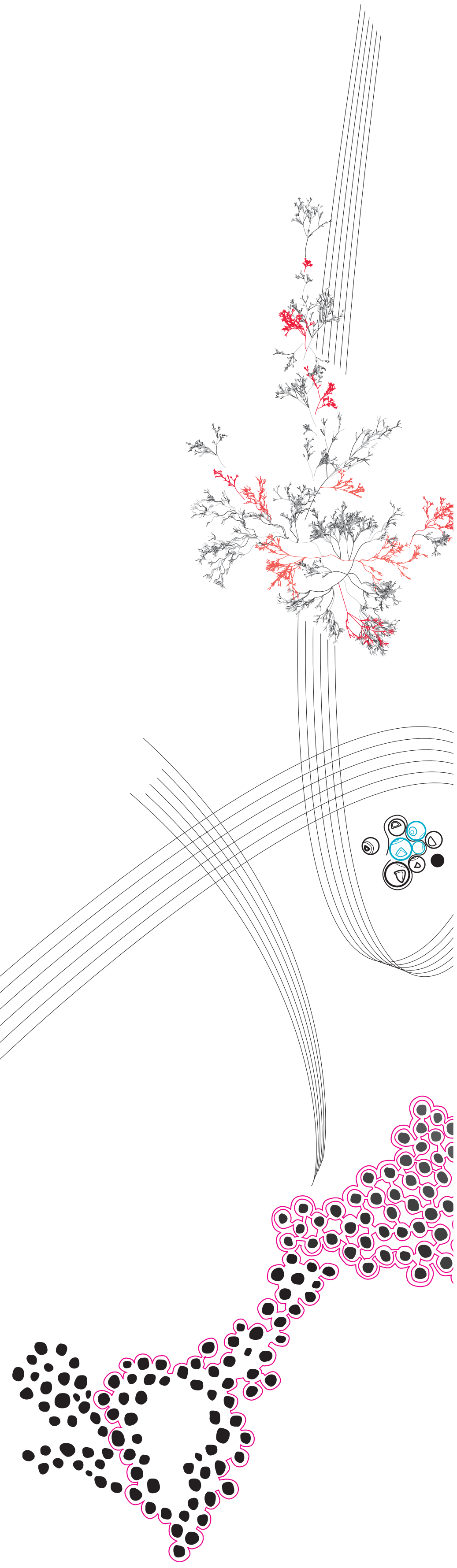
MSc Thesis Cyber Security



Comparing Zero-knowledge Proof Protocols for Practical Open Source Self-sovereign Identity Systems

Keanu Nurherbyanto Herbowo (S2430681)

Supervisor: dr.ing. F.W. Hahn

Chair: dr. R.G. Holz

March, 2022

Department of Services and Cyber Security
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

Thank you to my family and loved ones for supporting me during my studies. I would like to also extend my gratitude to my professors who have sparked my interest in cryptography, my supervisor and graduation committee who gave me an opportunity to conduct research in my chosen area, to authors of the papers I cite for indirectly teaching me new concepts, and self-sovereign identity researchers for sharing their insights through personal communication.

Abstract

Self-Sovereign Identities (SSI) are identity management systems that appoints users as the sole manager of their digital identities. Users of an SSI system are allowed to request digital credentials from issuers, and present a minimised set of data attributes to verifiers. Once issued to the user, digital credentials are only kept by the user themselves, or a third-party appointed by the user. Credential attributes can be shared by revealing them directly, or keeping them hidden to verifiers. Users can authenticate the presented attributes by providing a Zero-Knowledge Proof (ZKP) of valid issuer signatures on their credentials to the verifiers. A ZKP is one of the basis of SSI systems since it allows the user to proof that the signatures are true, without actually sending them to the verifier. Due to their growing popularity, several open source SSI systems have been developed. Some approaches such as Sovrin implement a blockchain, or a distributed ledger, to manage credential schemas. While other SSI projects such as IRMA does not. In this thesis, the two open source SSI systems; Sovrin and IRMA, will be studied using an SSI evaluation framework. Their differences in terms of their reliance to a central authority and usability of each system is highlighted. Then, a comparison of the ZKP protocol used in their underlying anonymous credential system was done. Both SSI system utilized the Identity Mixer (Idemix) anonymous credential system. Hence, the main difference between the Idemix implementation of Sovrin and IRMA is in the storage method of the credential schemas used to issue and verify credentials. Finally, the performance of the ZKP protocol used during credential disclosure in IRMA's Idemix implementation was pitted against the ZKP protocol of an alternative anonymous credential system based on Algebraic Message Authentication Code (AMAC). The evaluation was done in terms of the disclosure proof size, and disclosure proving time. The results showed that the AMAC implementation produced a proof with a smaller storage space than Idemix. However, while the AMAC implementation showed faster proof creation times than Idemix for a small number of attributes, its performance deteriorates when the number of proven attributes becomes larger than 25. Despite these benchmarks results, it seems that SSI systems such as IRMA prefer Idemix because the digital signatures on credentials can be verified using the issuer's public key. It was summarized that while alternative credential proving methods like AMACs are compatible with open source SSI systems such as IRMA, the best credential proving method depends on the SSI system's requirements.

Contents

1 Introduction

Organizations have their fair share of successes and failures when deploying identity management systems. Recently, security researchers have managed to gain access to 1.3 million user data from Indonesia's electronic Health Alert Card (eHAC) records that were stored in an insecure database [1]. The leak describes how hackers can extract Personal Identifiable Information (PII) and Covid-19 test results of eHAC users from the government's central database. Due to their ability to minimize the risk of storing sensitive user data, Self-Sovereign Identities (SSI) are seen as one of the solution to identity management problems. Delegating the autonomy of digital credentials to users can increase security and introduce efficient verification procedures [2]. An SSI system will empower users to have control over access to their digital credentials, by allowing users to be the sole custodians of their digital credentials and choose which credential information are shared with a verifying entity [2, 3]. Verifiers can request and validate the latest information with users to maintain up-to-date records. In addition, restricting presented attributes to a minimal set can achieve data minimization.

SSI systems use Attribute-Based Credentials (ABC) to represent data in a credential, and authenticate users who bear credentials containing identifying attributes required by the verifying entity [4, 5, 6]. Attributes possessed by the owner are represented as integers and contained within a transferable cryptographic container. A Zero-Knowledge Proof (ZKP) protocol allows selective disclosure of attributes contained in an ABC. ZKPs possess a useful feature that enhances data minimization by allowing users to prove that their credential contains a valid attribute without disclosing additional information about them [7]. With the advent of Privacy-Enhancing Technologies (PET), various projects have proposed SSI implementations using different techniques. Some incorporate a blockchain in their SSI system such as the Estonian government's national identity system [8], uPort [9], and Sovrin [10]. While other projects such as IRMA [11] and UnlimitID [12] rely on non-blockchain approaches. From these aforementioned SSI projects, Sovrin and IRMA have provided open source access to their resources and uses ZKPs to prove valid ownership of issuer signatures in their credentials.

Both Sovrin and IRMA implement the identity mixer (Idemix) anonymous credential system which exchanges ZKPs of Camenisch-Lysyanskaya (CL) signatures [13]. This design decision is taken despite the existence of other anonymous credential systems such as those based on Algebraic Message Authentication Code (AMAC) [14] that are reported to have better performance than signature-based approaches. Based on this observation, this thesis posits the main research question:

Why do Sovrin and IRMA use Idemix instead of keyed verification credentials based on AMACs?

In this thesis, the difference between Sovrin and IRMA as an SSI system, along with the ZKP protocols used for credential disclosure are examined. In addition, the applicability of an alternative credential proving method such as AMAC was also examined by implementing it in one of the aforementioned SSI systems. Therefore, the following additional research questions are formulated:

- **RQ1:** What is the difference between Sovrin and IRMA's SSI implementation? Through this research question, the following sub-questions are proposed:
  - **SQ1.1:** What are the benefits and downsides of a blockchain SSI system?
  - **SQ1.2:** What are the benefits and downsides of non-blockchain SSI system?
- **RQ2:** What is the difference between Sovrin's and IRMA's ZKP protocol? Are different ZKP protocols interchangeable between them?

- **RQ3:** Can keyed verification credentials based on AMACs be implemented as proving mechanism for Sovrin or IRMA? What are the differences in performance between their current ZKP protocols and the newly implemented alternative method?

Through answering these research questions, the findings presented in this thesis will contribute an evaluation of Sovrin and IRMA in terms of their SSI characteristics and Idemix credential proving mechanism. Recent studies have presented their evaluation of both SSI system in terms of their functionality [15, 16, 17]. But the literature review suggest that a direct in-depth study of their ZKP protocol does not exist. Furthermore, the findings described in this thesis will showcase the applicability of alternative credential proving methods for Sovrin or IRMA. This is done by comparing the credential proving performance of keyed verification credentials based on AMACs against the Idemix implementation used in IRMA.

The thesis document will be divided into the following chapters. In chapter 2 the literature review of relevant works are presented. The literature review contains the principles and components of SSI systems, cryptography preliminaries of anonymous credentials, along with an elaboration of ABC and blockchain for SSI systems. Then, in chapter 3 a brief description of existing SSI systems including Sovrin and IRMA are given. Next, in chapter 4 Sovrin and IRMA's SSI implementation are compared using an SSI evaluation framework [15]. Afterwards, in chapter 5 the difference between Sovrin and IRMA's ZKP protocols are elaborated. In chapter 6, the credential proving benchmarks of keyed verification credentials using AMACs against the Idemix implementation used in IRMA are presented. Finally, to discuss whether each research question have been addressed, all findings pertaining each research question are summarized in chapter 7.

2 Literature Review

This chapter will present a literature review of concepts relevant to this thesis. The first part of the literature review covers the principles of SSI systems including an elaboration of the participants in an SSI system and two proposed SSI standards. The second part of the literature review contains an explanation of the cryptographic concepts related to anonymous credentials. Finally, the third part of the literature review includes a brief description of ABCs and blockchain technology.

2.1 Principles of Self-sovereign Identity Systems

A digital identity is defined as a set of self-attested or assigned validated claims about a digital entity [18]. Digital identities can contain the same attributes defined in physical identities such as a birth certificate, passport, or diploma. Throughout the years, digital identity management models have evolved from a centralized model to more user-centric models [2, 3]. In a centralized identity model, each identity provider manages their own record of the user's digital identity. However, as users begin to use varying Internet services such as email and social media from different companies, this centralized model lacks portability, leading to the establishment of the federated identity model. Federated identities allow users to use an existing digital identity from one identity provider with services offered by third-parties. In the Federated identity model, user accounts are bound to policies and restrictions set by the identity provider. To introduce more freedom and autonomy to users, user-centered identity models such as OpenID[2] were created. User-centered identity models allow users to generate their own credentials and use them with third-parties. But, since managing these user-centered credentials require some technical knowledge, many users still gravitate to identity services provided by major online giants such as Facebook and Google that still share the common restrictions of federated identities. SSI systems are the latest iteration of identity management models.

In Allen's highly cited proposal [3], SSI is described as an emerging identity management model that allows individuals to have autonomy over third-party access to their credentials. Users are free to manage and present their credentials, without having to suffer from lock-ins and restrictions. Selective disclosure is a feature of SSI that can enhance user privacy, by allowing users to present a minimized set of attributes to verifiers. Another important characteristic of SSI is user consent. Therefore, even if credential access is given to a third-party, they still require the user's permission to share information about the credential with other third-parties. The ten principles that describe the objectives and characteristics of SSI are meticulously presented in Allen's proposal [3]. The ten SSI principles are:

1. **Existence**. A self-sovereign identity contains publicly accessible attributes of an existing physical identity.
2. **Control**. Users should always be able to present, alter, and keep their identity, as well as make claims to its validity.
3. **Access**. Users are not bound to policies and restrictions preventing them from retrieving and accessing identity claims or data.
4. **Transparency**. The systems and algorithms used to issue and verify identities in the network should be transparent.
5. **Persistence**. Identities should exist for a long time, or until its validity is revoked by the user.
6. **Portability**. Data regarding identities must not be kept solely by a centralized third-party.
7. **Interoperability**. Identity should be usable between a large number of services.
8. **Consent**. Users must consent to the access of their identity by a third-party.
9. **Minimalization**. When user data is disclosed, it should contain the least amount of information to complete a transaction.
10. **Protection**. The rights of users must be protected, even when a conflict arises between the needs of the network against the user's rights.

