] Worldcoin Foundation. *World ID*. 2024. URL: <https://de-de.worldcoin.org/world-id> (visited on Sept. 23, 2024).
- [46] YCharts. *Ethereum Average Block Time Daily Trends: Ethereum Statistics | YCharts*. Jan. 2025. URL: https://ycharts.com/indicators/ethereum_average_block_time (visited on Jan. 5, 2025).
- [47] Razieh Zaeem et al. “Blockchain-Based Self-Sovereign Identity: Survey, Requirements, Use-Cases, and Comparative Study”. In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 128–135.
- [48] Lu Zhou et al. “Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities”. In: *Journal of Information Security and Applications* 80 (2024), p. 103678.

Abbreviations

2FA	Two-Factor Authentication
EBSI	European Blockchain Service Infrastructure
eIDAS	Electronic Identification, Authentication and Trust Services
ESSIF	European Self-Sovereign Identity Framework
FR	Functional Requirements
ID	Identification
MFA	Multi-Factor Authentication
NFR	Non-Functional Requirements
NIST	National Institute of Standards and Technology
SSI	Self-Sovereign Identity
WFP	World Food Programme
ZKP	Zero-Knowledge Proof
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge
zk-STARK	Zero-Knowledge Scalable Transparent Arguments of Knowledge

List of Figures

3.1	UML Sequence Diagram of Enrollment Flow	23
3.2	UML Sequence Diagram of zk-SNARK Proof Generation	24
3.3	UML Sequence Diagram of Credchain-ZKP Verification Flow	25
3.4	UML Sequence Diagram of Credchain On-Chain ZKP Verification Flow	26
4.1	<i>Credchain with 2FA</i> : Before Verification	32
4.2	<i>Credchain with 2FA</i> : After Verification	32
4.3	<i>Credchain-ZKP</i> : Before Verification	35
4.4	<i>Credchain-ZKP</i> : After Successful Verification	36
4.5	<i>Credchain-ZKP</i> : After Failed Verification	37
4.6	<i>Credchain On-Chain ZKP</i> : Before Verification	43
4.7	<i>Credchain On-Chain ZKP</i> : After Successful Verification	44
4.8	<i>Credchain On-Chain ZKP</i> : After Failed Verification	45
5.1	Comparison of Average GenerateProof Execution Times Across Devices.	48
5.2	Comparison of Average VerifyDID Execution Times Across Devices.	49

List of Tables

2.1	Analysis of Six Identity Management Systems	13
3.1	Categorization of Requirements into Clusters	19
3.2	Clustered Requirements including Proposed Design	21
5.1	Execution Time of <i>GenerateProof</i> : iPhone 14	48
5.2	Execution Time of <i>GenerateProof</i> : OnePlus 7T	49
5.3	Execution Time of <i>GenerateProof</i> : Lenovo X1 Yoga G4 / i7 8565U	49
5.4	Execution Time of <i>VerifyDID</i> : iPhone 14	50
5.5	Execution Time of <i>VerifyDID</i> : OnePlus 7T	50
5.6	Execution Time of <i>VerifyDID</i> : Lenovo X1 Yoga G4 / i7 8565U	50
5.7	Summary of Gas Units and Cost for the Associated Smart Contracts	51

Listings

4.1	Code of <i>RegisterIdentity</i>	33
4.2	Code of <i>VerifyDID</i> in <i>Credchain with 2FA</i>	34
4.3	Code of <i>GenerateProof</i>	38
4.4	Code of <i>VerifyDID</i> in <i>Credchain-ZKP</i>	39
4.5	Code of Circom Circuit	40
4.6	Command-Line Interaction: Convert Circom Circuit into Multiple Artifacts	41
4.7	Command-Line Interaction: Initialize a Powers of Tau File	41
4.8	Command-Line Interaction: Contribute to Random Entrophy	41
4.9	Command-Line Interaction: Fix Random Entropy into Final File	42
4.10	Command-Line Interaction: Produce Proving and Verification Key	42
4.11	Command-Line Interaction: Separate Verification Key	42
4.12	Code of <i>VerifyDID</i> in <i>Credchain on-Chain ZKP</i>	46
4.13	Command-Line Interaction: Generation of Verifier.sol	46
A.1	Hardhat.config.js Including Custom Network	73
A.2	Generation of DID Object	74
A.3	Hardcoded User Data and Generated DID	74
A.4	Generated Proof and PublicSignals	74

Appendix A

Installation and Deployment

This chapter outlines installing, running and interacting with *Credchain with 2FA*, *Credchain-ZKP* and *Credchain On-Chain ZKP*.