2.2 Participants in Self-sovereign Identity Systems

Up until now, several participants of an SSI system have been mentioned. But this section will formally define the role of issuers, provers, verifiers, and verifiable data registries in an SSI system. These descriptions are summarized from [4, 19].

Issuer. An issuer is a trusted entity in the SSI system that can create claims about a user or subject. These claims are then packaged into credentials and given to the user. The credentials are signed with digital signatures to certify their authenticity for when they are presented to a verifying entity.

Prover. Once a credential is issued, a prover can present attributes in the credential to authenticate themselves with verifiers. To avoid confusion, *users* will be generalized as provers for the rest of this discussion. It should also be mentioned that provers do not necessarily have to be the subjects of a credential, as shown in section 2.3 about SSI standardization efforts where a prover can present the credential of another individual.

Verifier. A verifier is an entity that receives and processes presented credentials. Verifiers will validate a credential to determine if they contain a valid issuer's signature. Issuers and verifiers can be a separate or a single entity.

Verifiable data registry. A verifiable data registry can either be a participant or a system-specific repository. Their main obligation is to manage the credential schemas used during credential issuance and verification, manage issuer keys, and maintain a revocation list of revoked prover credentials. In an SSI system that uses a blockchain such as Sovrin, the blockchain acts as a verifiable data registry[20]. In a non-blockchain SSI system such as IRMA, this role is fulfilled by the schema manager [4].

2.3 SSI Standardization Efforts

The World Wide Web Consortium (W3C) have established the Verifiable Credentials Task Force (VCTF) tasked with creating SSI standards online [21]. The VCTF has recommended two standards for SSI systems which have abstract descriptions to allow for a wide variety of implementation.

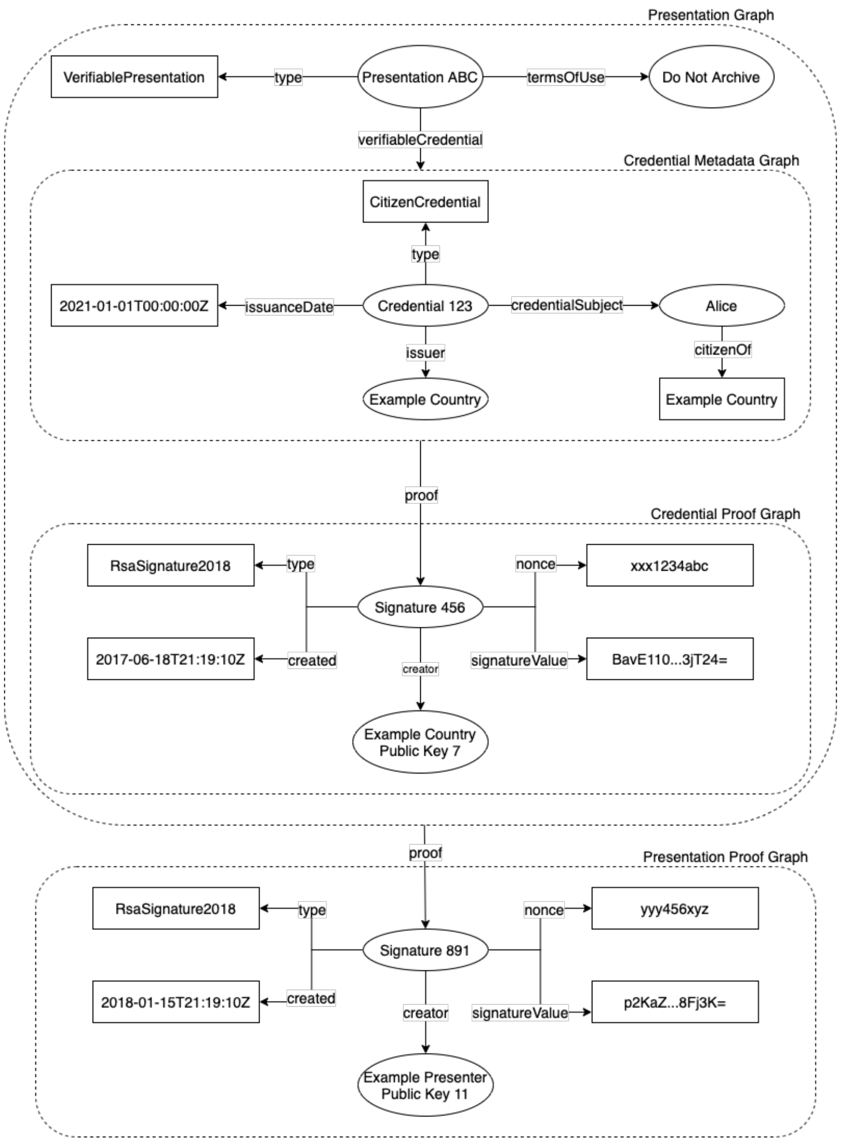


FIGURE 1: Verifiable Credential and Verifiable Presentation Graph

The first online standard recommended by the VCTF is the Decentralized Identifiers (DID) data model [22]. Decentralized identifiers are unique identifiers that reference an object called a DID document which contains information associated to a digital identity. In the documentation [22], DID is akin to Uniform Resource Identifiers (URI). DID documents are serialized data models which contain the identifier property, verification method property, and service property. The identifier property can be used to identify the owner of the identity called the DID subject, and entities who are authorized to make changes to the DID document are called DID controllers. Cryptographic keys can be included inside the verification method property to specify how subjects can present authorization or cryptographic proofs to third-parties. The service property of a DID document expresses ways that the DID document can be used to communicate with the DID subject such as authentication or interaction methods.

The second online standard recommended by the VCTF is the Verifiable Credentials (VC) data model[19]. A verifiable credential is a collection of tamper-evident claims made possible by the addition of digital signatures. A claim is an assertion made about a subject's attribute (*e.g*, prover is a citizen of a certain country). In the case of digital identities, a claim can describe how a prover possesses a unique identifier, certain role permissions, or knowledge of a secret key [18]. The VC recommendation describes three roles that an entity in a network can perform [19]. These roles are akin to the SSI participants that have been previously described. An issuer, is an entity that performs assertions about a subject's attribute and signs the claims for a verifiable credential. A subject is the entity whose assertions are made to. The subject can present their credentials to verifiers, but the VC recommendation also states that subjects may delegate their credentials to a third-party that presents them on their behalf. Lastly, a verifier is an entity that validates a verifiable credential.

The three basic component of a verifiable credential are the credential metadata, claims, and proofs [19]. These components are expressed using the DID document specification and is shown in Figure 1 as the credential metadata graph. A graph describes how a subject who have claims, are related to other subjects or data [19]. As shown in Figure 1, the credential metadata contains the type of the verifiable credential, issuer DID, the issuance timestamp, and a claim referred by the credential. In this case, `Alice` claims to be a citizen of `Example Country`, and this claim is enclosed in a credential issued by the country's central government. The credential metadata graph is linked to a proof graph that contains the issuer's digital signature. The proof component of a verifiable credential consists of the digital signature type, nonce, signature value, signature timestamp, and the public key of the signing entity. The proof graph shown in Figure 1 contains a digital signature signed by `Example Country`, and a copy of their public key is provided to allow verifiers to validate the signature.

When a prover exchanges a verifiable credential with a verifier, it will package it in a verifiable presentation. A verifiable presentation is a representation of the verifiable credential data given to the verifier. The verifiable presentation consists of the presentation metadata component, the verifiable credential, and the presentation proof component. In Figure 1, the credential metadata and credential proof are combined with a presentation metadata that contains the type and terms-of-use of the verifiable presentation. The verifiable presentation is then linked to a presentation proof that contains the prover's digital signature. The presentation proof consists of the digital signature type, nonce, signature value, signature timestamp, and the public key of the proving entity.

To allow presentations without attribute disclosure, the verifiable credential and verifiable presentation can be extended using ZKPs to create presentations that contain claims derived from a verifiable credential. While the details of the format of ZKPs for a verifiable presentation are not specified, the VC data model [19] specified three requirements:

1. A `credentialSchema` property must be present in the derived verifiable credentials to allow referencing of the credential model used to generate the credential proof.

2. The verifiable presentation must not contain any data that can enable the verifier to link the prover between multiple verifiable presentations.
3. The verifiable presentation must contain a `proof` property to allow the verifier to validate that the credentials presented in the verifiable presentation were issued to the prover without disclosing additional data.

2.4 Anonymous Credentials

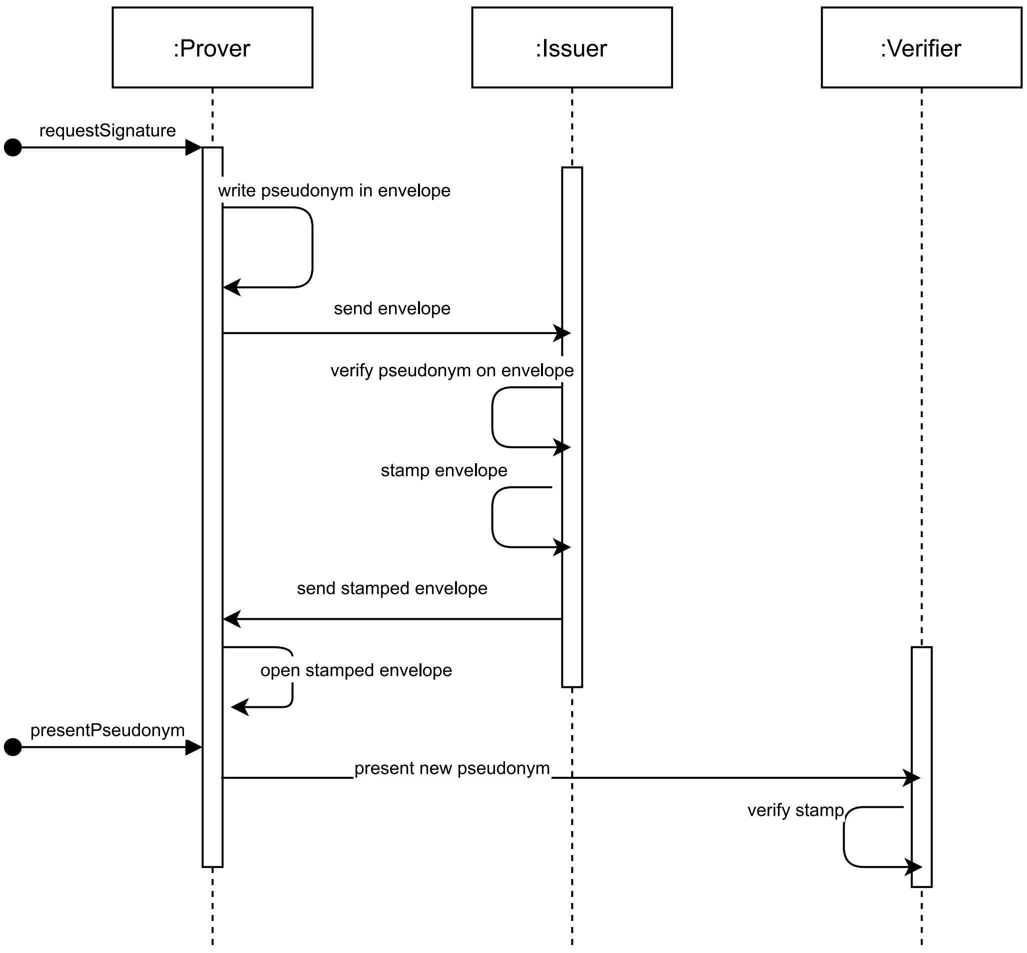


FIGURE 2: Sequence diagram of Chaum's basic anonymous credential system

Anonymous credentials are used to exchange messages during issuance, presentation, and verification of digital credentials. Anonymous credentials was initially proposed by Chaum [23], illustrated using the analogy of a prover and an issuer in the midst of a credential issuance process. Figure 2 describes Chaum's anonymous credential proposal pictorially. A prover requests an issuer to sign a new pseudonym written on a piece of paper, lined with carbon paper, and inserted into an envelope that has the prover's known pseudonym. The issuer then receives the envelope with the known pseudonym written on the envelope, signs the envelope with a stamp, and sends the stamped envelope back to the prover. The pseudonym inside the envelope now possesses the issuer's signature (due to the carbon paper being stamped), while the issuer has no knowledge of the new pseudonym inside the envelope. The prover can then proceed to use the newly signed pseudonym with a verifying entity. The verifier accepts the presented pseudonym because it has the issuer's signature.

The basic anonymous credential system proposed by Chaum [23] has several weaknesses. First, the pseudonyms can only be presented to the verifier once, lacking anonymity since a used pseudonym can possibly be linked to an individual. Second, the prover authenticates using pseudonyms, not attributes like modern credential systems. The revisions addressing these weaknesses are not discussed in this document. However, the concepts introduced in Chaum's proposal are implemented in modern anonymous credential systems.

2.4.1 Commitment Schemes

A method that prevents other parties from learning about a committed value is called a commitment scheme [7]. A commitment scheme consists of a *commitment* phase where each party in the network commits a secret value unbeknownst to their counterparts, and a *reveal* phase where each party make their secret values public. In Chaum's anonymous credential system [23], the pseudonym given to the issuer is enclosed in an envelope to prevent the issuer from learning about the pseudonym. Furthermore, the prover cannot simply take an arbitrary pseudonym and combine it with a signature from another pseudonym because the issuer stamps the pseudonym while it is still sealed in the envelope. The issuer's inability to distinguish the contents of the envelope and the prover's inability to alter the pseudonym signed by the issuer correspond to a commitment scheme's hiding and binding security properties.

In order to intuitively define the security properties of commitment schemes, an abstract commitment scheme shall be used as an example. Let $\mathbb{P}$ be a set of prime numbers and $\mathbb{R}$ be a set of real numbers. A value $x \in \mathbb{P}$ and a random element $r \in_R \mathbb{R}$ are committed in a commitment scheme using the public algorithm $c \leftarrow C(x,r)$.

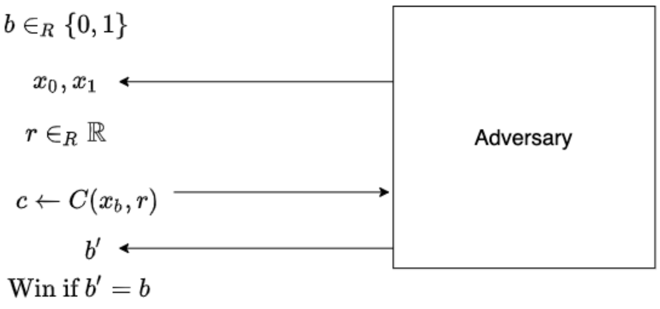


FIGURE 3: Hiding security game for commitment schemes

The hiding property corresponds to the verifier's ability to distinguish a prover's committed secret. A commitment scheme is information-theoretically (resp. computationally) hiding if no infinitely powerful (resp. computationally bounded) adversary can win the hiding security game [7]. Figure 3 displays the hiding security game for the abstract commitment scheme. The game starts by the adversary generating two messages x_0 and x_1 with equal lengths. The challenger then generates a random $r \in_R \mathbb{R}$ and a random bit $b \in_R \{0,1\}$. The challenger outputs $c \leftarrow C(x_b,r)$ to the adversary, with the bit b randomly selecting either message x_0 or x_1. Finally, the adversary wins if they can successfully determine whether x_0 or x_1 is used in the commitment c.

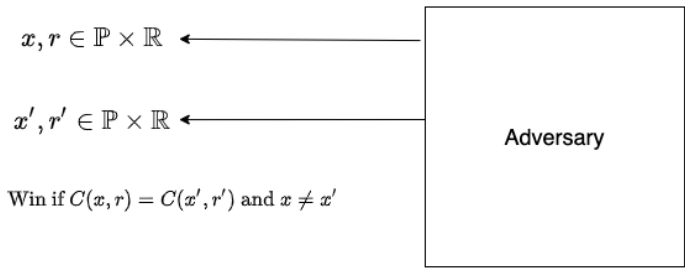

$x, r \in \mathbb{P} \times \mathbb{R}$
$x', r' \in \mathbb{P} \times \mathbb{R}$
Win if $C(x,r) = C(x',r')$ and $x \neq x'$
Adversary

FIGURE 4: Binding security game for commitment schemes

The binding property corresponds to the prover's ability to alter their committed secret after making a commitment to the verifier. A commitment scheme is information-theoretically (resp. computationally) binding if no infinitely powerful (resp. computationally bounded) adversary can win the binding security game [7]. Figure 4 displays the binding security game for the abstract commitment scheme. The game begins by the adversary outputting the values $x \in \mathbb{P}$ and $r \in \mathbb{R}$. The adversary wins if they can output $x' \neq x$ and $r' \in \mathbb{R}$, where $C(x,r) = C(x',r')$.

The literature [7] describes efficient commitment schemes based on group operations such as the Pedersen's commitment and Elgamal commitment, and their security properties are the same with the abstract commitment scheme described above.

2.4.2 Zero-Knowledge Proofs

A Zero-Knowledge Proof (ZKP) protocol is used to show how a prover can convince a verifier that they possess a secret attribute without disclosing the attribute itself [7]. Referring back to Chaum's anonymous credential system [23], the issuer does not know the pseudonym in the envelope, yet is convinced that a new pseudonym is enclosed for them to sign. An interactive protocol between mutually untrusting parties can be used to exchange proofs of knowledge. In order to simply explain ZKPs and effectively describe information exchange between a prover and verifier, an honest-verifier ZKP shall be assumed for the remainder of the discussion. An honest-verifier means that the verifier will follow the protocol steps correctly [7].

A ZKP can be represented with an interactive protocol called a sigma protocol [7]. Sigma protocols consist of the commitment phase, challenge phase, and response phase. In the commitment phase, a prover creates a commitment containing the knowledge of a secret attribute. Then, the verifier will present a challenge to the prover in the challenge phase. Afterwards, the prover will generate a response that will be validated by the prover.

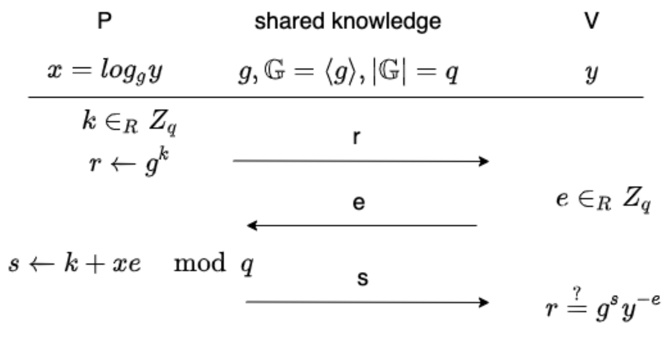

P
shared knowledge
V
$x = log_g y$
$g, \mathbb{G} = \langle g \rangle, |\mathbb{G}| = q$
y
$k \in_R Z_q$
$r \leftarrow g^k$
r
e
$e \in_R Z_q$
$s \leftarrow k + xe \mod q$
s
$r \overset{?}{=} g^s y^{-e}$

FIGURE 5: Interactive Schnorr's identity sigma protocol

A sigma protocol for the Schnorr identification protocol illustrated in Figure 5 will be used to demonstrate how ZKPs are constructed. Suppose P possesses a secret x, which is a discrete logarithm of $y \leftarrow g^x$, with respect to g in a finite abelian group G of prime order q. The prover P can proof in zero-knowledge to a verifier V that they posses the secret value using a sigma protocol [7]. The sigma protocol begins by P producing a commitment r and sends the commitment to V. Afterwards, V produces a challenge e and sends it to P. Then, P encodes the secret value x in the response s and outputs the result to V. Finally, V produces a valid or invalid bit based on the result of the verification algorithm $r \overset{?}{=} g^s y^{-e}$. The proof of knowledge of the secret value $x = \log_g y$ can be written using the notation $\pi = PK\{(x) : y \leftarrow g^x\}$ as done in [14], where the relations of the values in the proof of knowledge can be extended using the $\wedge$ operator to write proofs for multiple secrets. In later discussions, some proofs may have a nonce to ensure the freshness of the challenge and response values in the protocol. The notation used to describe a proof with a nonce will be done in a similar fashion to [24]. For example, the proof of knowledge of a secret value $x = \log_g y$ in a Schnorr identification protocol with a nonce n can be written as $\pi = PK\{(x) : y \leftarrow g^x\}(n)$.

The following formal notations for a sigma protocol can be used to define the elements of a sigma protocol [7]. The algorithm $R(w,k)$ is used to create the commitment r from a random value k and a secret value or "witness" w. The value e is the challenge from the challenge set $\mathbb{E}$. The algorithm $S(e,w,k)$ is used to create the response $s \in \mathbb{S}$ from e and k. The verification process $V(r,e,s)$ outputs a valid or invalid bit. The algorithm $S'(e,s)$ is used by the verifier during the verification process to produce the commitment r that can verify the transcript (r,e,s). Finally, the algorithm $E((r,e,s),(r,e',s'))$, sometimes called the knowledge extractor, is used to extract the secret w. This knowledge extractor is only valid for protocols that possess the special soundness property [7].

Using the previously mentioned formal notations of a sigma protocol, the components of the Schnorr identification protocol in Figure 5 can be reinterpreted as:

$$\begin{gathered}
\exists x \text{ where } y = g^x \text{ and witness } w = x \\
R(x,k) := r \leftarrow g^k \\
S(e,x,k) := s \leftarrow k + xe \mod q \\
V(r,e,s) := \text{true if and only if } r = g^s y^{-e} \\
S'(e,s) := r \leftarrow g^s y^{-e} \\
E((r,e,s).(r.e',s')) := x \leftarrow \frac{s-s'}{e-e'}
\end{gathered}$$

A ZKP protocol needs to have the completeness, soundness, and zero-knowledge properties [7].

A *complete* protocol should allow the verifier to accept the transcript with a probability of 1, if the prover possesses the secret and follows the protocol correctly. The Schnorr identification protocol is complete because if the prover provides the correct commitment and response values, the verifier can compute and verify $r \leftarrow g^s y^{-e}$ to produce a valid response bit.

The *soundness* property denotes that a verifier should only have a small chance of accepting the proof if the prover is lying about their secret. The prover in the Schnorr identification protocol can lie about having a valid secret x with the probability of $1/q$. If the protocol is executed with an invalid secret bit, the prover has a $1/q$ chance of successfully producing a valid bit in the verification phase. In some cases, a ZKP protocol possess a *special soundness* property if the secret attribute can be recovered using two sets of transcripts with the same commitment but different challenge and response bits. The Schnorr identification protocol is special sound because the secret can be recovered by computing $x \leftarrow \frac{s-s'}{e-e'}$.

The final property is *zero-knowledge*, where a set of valid transcripts V is indistinguishable from a set of simulations S. A simulation is a transcript of the sigma protocol produced by the verifier without interacting with the prover. For example, assume the simulation $S = \{(r,e,s) : s \in_R \mathbb{Z}_q, e \in_R \mathbb{Z}_q, r = g^s y^{-e}\}$ and valid transcript V of the Schnorr identification protocol. In both S and V, the values for r,e,s are selected with the same distribution as long as the requirements $r = g^s y^{-e}$ holds. Therefore, the simulation and valid transcripts of the Schnorr identification protocol are indistinguishable, making the protocol zero-knowledge.

A ZKP protocol can be interactive, or non-interactive. The Fiat Shamir heuristic can be used to convert an interactive protocol into a non-interactive protocol [7]. SSI systems such as IRMA uses a sigma protocol in the Fiat Shamir heuristic [25]. In an interactive protocol, the challenge is selected by the verifier. In a non-interactive protocol, a cryptographically secure hash function is used to hash the prover's commitment along with the public parameters and statement that has to be proven. The result of the hash function is used to replace the challenge selected by the verifier in an interactive sigma protocol. The non-interactive sigma protocol of the Schnorr identification protocol can be seen in Figure 6.

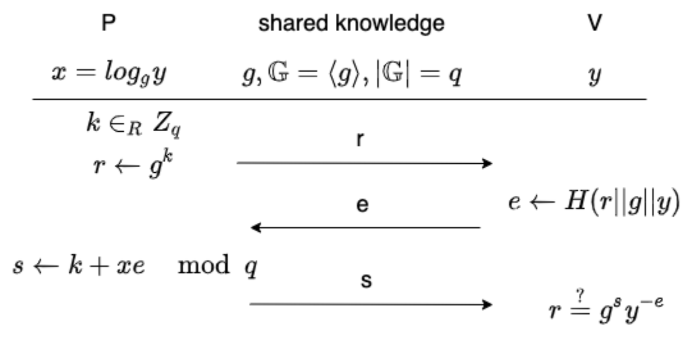

P
shared knowledge
V
$x = log_g y$
$g, \mathbb{G} = \langle g \rangle, |\mathbb{G}| = q$
y
$k \in_R Z_q$
$r \leftarrow g^k$
r
e
$e \leftarrow H(r||g||y)$
$s \leftarrow k + xe \mod q$
s
$r \overset{?}{=} g^s y^{-e}$

FIGURE 6: Non-interactive Schnorr's identity sigma protocol

2.4.3 Camenisch-Lysyanskaya Signature

Digital signature schemes are used to provide message integrity. A digital signature scheme formally consists of the signing algorithm $s \leftarrow Sig_{sk}(m)$ and the verification algorithm $Verify_{pk}(s,m)$ with the public and private key pairs (pk,sk) [7]. The sender can attach a signature generated with their private key to a message, and the recipient can verify the signature using the sender's public key. A secure signature scheme must possess the Existential Unforgeability against a Chosen Message Attack (EUF-CMA) property, where it is unfeasible for an adversary to create an existential forgery, which is a valid signature on a random message generated by the adversary [7].