A.1 System's Point of View

All three implementations are available in public GitHub repositories and contain all files necessary to run all interactions proposed in this thesis:

- *Credchain with 2FA*: [24]
- *Credchain-ZKP* [26]
- *Credchain On-Chain ZKP* [25]

All three repositories are standalone applications able to run:

- A local HardHat Node
- Smart contracts deployed by helper scripts
- A React frontend application

A.2 Installation

Make sure the code repository is on your local file system and that **npm** is available on your machine.

In the base directory, open a terminal and install dependencies. Then, go to the **frontend** subdirectory and install dependencies as well:

```
npm install
cd ./frontend/
npm install
```

A.3 Deployment and Startup

1. Open a terminal in the base directory and start the local Hardhat Node:

```
npx hardhat node
```

2. Open a second terminal and deploy the smart contracts:

```
npx hardhat run scripts/deploy.js
```

3. Open a third terminal, navigate to the **frontend** subdirectory, and start the React application:

```
cd ./frontend/
npm start
```

4. Open user interface in a browser under `http://localhost:3000/`

If you would like to run the frontend application on another device, e.g. a mobile phone, you need to point the project to your local machine's IP address instead of `localhost`:

1. Edit `frontend/src/App.js` and replace `localhost` with your machine's local IP:

```
const web3 = new Web3("ws://localhost:8545")
// Change to:
const web3 = new Web3("ws://192.168.178.26:8545")
```

2. Edit `hardhat.config.js` to add a network definition with your local IP:

```
networks: {  
  localNetwork: {  
    url: "http://192.168.178.26:8545",  
    chainId: 31337  
  }  
}
```

A final `hardhat.config.js` could look like:

```
1 require("@nomicfoundation/hardhat-toolbox");  
2 require("@nomiclabs/hardhat-truffle5");  
3  
4 module.exports = {  
5   solidity: "0.8.9",  
6   mocha: {  
7     timeout: 100000000  
8   },  
9   networks: {  
10    localNetwork: {  
11      url: "http://192.168.178.26:8545",  
12      chainId: 31337  
13    }  
14  }  
15 };  
16
```

Listing A.1: Hardhat.config.js Including Custom Network

3. Open a terminal in the base directory and start the Hardhat Node with your local IP:

```
npx hardhat node --hostname 192.168.178.26 --port 8545
```

4. Open a second terminal and deploy the contracts to your new network:

```
npx hardhat run --network localNetwork scripts/deploy.js
```

5. Open a third terminal, go to the `frontend` folder, and start the React app:

```
cd ./frontend/  
npm start
```

6. Connect with your IP in the browser of any device on your network:

```
http://192.168.178.26:3000/
```

The application should be accessible, allowing you to proceed with each step of the user interface.

A.4 Data Structures

Following central artifacts are used, generated and output by all implementations. Some numbers have been abbreviated for improved readability.

```
1 const ubaasDID = poseidon3([biometricDecimalString,
  issuerToDecimalString, nowDecimalString], 1).toString();
```

Listing A.2: Generation of DID Object

```
1 Holder is: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
2 Issuer is: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
3 Date is: 562
4 Biometric template Mock is: a24f98a8b2c2ffcf6d7777e73e...8e0
5 DID is: 137804003922047246589675762488379214062223513606362708717...094
```

Listing A.3: Hardcoded User Data and Generated DID

```
1 {
2   "proof":{
3     "pi_a":[
4       "17067569700893446944483687844131654135751603...302",
5       "13227015223689131480594699190365044235335064...254",
6       "1"
7     ],
8     "pi_b":[
9       [
10        "15535293410294163275097291947651519083381...047",
11        "138679231088626801734051173943572958123943...422"
12      ],
13      [
14        "203294632336588663031935993840288536086317...461",
15        "130742743761935980242122254414224779220657...593"
16      ],
17      [
18        "1",
19        "0"
20      ]
21    ],
22    "pi_c":[
23      "219264817017124704975608980044782359608185648...920",
24      "176221158428468467996695876065798429760596575...464",
25      "1"
26    ],
27    "protocol":"groth16",
28    "curve":"bn128"
29  },
30  "publicSignals":[
31    "344073830386746567427978432078835137280280269756",
32    "562",
33    "137804003922047246589675762488379214062223513606362708717...094"
34  ]
35 }
```

Listing A.4: Generated Proof and PublicSignals