One example of a digital signature scheme is the Camenisch-Lysyanskaya (CL) signature scheme used in Idemix [26]. The CL-signature scheme is selected as the basis for Idemix because a single signature can be used to certify multiple attributes, supports blind issuance of attributes, and allows provers to prove signatures in zero-knowledge [13]. The CL-signature scheme consists of the KeyGen function to generate the public and private key pairs, as well as the Sign and Verify function used to create and validate signatures. The functions used in the CL-signature scheme will be presented in the following text. Table 1 contains the bit lengths of the system parameters of CL-signatures used during the discussion.

Parameter	Description	Bitlength
ℓ_n	size of the RSA modulus	2048
ℓ_e	size of e values of certificates	597
ℓ_m	size of message attributes	256
ℓ_v	size of the v values of certificates	2724
ℓ_r	security parameter in the security proof of the system	80
ℓ_ϕ	security parameter of the statistical zero-knowledge property	80
ℓ_H	domain of the hash function H used in the Fiat-Shamir heuristic	256
ℓ_e	size of the interval of the e values	120

TABLE 1: Idemix and CL-signature bit lengths of system parameters from [13]

KeyGen(ℓ_n): Select an ℓ_n-bit RSA modulus $n \leftarrow pq$. Select $p \leftarrow 2p' + 1$, $q \leftarrow 2q' + 1$, where p, q, p', q' are prime numbers. For a length of messages l, select $R_1, ..., R_l, S, Z \in_R QR_n$, where QR_n is the set of quadratic residues modulo n. Then, output the public key $(n, R_1, ..., R_l, S, Z)$ and store the secret key p.

Sign($\vec{m}$): Parse the messages $\vec{m} \leftarrow (m_1, ..., m_l)$, then choose a random prime number e with length $\ell_e > \ell_m + 2$, and random number v of length $\ell_v \leftarrow \ell_n + \ell_m + \ell_r$ with ℓ_r as the security parameter. Afterwards, compute

$$A \leftarrow \left(\frac{Z}{\prod_{i=1}^{l} R_i^{m_i} S^v}\right)^{1/e} \mod n.$$

Finally output the signature $\sigma \leftarrow (A, e, v)$.

Verify($\vec{m}, \sigma$): Verify the signature in tuple $\sigma \leftarrow (A, e, v)$ on messages $\vec{m} \leftarrow (m_1, ..., m_l)$ by observing whether the comparison

$$Z \equiv A^e S^v \prod_{i=1}^{l} R_i^{m_i} \mod n \wedge m_i \in \{0,1\}^{\ell_m} \wedge 2^{\ell_e} > e > 2^{\ell_e - 1}$$

holds.

2.4.4 Algebraic Message Authentication Code

Message Authentication Codes (MAC) are akin to digital signatures as they are also used to provide message integrity. But unlike digital signature schemes that uses public key infrastructure (PKI), the signing and verification algorithms for MACs uses a shared private key. A MAC scheme consists of a pair of public algorithms $tag \leftarrow Mac_k(m)$ and $Verify_k(tag, m)$, where the former is used to produce the MAC or "tag" of a message, and the latter is used to verify a MAC [7]. A secure MAC scheme must possess the EUF-CMA security property, where it is unfeasible for an adversary to generate an existential forgery in the form of a valid MAC for a random message chosen by the adversary [7].

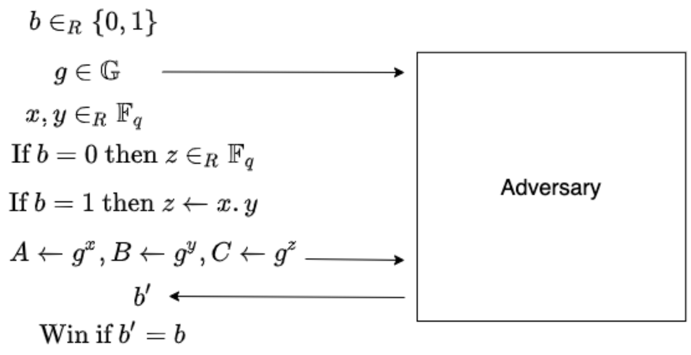

$b \in_R \{0,1\}$
$g \in \mathbb{G}$
$x, y \in_R \mathbb{F}_q$
If $b = 0$ then $z \in_R \mathbb{F}_q$
If $b = 1$ then $z \leftarrow x.y$
$A \leftarrow g^x, B \leftarrow g^y, C \leftarrow g^z$
b'
Win if $b' = b$
Adversary

FIGURE 7: Decisional Diffie-Hellman Security Game

MACs constructed using prime order groups are called Algebraic Message Authentication Codes (AMAC) [14]. AMACs have similar properties to CL-signatures in its ability to support blocks of messages, supports blind issuance protocols, and allow ZKPs to be used when proving ownership of valid MACs. There exists two AMAC constructions namely $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$ [14]. This document will focus on the $\mathsf{MAC}_{\mathsf{DDH}}$ construction, which is based on the Decisional Diffie-Hellman (DDH) problem. Figure 7 illustrates the DDH security game [7, 14]. Assume $\mathbb{G}$ is a cyclic group with prime order q and generator g, the values $x, y \in_R \mathbb{F}_q$, and $z \in_R \mathbb{F}_q$ or $z \leftarrow xy$. The adversary is given $(A \leftarrow g^x, B \leftarrow g^y, C \leftarrow g^z)$, and asked to distinguish whether C was generated using $z \in_R \mathbb{F}_q$ or $z \leftarrow xy$.

The $\mathsf{MAC}_{\mathsf{DDH}}$ construction consists of the Setup algorithm used to initialize public parameters *params*, the key generation function KeyGen, a function MAC that produces an authentication tag for a message, and a Verify function used to verify whether a tag is valid in respect to a key and message [14]. The vector $\vec{m} = (m_1, ..., m_n)$ is a collection of n messages in $\mathbb{F}_q$, that correspond to a set of attributes in anonymous credential systems. The functions used in the $\mathsf{MAC}_{\mathsf{DDH}}$ construction are as follows.

Setup(λ): Generate a cyclic group G with order q, where q is a λ-bit prime. Determine the generators $g, h \in_R G$, where $log_g(h)$ is unknown. Output the system parameters $params \leftarrow (g, h, q, G)$.

KeyGen(*params*): Choose $sk = z, x_0, y_0, ..., x_n, y_n \in_R \mathbb{F}_q$ and output $sk \leftarrow (\vec{x}, \vec{y}, z)$.

MAC($sk, \vec{m}$): Compute $H_x(\vec{m}) \leftarrow x_0 + \Sigma_{i=1}^n x_i m_i$, and $H_y(\vec{m}) \leftarrow y_0 + \Sigma_{i=1}^n y_i m_i$. Generate $r \in_R \mathbb{F}_q$ and set $\sigma_w \leftarrow g^r$, $\sigma_x \leftarrow g^{rH_x(\vec{m})}$, $\sigma_y \leftarrow g^{rH_y(\vec{m})}$, and $\sigma_z \leftarrow g^{zr}$. Output the tuple $\sigma \leftarrow (\sigma_w, \sigma_x, \sigma_y, \sigma_z)$.

Verify($sk, \vec{m}, \sigma$): Check that $\sigma_w \neq 1$, $\sigma_x = \sigma_w^{H_x(\vec{m})}$, $\sigma_y = \sigma_w^{H_y(\vec{m})}$, and $\sigma_z = \sigma_w^z$. If these checks pass, output a valid bit (1), otherwise output an invalid bit (0).

2.4.5 Comparison of Digital Signatures and AMAC Security Games

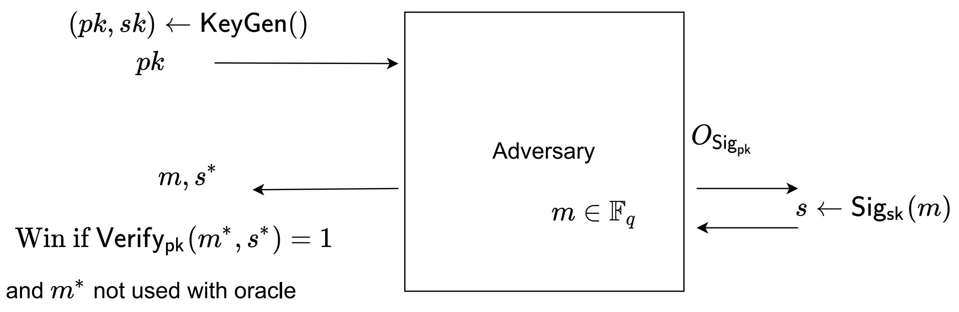

$(pk, sk) \leftarrow \mathsf{KeyGen}()$
pk
Adversary
$m \in \mathbb{F}_q$
$O_{\mathsf{Sig}_{\mathsf{pk}}}$
$s \leftarrow \mathsf{Sig}_{\mathsf{sk}}(m)$
m, s^*
Win if $\mathsf{Verify}_{\mathsf{pk}}(m^*, s^*) = 1$
and m^* not used with oracle

FIGURE 8: Digital Signatures EUF-CMA Security Game

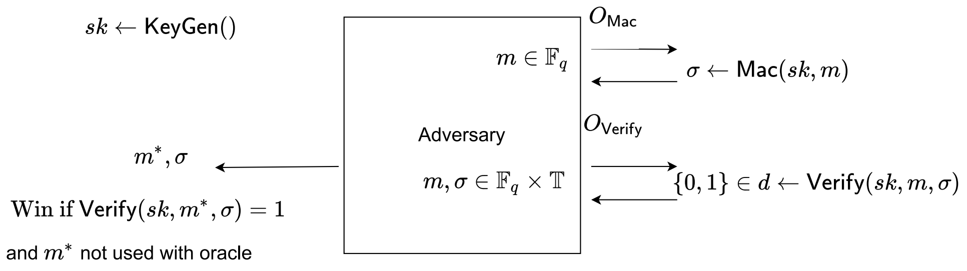

$sk \leftarrow \mathsf{KeyGen}()$
$m \in \mathbb{F}_q$
O_{Mac}
$\sigma \leftarrow \mathsf{Mac}(sk, m)$
Adversary
O_{Verify}
$m, \sigma \in \mathbb{F}_q \times \mathbb{T}$
$\{0,1\} \in d \leftarrow \mathsf{Verify}(sk, m, \sigma)$
m^*, σ
Win if $\mathsf{Verify}(sk, m^*, \sigma) = 1$
and m^* not used with oracle

FIGURE 9: MAC EUF-CMA Security Game

Both digital signatures and MACs have the same goal of achieving EUF-CMA security. The EUF-CMA security game for digital signatures and MACs are displayed in Figure 8 and Figure 9 respectively. The EUF-CMA security game simulates an adaptive adversary that can query oracles to help them sign and verify a signature / MAC on a chosen message. In Figure 8, the adversary only has a signing oracle because digital signatures can be verified with the verification algorithm that requires the public key *pk*. In Figure 9, the adversary has access to a signing and verification oracle because the adversary does not have the challenger's secret key *sk* used by the MAC and verification algorithms. In both security games, the adversary wins if they can output a message and signature / MAC that have not been queried with the oracle.

In [14], EUF-CMA is coined as existential Unforgeability against a Chosen Message Attack given a Verification Oracle (uf-cmva). Another security definition is the strong existential unforgeability against a chosen message attack (sEUF-CMA) where it is unfeasible for an adversary to provide both a valid message and a valid MAC [7]. The Idemix specifications [13] do not state whether or not CL-signatures are EUF-CMA. But CL-signatures are assumed to be secure if the underlying RSA problem is hard [26, 24]. The $\mathsf{MAC}_{\mathsf{DDH}}$ construction is designed and proven to be EUF-CMA but not sEUF-CMA [14].

2.4.6 Idemix

The Identity Mixer Cryptography Library (Idemix) is an anonymous credential system developed by IBM [13]. Idemix credentials rely on the authenticity of CL-signatures generated by the issuer. Pseudonyms *nym* and domain pseudonyms *dNym* are commitments of the prover's unique *master secret* attribute, used to mark their relationship with an issuer or verifier in an unlinkable manner

[24]. In the credential issuance phase, provers must show in zero-knowledge, that their master secret ties-in to a unique pseudonym. Thus, they form a commitment on their master secret, and sends a ZKP of a valid and unique master secret. Once the proof verifies, the issuer creates CL-signatures for the requested attributes and sends to the prover a ZKP attesting that they were generated using correct issuer keys. Alternatively, if the credential system does implement pseudonyms like in Sovrin and IRMA, the prover can omit the commitment on the master secret and provide the commitments $C_i \leftarrow Z^{m_i} S^{r_i} (\mod n)$ for each $i \in \{1, ..., l\}$ instead. The resulting commitments contain the prover's unique master secret since that unique attribute is m_1. In the credential presentation phase, the prover can present to the verifier a ZKP of valid CL-signatures on their credentials.

The following processes of an Idemix anonymous credential system were taken from [13] and [24]. For a description of the symbols used while describing an Idemix system, refer to Table 1 in section 2.4.3, and Table 2.

Parameter	Description
Γ	modulus of the commitment group
ρ	prime order of subgroup $\mathbb{Z}_\Gamma^*$
β	cofactor of $\Gamma - 1$

TABLE 2: Idemix system parameters from [13]

System Setup: Generate a commitment group $\mathbb{Z}_\Gamma^*$ with order $\Gamma - 1 = \rho . \beta$ for a large prime ρ. Compute a generator g from the group $\mathbb{Z}_\Gamma^*$, by selecting $g' \in_R \mathbb{Z}_\Gamma^*$, where $g'^{\beta} \neq 1 (\mod \Gamma)$, and compute $g \leftarrow g'^{\beta} (\mod \Gamma)$. Afterwards, compute $h \leftarrow g^r$, where $r \in_R [0, ..., \rho]$. Finally output $params \leftarrow (\Gamma, \rho, g, h)$.

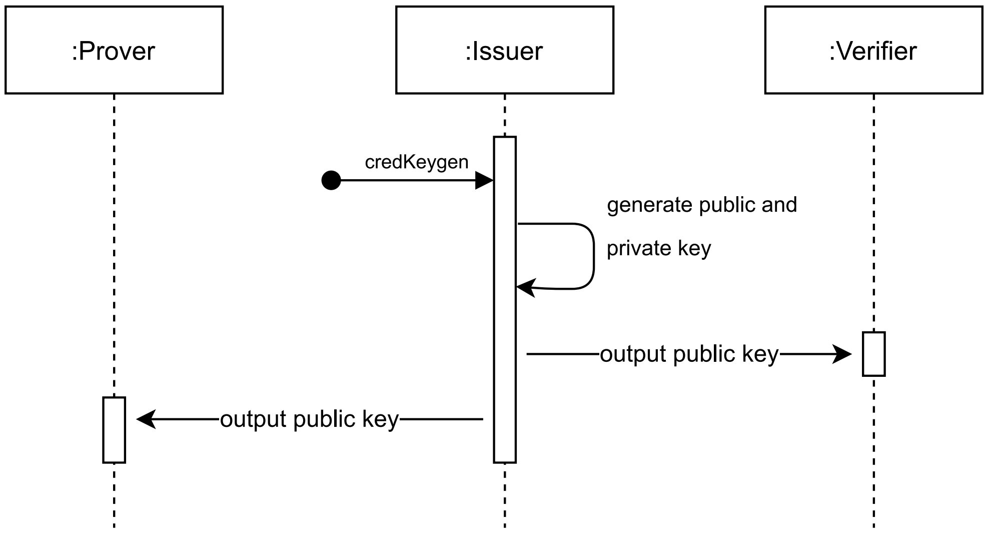


FIGURE 10: Sequence diagram of Idemix key generation

CredKeygen(params): The key generation process for Idemix is summarized in figure 10. The issuer generates a safe RSA key pair $n \leftarrow pq$ by first generating p and q, where $p \leftarrow 2p' + 1$ and $q \leftarrow 2q' + 1$. Then the issuer create parameters for the CL-signature by generating $S \in_R QR_n$ and $Z, R_1, ..., R_l \in_R \langle S \rangle$. The subgroup $\langle S \rangle$ is generated by S, and S must have order $\#QR_n = p'q'$. Afterwards the issuer selects $x_Z, x_{R_1}, ..., x_{R_l} \in_R [2, ..., p'q' - 1]$. These values are then used to compute $Z \leftarrow S^{x_Z}$ and $R_i \leftarrow S^{x_{R_i}} \forall \{1, ..., l\}$. Output the issuer public key $pk_I \leftarrow (n, S, Z, R_1, ..., R_l)$ and store the private key $sk_I \leftarrow (p, q)$.

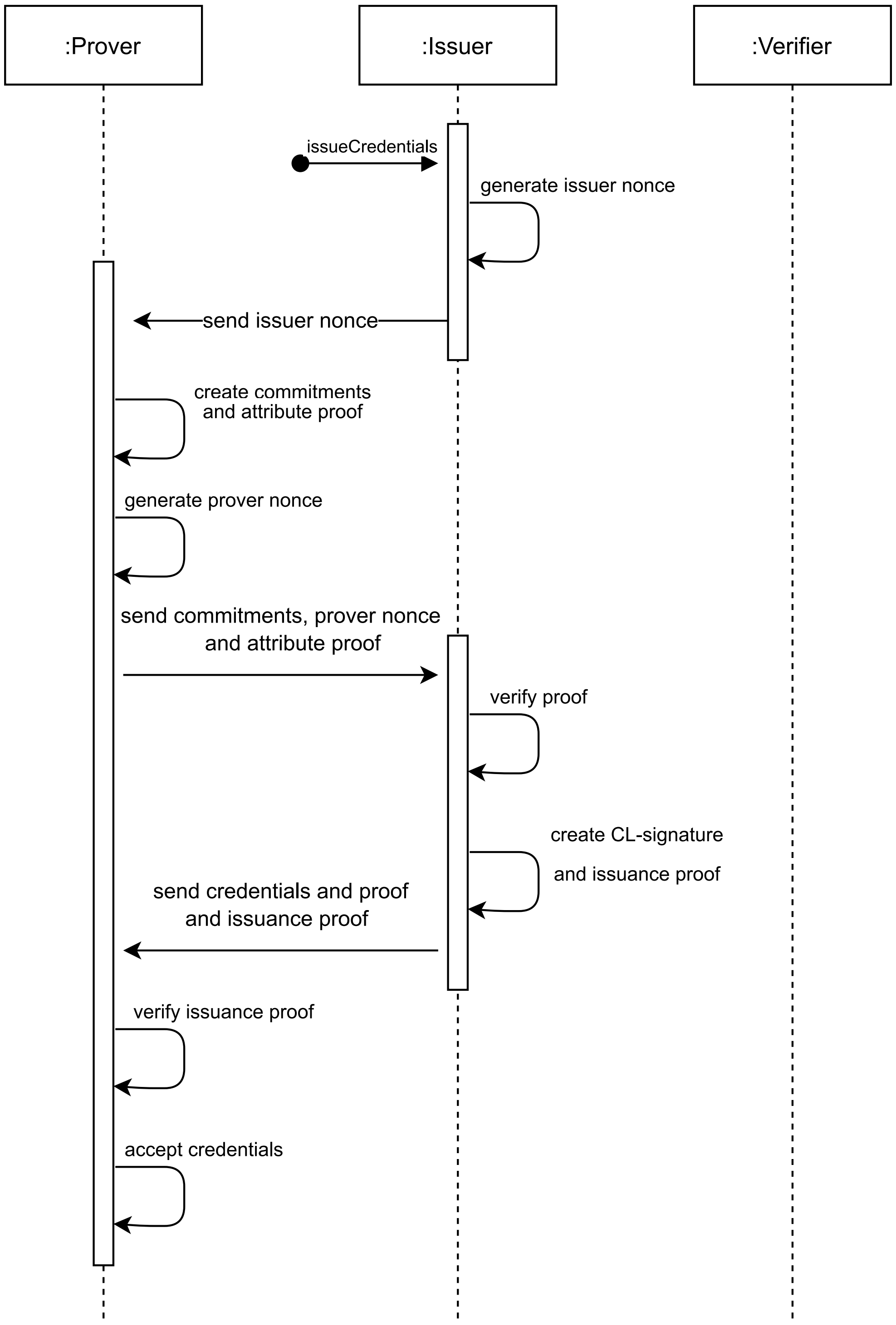

:Prover
:Issuer
:Verifier
issueCredentials
generate issuer nonce
send issuer nonce
create commitments and attribute proof
generate prover nonce
send commitments, prover nonce and attribute proof
verify proof
create CL-signature and issuance proof
send credentials and proof and issuance proof
verify issuance proof
accept credentials

FIGURE 11: Sequence diagram of Idemix credential issuance

Issuing Credentials: The credential issuance process for Idemix is summarized in figure 11. The issuer chooses a random nonce $n_1 \in_R \{0,1\}^{\ell_\phi}$, and sends it to the prover. This nonce is used to ensure the freshness of the prover's ZKP of attributes to be signed by the issuer. Then, for a block of messages $\overrightarrow{m}$ the prover generates a random $r_i \in_R \{0,1\}^{\ell_n+2\ell_\phi+\ell_H}$ and the commitment $C_i \leftarrow Z^{m_i}S^{r_i}(\bmod n)$ for each $i \in \{1,...,l\}$. The prover also generates an integer $v' \in_R \{0,1\}^{\ell_n+\ell\phi}$ required to compute $U \leftarrow S^{v'} \prod_{i=1}^{l} R_i^{m_i} (\mod n)$. Afterwards, the prover sends U, the commitments C_i for each $i \in \{1,...,l\}$, a new nonce called the prover's nonce $n_2 \in_R \{0,1\}^{\ell\phi}$, and the proof of knowledge

$$\begin{gathered}\pi := PK\{(\overrightarrow{m}, v', \overrightarrow{r}) : U \equiv S^{v'} \prod_{i=1}^{l} R_i^{m_i} \\ \wedge C_i \leftarrow Z^{m_i} S^{r_i} \forall i \in \{1...l\} \\ \wedge m_i \in \{0,1\}^{\ell_m + \ell_\phi + \ell_H + 1} \forall i \in \{1...l\}\}(n_1).\end{gathered}$$

to the issuer. If the proof verifies, the issuer selects a random prime number $e \in_R [2^{\ell_e - 1}, ...2^{\ell_e - 1} + 2^{\ell'_e - 1}]$, a random integer $\tilde{v} \in_R \{0,1\}^{\ell_v - 1}$, and computes $v'' \leftarrow 2^{\ell_v - 1} + \tilde{v}$. Then, the issuer computes $Q \leftarrow \frac{Z}{US^{v''} \prod_{i=1}^{l} R_i^{m_i}} (\mod n)$ and $A \leftarrow Q^{e^{-1} \mod p'q'} (\mod n)$. Finally, the issuer sends the CL-signature (A, e, v'') and the proof

$$\pi := PK\{(e^{-1}) : A \equiv Q^{e^{-1}} (\mod n)\}(n_2)$$

to the prover. The nonce n_2 provided by the prover is used to ensure the freshness of the ZKP of the CL-signature. If the proof validates, the prover accepts the CL-signature on their given attributes.

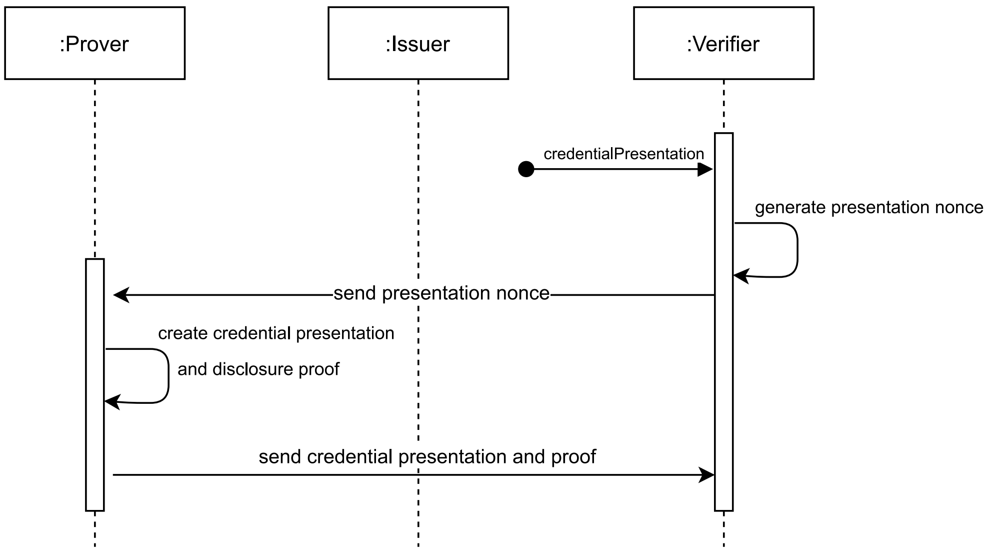

:Prover
:Issuer
:Verifier
credentialPresentation
generate presentation nonce
send presentation nonce
create credential presentation
and disclosure proof
send credential presentation and proof

FIGURE 12: Sequence diagram of Idemix credential presentation

Credential Presentation: The credential presentation process for Idemix is summarized in figure 12. The verifier first sends the presentation nonce $n_1 \in_R \{0,1\}^{\ell_\phi}$ to the prover. Then, for each hidden attribute in $\overrightarrow{m}$ that has its index represented as h, the prover generates randomized attribute values $\tilde{m_i} \in_R \{0,1\}^{\ell_m + \ell_\phi + \ell_H}$ for $i \in h$. The prover randomizes their CL-signature by calculating $r_A \in_R \{0,1\}^{\ell_n + \ell_\phi}$, $A' \leftarrow AS^{r_A}$, $v' \leftarrow v - er_A$, and $e' \leftarrow e - 2^{\ell_e - 1}$. Next, the prover selects the random integers $\tilde{e} \in_R \{0,1\}^{\ell'_e + \ell_\phi + \ell_H}$ and $\tilde{v}' \in_R \{0,1\}^{\ell'_v + \ell_\phi + \ell_H}$. These values are then used to calculate $\tilde{Z} \leftarrow A'^{\tilde{e}} S^{\tilde{v}'} \prod_{i \in h} R_i^{\tilde{m}_i} (\mod n)$. The prover also computes the challenge hash value $c \leftarrow H(\tilde{Z}, A', n_1, pk_I)$ and the response values $\hat{e} \leftarrow \tilde{e} + ce'$, $\hat{v}' \leftarrow \tilde{v}' + cv'$ and $\hat{m}_i \leftarrow \tilde{m}_i + cm_i \forall i \notin h$. These response values are called *s-values* in [13]. Finally, the values $(\tilde{Z}, A', \hat{e}, \hat{v}', \overrightarrow{\hat{m}}, c)$, are presented to the verifier along with the proof of knowledge

$$\begin{gathered}\pi := PK\{(e, \{m_i \forall i \in h\}, v) : \frac{Z}{\prod_{i \notin h} R_i^{m_i}} \equiv A^e S^v \prod_{i \in h} R_i^{m_i} (\mod n) \\ \wedge m_i \in \{0,1\}^{\ell_m + \ell_\phi + \ell_H + 2} \forall i \in h \\ \wedge e - 2^{\ell_e - 1} \in \{0,1\}^{\ell'_e + \ell_\phi + \ell_H + 2}\}(n_1).\end{gathered}$$

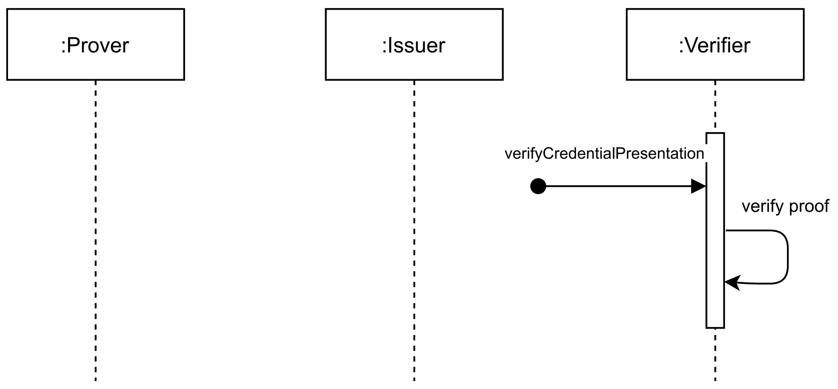


FIGURE 13: Sequence diagram of Idemix credential verification

Verify Credential Presentation: The credential verification process for Idemix is summarized in figure 13. In order to verify $(\tilde{Z}, A', \hat{e}, \hat{v}', \overrightarrow{\hat{m}}, c)$ the verifier first computes

$$\hat{T} \leftarrow \left(\frac{Z}{\prod_{i \notin h} R_i^{m_i} A'^{2^{\ell_e - 1}}}\right)^{-c} A'^{\hat{e}} S^{\hat{v}'} \prod_{i \in h} R_i^{\hat{m}_i} \mod n.$$

Afterwards, the verifier verifies the proof π by checking if the result of hashing $\hat{T}$ together with the public parameters pk_I and nonce n_1 equals to the challenge given by the prover in the proof π.

2.4.7 Keyed Verification Credentials From DDH

AMACs can be used to certify the authenticity of attributes by including the tag σ when plaintext attributes $\vec{m}$ are sent to the verifier. However, they can also be used in conjunction with commitment schemes and ZKPs to create an anonymous credential system. During credential issuance, the issuer creates commitments of the secret keys used to generate AMACs for the credentials. Then, these commitments are shared using ZKPs to the prover along with their AMACs in plaintext, to show that the AMACs were created using the proper parameters and secret keys. During credential presentation, the prover creates randomizes the AMACs on their attributes and makes commitments on them. In addition, they also create commitments on their hidden attributes. Then these commitments are sent to the verifier along with a ZKP of their randomized AMACs and attributes. The following are processes used in keyed verification credential based on $\mathsf{MAC}_{\mathsf{DDH}}$ [14].

Setup(λ): Compute $(g, h, q, G) \leftarrow \mathsf{Setup}_{\mathsf{DDH}}(\lambda)$, and output $params \leftarrow (g, h, q, G)$.

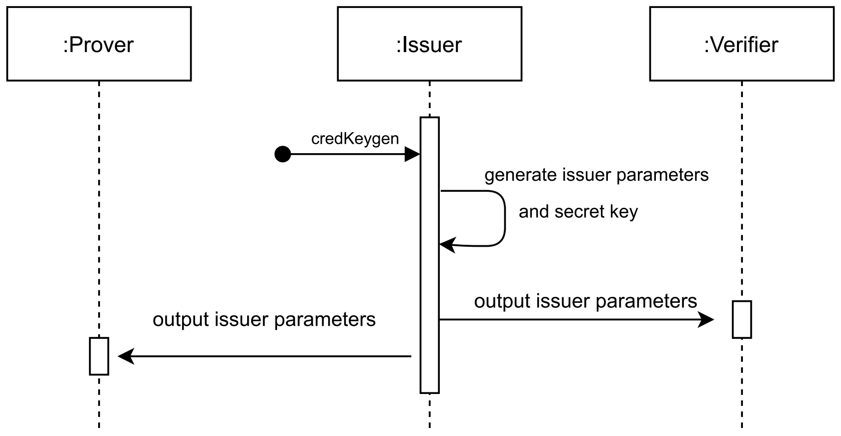


FIGURE 14: Sequence diagram of $\mathsf{AMAC}_{\mathsf{DDH}}$ key generation

CredKeygen(*params*): The key generation process for the $\mathsf{AMAC}_{\mathsf{DDH}}$ anonymous credential system is summarized in figure 14. The issuer computes the MAC secret keys $(\vec{x}, \vec{y}, z) \leftarrow \mathsf{KeyGen}_{\mathsf{DDH}}(params)$. Commit the secret values $(\tilde{x_0}, \tilde{y_0}, \tilde{z}) \in_R \mathbb{F}_q$ and create the commitments $C_{x_0} \leftarrow g^{x_0} h^{\tilde{x_0}}$, $C_{y_0} \leftarrow g^{y_0} h^{\tilde{y_0}}$, $C_z \leftarrow g^z h^{\tilde{z}}$. Then compute $X_i \leftarrow h^{x_i}$ and $Y_i \leftarrow h^{y_i}$ for each $i \in \{1, ..., n\}$. Finally, output the issuer parameters $iparams \leftarrow (C_{x_0}, C_{y_0}, C_z, \vec{X}, \vec{Y})$ and store the secret key $sk \leftarrow (\vec{x}, \vec{y}, z, \tilde{x_0}, \tilde{y_0}, \tilde{z})$.

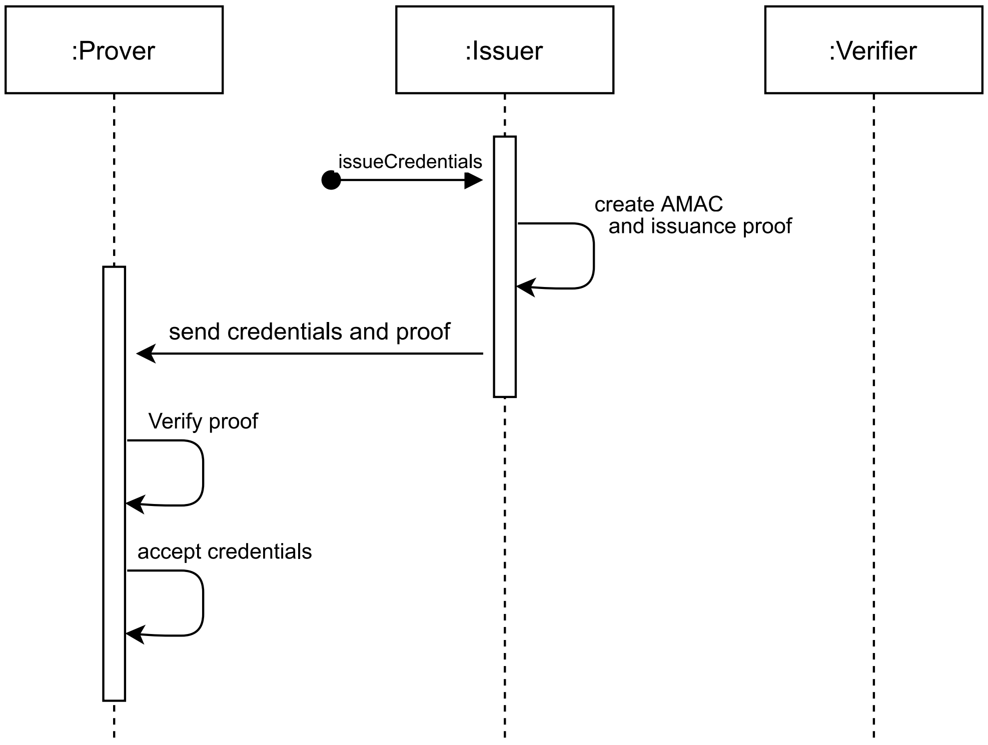


FIGURE 15: Sequence diagram of $\mathsf{AMAC}_{\mathsf{DDH}}$ credential issuance

Issuing Credentials. The credential issuance process for the $\mathsf{AMAC}_{\mathsf{DDH}}$ anonymous credential system is summarized in figure 15. To issue a credential on all known attributes $\vec{m}$ of the prover, the issuer computes $(\sigma_w, \sigma_x, \sigma_y, \sigma_z) \leftarrow \mathsf{MAC}_{\mathsf{DDH}}(sk, \overrightarrow{m})$, and sends it together with the proof π that shows the issuer's knowledge of the secret key required to compute $\sigma \leftarrow (\sigma_w, \sigma_x, \sigma_y, \sigma_z)$.

$$\pi := PK\{(\overrightarrow{x}, \overrightarrow{y}, z, \tilde{x_0}, \tilde{y_0}, \tilde{z}) : Verify_{DDH}(sk, \overrightarrow{m}, \sigma)$$
$$\wedge C_{x_0} \leftarrow g^{x_0} h^{\tilde{x_0}} \wedge C_{y_0} \leftarrow g^{y_0} h^{\tilde{y_0}} \wedge C_z \leftarrow g^z h^{\tilde{z}}$$
$$\wedge \mathrm{X}_i \leftarrow h^{x_i} \wedge Y_i \leftarrow h^{y_i} \forall i \in \{1, ..., n\}\}.$$

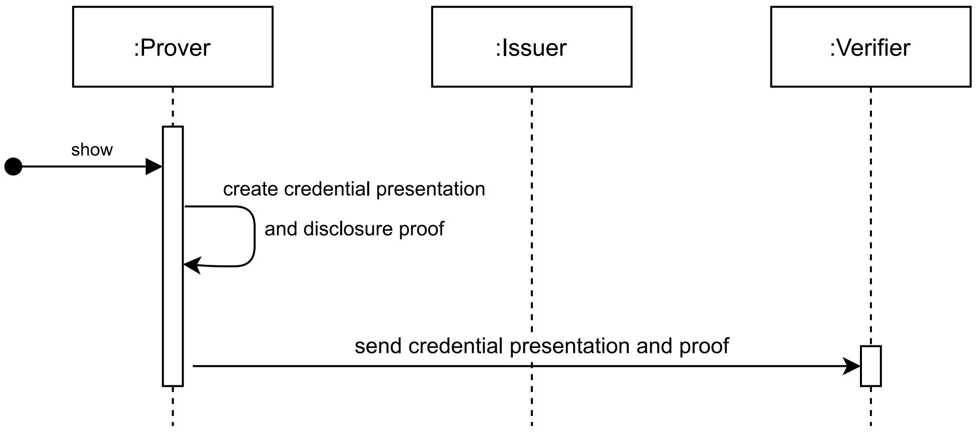

:Prover
:Issuer
:Verifier
show
create credential presentation
and disclosure proof
send credential presentation and proof

FIGURE 16: Sequence diagram of $\mathsf{MAC}_{\mathsf{DDH}}$ credential presentation

Show$(params, iparams, \phi, cred, \{m_i\}_i^n)$: The credential presentation process for the $\mathsf{AMAC}_{\mathsf{DDH}}$ anonymous credential system is summarized in figure 16. The prover chooses $r, r_x, r_y, z_1, ..., z_n \in_R \mathbb{F}_q$ and parses $cred \leftarrow (\sigma_w, \sigma_x, \sigma_y, \sigma_z)$. It randomises the credential by calculating $\sigma_w \leftarrow \sigma_w^r, \sigma_x \leftarrow \sigma_x^r, \sigma_y \leftarrow \sigma_y^r, \sigma_z \leftarrow \sigma_z^r$. Then, computes $\{C_{m_i} \leftarrow \sigma_w^{m_i} h^{z_i}\}_{i=1}^n, C_{\sigma_x} \leftarrow \sigma_x g^{r_x}, C_{\sigma_y} \leftarrow \sigma_y g^{r_y}, V_x \leftarrow g^{-r_x} \prod_{i=1}^n X_i^{z_i}$, and $V_y \leftarrow g^{-r_y} \prod_{i=1}^n Y_i^{z_i}$. The set of attributes $(m_1, ...m_n)$ satisfies a set of statements ϕ that should validate to true, according to their keys in *iparams*.

Afterwards, the prover sends $\sigma \leftarrow (\sigma_w, \sigma_z, C_{\sigma_x}, C_{\sigma_y}, V_x, V_y, \{C_{m_i}\}_i^n)$ together with the proof of knowledge π, which is computed as

$$\pi = PK\{\overrightarrow{m}, \overrightarrow{z}, -r_x, -r_y) : \phi(m_1, ...m_n) = 1$$
$$\wedge C_{m_i} = \sigma_w^{m_1} h^{z_1} \forall i \in \{1, ..., n\}$$
$$\wedge V_x = g^{-r_x} \prod_{i=1}^n X_i^{z_i} \wedge V_y = g^{-ryx} \prod_{i=1}^n Y_i^{z_i}\}.$$

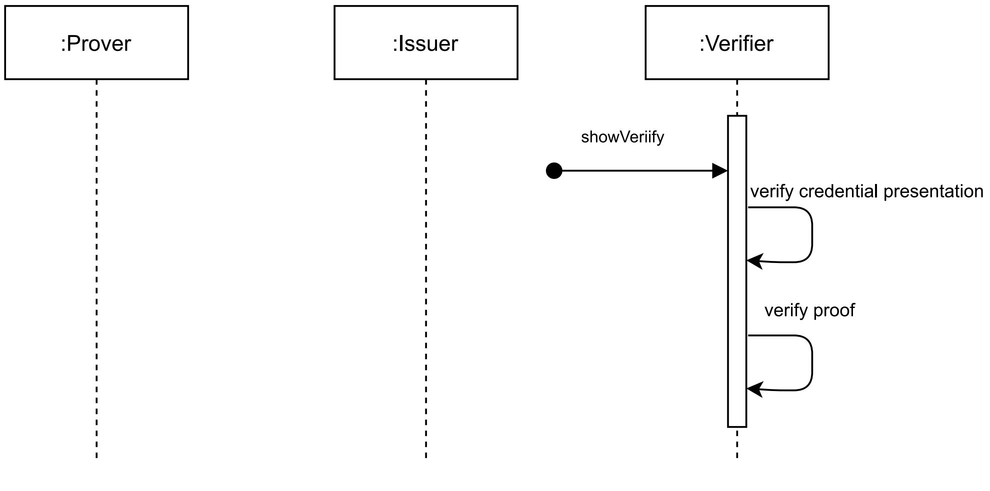


FIGURE 17: Sequence diagram of $\mathsf{MAC}_{\mathsf{DDH}}$ credential verification

ShowVerify$(params, iparams, \phi, \{x_i, y_i\}_i^n, z, \sigma, \pi)$: The credential verification process for the anonymous credential system using $\mathsf{AMAC_{DDH}}$ is summarized in figure 17. The verifier parses $\sigma \leftarrow (\sigma_w, \sigma_y, \sigma_z, V_x, V_y, \{C_{m_i}\}_i^n, C_{\sigma_x}, C_{\sigma_y})$, and verifies that $V_x = \frac{\sigma_w^{x_0} \prod_{i=1}^n C_{m_i}^{x_i}}{C_{\sigma_x}}$ and $V_y = \frac{\sigma_w^{y_0} \prod_{i=1}^n C_{m_i}^{y_i}}{C_{\sigma_y}}$. Afterwards, the verifier also verifies the proof π. If it is valid and if $\sigma_z = \sigma_w^z$, the algorithm accepts and outputs $(C_{m_1}, ..., C_{m_n})$, otherwise it will reject and output an invalid bit.

2.5 Attribute-Based Credentials

Attribute-Based Credentials (ABC) is an authentication method that allows provers to authenticate using secured digital credentials containing identifying attributes [4, 5]. The goal of ABC is to enable unlinkable, data-minimized transactions [6]. Attributes are properties about an entity that one wishes to share with a verifier, and credentials are the cryptographically signed containers of an attribute that hides their specific details [4]. The design of attributes and credentials used in ABC aligns with the W3C's verifiable credential recommendation [19].

ABC functions using the basis of anonymous credentials and ZKPs to provide the following security properties [4, 5]. Firstly, the issuer's digital signature contained in the credential provides a guarantee for the authenticity of the attributes within. Secondly, the integrity of the *freshness* of the attributes in the credential is also guaranteed by the issuer's signature. Thirdly, [4] implies that credentials are bound to a unique hardware (or digital wallet), hence they are not transferable. However, non-transferable credentials in this sense do not lock-in provers to a specific device, as they can simply revoke their credential linked to an old device and reinitialize new credentials with the same attributes in another. Thirdly, attribute-based credentials hide attributes with encryption, and provide issuer and multi-show unlinkability. Unlinkability of presented credential is achieved by omitting the issuer from the credential verification process, and encapsulating credentials in a verifiable presentation for the verification process.

2.6 Blockchain Technology for SSI

A blockchain, or a distributed ledger, is an append only data structure that is shared between multiple nodes within a peer-to-peer network [27]. In a blockchain network, validators or full nodes are responsible for appending the latest record to the blockchain [28]. The validator's permission model can be used to categorize different blockchain types. In a permissionless blockchain, any node can append the latest record to the blockchain. In a permissioned blockchain, only authorized nodes within the network can append the latest record to the blockchain. A consensus algorithm is used by the network of validating nodes to determine the latest record appended to the blockchain [28].

The technologies used in a blockchain such as hash algorithms and public key cryptography [28] is deemed applicable for SSI systems [27]. Hashed decentralized identifiers can be appended to a blockchain and maintained by multiple parties over the network. The blockchain can be a source of truth to verify an identity. Provers can present claims via decentralized identifiers appended to the blockchain, without being tethered to a centralized identity provider [27]. Public key cryptography used in a blockchain can be adapted to sign and validate verifiable credentials. Blockchain technology promises decentralization, tamper-resistance, inclusiveness, cost-efficiency, and user-control [27]. These promises are the reason why some experts encourages the development of blockchain-based SSI initiatives.

Blockchain-based identity management systems fall within two categories: self-sovereign identities and decentralized trusted identities [27]. Self-sovereign systems implements characteristics that strive to achieve the goals of SSI such as the portability and control of prover credentials. While decentralized trusted identity systems uses existing identity documents issued by a centralized authority (*e.g* identity cards or passports) and append claims towards them in a blockchain. Their reliance on existing identity documents, contradicts a self-sovereign system's requirement that requires them to be independent of a centralized authority [29].

3 Existing SSI Systems

This chapter will describe several existing SSI systems that are either a blockchain or non-blockchain approach. For each category, implementations that are of high-interest to this research questions posited in this thesis are discussed first, before mentioning other less-relevant implementations.

3.1 Blockchain Based Implementations

The Sovrin foundation created a project that leverages permissioned blockchains to share and verify anonymous credentials based on Idemix [27, 17]. According to their specifications [10, 30], Sovrin implements SSI policies by providing autonomy to the prover regarding credential access for verifying parties. An identity credential in the form of a DID document is issued on the request of the prover [20]. During the issuance process, the schema of a credential and revocation methods that describe the validation criteria of a credential are appended to the blockchain [20]. This schema can be used by verifiers to validate the attributes of a presented credential. A prover can prove ownership of identity attributes by presenting ZKPs of the credential to the verifier [20]. The code for the Sovrin project has been contributed as an open-source project called Hyperledger Indy [30].

The utilization of ZKPs to verify credentials in Sovrin is of high-relevance to this study. Sovrin's ability to be deployed in a private network allows evaluation in a controlled environment regarding the credential verification processes. Furthermore, Sovrin's code is open-source, allowing extensive exploration to its algorithms. There are numerous documents that describe Sovrin's functionality online, but the amount of spin-off projects and unfinished documentations that claimed to be part of either Sovrin or Hyperledger Indy are abundant.

An SSI project called uPort uses smart-contract applications created on the Ethereum blockchain [27, 17, 9]. The smart-contracts define operations regarding the issuance, and verification of identity credentials. Unique credentials, in the form of issued uPortIDs are not stored in the blockchain but are kept by their respective owners using a mobile client [27, 17, 9]. The registry smart-contract is used to store mappings of uPortIDs to identity attributes, that can possibly disclose metadata about a uPortID based on the JSON schema mapped to a specific identifier [27].

The developer documentation for uPort [31] is abstract and does not contain explanation about its proof mechanisms. Although several literature have documented the features of uPort in terms of its compliance with SSI principles [27, 17, 9], one of the research question posited in this thesis is intended to study a system's ZKP protocols. Therefore, studying uPort's algorithm without proper documentation would be an unfeasible task.

3.2 Non-blockchain Implementations

One ABC based implementation is the Dutch "I Reveal My Attributes" (IRMA) project [11]. IRMA uses Idemix public and private keys in its verifiable credential models. Issuers in IRMA uses Idemix private keys to sign credentials, while verifiers in IRMA uses Idemix public keys to validate IRMA credentials [32]. A signed credential is generated by the issuer on the request of the prover. IRMA does not have a centralized credential repository, and all credentials are stored by the prover in their client applications. Provers can present their credentials to verifiers by forming credential presentations that contains ZKPs of valid issuer CL-signatures [25]. The underlying ZKP protocol is based on solving discrete logarithm in a cyclic group of quadratic residue. IRMA consists of a server-side component used to verify and issue credentials, a mobile client or Go library that allows provers to manage and present issued credentials, as well as a JavaScript library used to generate QR codes that bridge operations between the server-side and frontend components [33].

The IRMA application components are deployable in a private setting, allowing extensive evaluation of its functionalities. Schemes or templates to test IRMA credentials can be created for testing purposes, but several templates have already been hardcoded to the IRMA application [34]. Furthermore, IRMA is still undergoing continued development, and has been utilized in various pilot projects around the Netherlands. It is hoped that the findings pertaining to IRMA's SSI properties and credential proving capabilities discussed in this thesis can improve the maturity of the project.

4 SSI System Evaluation of Sovrin and IRMA

This chapter will discuss the evaluation methodology and findings pertaining research question **RQ1**. The distinction between Sovrin and IRMA will be highlighted using a number of evaluation criteria described in an SSI system evaluation framework.

4.1 Methodology

In order to investigate **RQ1**, both Sovrin and IRMA are subjected to a comparison of their SSI properties. An evaluation framework [15] based on Allen's SSI definition [3] is used to compare the SSI properties of Sovrin and IRMA. For each of the 8 evaluation criteria, their description will first be given. Then, an analysis of Sovrin and IRMA according to the evaluation criterion will follow. This comparison will assist the explanation of the benefits and downsides of each SSI system. Several points describing Sovrin's properties in the findings are cited from [16].

4.2 Findings

The summary of each system's SSI characteristics can be seen in Table 3. The following text will elaborate the criteria and corresponding SSI properties in each system.

No.	Evaluation Criteria	Sovrin	IRMA
1	User control and consent	✓	✓
2	Privacy and protection	✓	✓
3	No trust in central authority	✓	✓
4	Portability and persistence	✓	✓
5	Transparency	✓	✓
6	Interoperability	✓	✓
7	Scalability	✓	✓
8	Usability	×	✓

TABLE 3: Summary of the SSI Evaluation of Sovrin and IRMA

1. User control and consent. Provers in an SSI system should always be able to access and share their credentials. The prover must give their consent before any personal information in the credential can be shared with verifiers.

In both SSI systems, credentials are solely kept by the prover. Sovrin credentials are either stored in the prover's wallet or entrusted to a Sovrin agent running on edge networks [35]. IRMA credentials are stored inside the prover's mobile application [11]. The same mobile application allows provers to confirm credential disclosure requests after scanning a QR code. In Sovrin, provers are given full autonomy regarding their credentials as they can select which attributes are revealed during credential disclosure [15].

2. Privacy and protection. The SSI system must advocate for the protection of the prover's privacy. Disclosure of identity claims or personal attributes should be set to the minimum amount required by verifiers. Furthermore, the cryptography used in the system must have sound security guarantees.

Besides using ZKPs during selective disclosure of attributes, unique pairwise-pseudonymous DIDs and public keys are established between Sovrin provers, issuers, and verifiers [15]. This unique DID, can be used as a private communication channel to exchange credentials between two parties for as long as they require [16]. IRMA does not use DIDs to establish a relationship between each participant. During IRMA issuance and proving protocols, communication between parties are done through secure JSON web token (JWT) sessions created by the IRMA server [16]. Both Sovrin and IRMA implement the Idemix anonymous credential system that guarantees its security in the documentation [13]. The amount of claims or attributes that verifier's can request, are limited to the attributes defined in either Sovrin's credential definition [35] or IRMA's credential scheme [34]. Thus, both SSI systems ensure that the attributes requested by a verifier during disclosure does not exceed the required amount.

3. No trust in central authority. Apart from the prover, credentials should not be stored by a centralized third-party.

Credential schemas are used as a model to define attributes in a credential, while credential definitions are credential schemas that contains the issuer's public keys [35]. These two aforementioned artefacts are not considered credentials, since they do not contain any identifying prover attributes. Sovrin does not have a central authority that stores issued credentials, and network policies are enforced to agents and stewards to prevent collusion in the network [15]. Schemas are stored on the blockchain, but credentials are kept in each prover's wallet or an agent appointed by the prover [35]. Credential schemas in IRMA are managed by the scheme manager that periodically updates their records from the global IRMA attribute index [1] or a self-hosted attribute index containing the schemas for a private organization [34]. IRMA credentials are stored in the prover's IRMA mobile application [11, 16]. As an added layer of security, some IRMA credentials require a verification via keyshare server that asks for the prover's PIN before a claim or attribute can be disclosed [36]. However, keyshare servers do not have access to the prover's credentials.

4. Portability and persistence. Credentials should be long-lasting or valid as long as its owner desires. All traces of information pertaining to the prover's credential, must be removable from the SSI system. Issued credentials should be transportable, in a sense that the owner can freely update or present their credentials. In addition, credential recovery methods should be available.

In case of theft, Sovrin credentials can be recovered by revoking old credentials or secret keys, and re-creating them [20]. Sovrin credentials stored in the prover wallet or agent can be freely used to interact with other verifying entities, and be deleted whenever the prover desires [35]. Besides authenticating credential issuance or disclosure protocols, the IRMA keyshare server can be used to revoke IRMA credentials [36]. The IRMA mobile application stores the prover's issued credential, and the prover can scan QR codes of IRMA sessions to receive new credentials or present them to a verifier [11]. IRMA credential backup and recovery options such as [37] are still in development.

[1] https://privacybydesign.foundation/attribute-index/en/

5. Transparency. The components, system functions, and algorithms used in an SSI system must be transparent and well-documented. In other words, they must be implemented using open source components. This also extends to the structure and life-cycle of credentials.

The anonymous credentials used in Sovrin and IRMA are based on Idemix [13]. Sovrin's protocols and schemas are based on open standards, and their system is created using the open source implementation of the Hyperledger Indy Project [15]. To manage the identity blockchain, the Sovrin foundation partnered with experts in the field of digital identity to become network stewards responsible for serving a blockchain filled with credential schemas and credential definitions [15]. The complete list of obligations and criteria of a steward are stated in the Sovrin whitepaper [10], along with a list of network stewards in [38]. Code for Hyperledger Indy is publicly available [2]. Similarly, IRMA uses open source standards, and has an extensive documentation [11, 25, 33, 34, 36]. The IRMA attribute index contains schemas issued by the Privacy By Design Foundation (PBDF), but private organizations may also host their own attribute index. The attribute index defines attribute parameters for IRMA credentials that can be issued and verified by different organizations. Besides documentation, code for the numerous IRMA components are available in their public github repository [3].

6. Interoperability. Digital identities used in an SSI system should be usable across various jurisdictions and system implementations. This implies that identities issued in an SSI system should be usable with different verifying parties.

The interoperability of credentials in each SSI system relies on the amount of issuers and verifiers that a prover can use their credentials with. The Sovrin foundation is actively recruiting new partners to join the role of agents or stewards [15], to increase the usability of their credentials with a variety of providers. The Sovrin whitepaper states that credentials are portable and usable across any device that can act as a Sovrin client [39]. Currently, IRMA credential schemes are listed in the online attribute index, where scheme managers can periodically update their records. New issuers adopting IRMA credentials can also have their credential definition published on the global index [16], or they can host their own attribute index in their own servers [34]. In conclusion, credentials in both Sovrin and IRMA are usable with other verifiers, as long as verifiers have access to credential schemas and definitions of each respective SSI systems.

7. Scalability. An SSI system must be highly scalable. The system functionalities should be able to operate effectively when resources are dwindling, and the number of users and load increases. Therefore, an SSI system must have various configuration options.

The blockchain nodes in Sovrin are divided into a group of validator nodes that can append new transactions to the blockchain, and a larger group of observing nodes hosting read-only blockchains that processes prover requests [15]. The Hyperledger Indy github repository contains dockerfiles describing a basic Sovrin network setup. Different dockerfile configurations can be used to deploy Sovrin blockchain nodes in a way that suits the network requirement. The IRMA documentation states that server components are made up of the backend and the IRMA server binary [40]. IRMA server components can be compiled into an executable binary [33]. Hence they can be deployed with performance enhancers such as load balancers or deployed as orchestrated container instances.

8. Usability. Finally, SSI systems must be usable and user-friendly. Independent of system or server configuration, provers should have access to an easy-to-use client application for interacting with their credentials. An SSI system that fulfills this criteria can encourage widespread adoption.

[2]https://wiki.hyperledger.org/display/indy

[3]https://github.com/privacybydesign

A previous study has reported that client-side implementations for Sovrin is still in its infancy [15]. To encourage developer activity, the Hyperledger Indy project provides the indy-client library that can be used to create client applications for provers, along with the Hyperledger Aries agent library that can be used to create Sovrin agents [41]. One notable Sovrin client created using these libraries is Orgbook of Canada [16] that implement identity wallets. On the other hand, IRMA provides a ready-to-use mobile application as the primary means for provers to use their credentials [11, 16]. In addition to the mobile application, the PBDF also provides a client library that developers can use to create new IRMA client applications.

4.3 Interpretation

The previous analysis and summary in table 3 describes how both Sovrin and IRMA possesses the traits required by an SSI system. A main feature that Sovrin lacks according to the evaluation framework is the absence of a user-friendly client application. A Sovrin client application may exist, but it is not publicly available except to partners that Sovrin have created a pilot project with. As an alternative, developers keen on creating a Sovrin client application are encouraged to use the Hyperledger Indy client libraries. However, this shortcoming does not invalidate Sovrin as an SSI system.

Sovrin possesses traits that still qualifies it as an SSI system. The privacy of prover credentials are guaranteed, credentials are not kept by a centralized authority, and the inner-workings of Sovrin are transparent and documented. The blockchain used as a source of record for credential schemas provide an extra layer of trust. Since these schemas are stored on the blockchain, issuers and verifiers can cross-validate a credential with the latest schemas in the blockchain. A prover receiving their credentials can also verify that their credential contains the required attributes and proper issuer keys. Furthermore, since multiple Sovrin validator and observer nodes manage a blockchain containing these schemas, decentralization is achieved, ensuring that no centralized authorities can manipulate them.

Using a blockchain to manage credential schemas comes with the drawback of additional system complexity. Additional server infrastructure is required to host and manage the blockchain nodes deployed alongside backend server components used by issuers and verifiers. These complexities within a blockchain-based SSI system affects the availability of client applications [15, 16]. An SSI system that solely relies on PKI like IRMA do not require additional system to manage a distributed ledger of schemas. At a minimum, an IRMA system can be created by deploying a single verifier and issuer backend server acting as an IRMA server or use REST API to interact with the public IRMA server [40]. This relatively easy deployment makes IRMA an attractive SSI system for organizations eager to quickly create a prototype SSI implementation.

Just like Sovrin, IRMA gives provers the ability to use of self-sovereign digital credentials. Once issued, credentials are stored in the prover’s mobile application and no other copies are available elsewhere. The extensive code repository and technical documentation allows developers to quickly create and test new prototypes, while also showing transparency to organizations concerned about user privacy. In addition, IRMA gains the upper hand on Sovrin by having a publicly available mobile application that is attractive to new users.

One drawback of IRMA is its reliance to the public attribute index governed by the PBDF, or a self-hosted attribute index [34]. These indexes are used by IRMA scheme managers as a trusted source for credential definitions. Thus, a dishonest custodian of the schema manager can possibly manipulate a credential definition without notice, invalidating credentials already issued to provers. In a development setting, credential definitions can be created by an issuer and used independent of the IRMA scheme index. Furthermore, if an organization desires to publish their credential definitions on the public attribute index, they must ask for the permission of the PBDF. Contrarily, the blockchain used by Sovrin to manage credential schemas and definitions is harder to manipulate. A dishonest custodian would have to append changes to the ledger that was entrusted to them, as well as the other ledgers managed by other custodians. This observation can be used to argue that IRMA lacks to Sovrin in terms of evaluation criterion 3, restricting SSI systems to have no trust in a central authority [15]. But the evaluation criterion stated that only *credentials* should not be kept by a central authority, it does not however cover credential schemas. If this criterion is extended to include credential schemas and credential definitions, the new summary of the SSI characteristics for each SSI system can be seen below in Table 4.

No.	Evaluation Criteria	Sovrin	IRMA
1	User control and consent	✓	✓
2	Privacy and protection	✓	✓
3	No trust in central authority	✓	×
4	Portability and persistence	✓	✓
5	Transparency	✓	✓
6	Interoperability	✓	✓
7	Scalability	✓	✓
8	Usability	×	✓

TABLE 4: Alternative summary of the SSI Evaluation of Sovrin and IRMA

5 Comparison of Credential Proving Methods in Sovrin and IRMA

This chapter will describe the research method and findings concerning the credential proving methods of Sovrin and IRMA, as posited in research question **RQ2**.

5.1 Methodology

In order to investigate **RQ2**, the documentation and implementation of both Sovrin and IRMA are studied in terms of their use of ZKPs to prove ownership of valid issuer signature on their credentials. The original intent was to discover the differences between the credential proving methods of each SSI system, and the possibility of protocol interchangeability. But, both Sovrin and IRMA implements Idemix anonymous credentials. Hence interchanging credential proving methods between Sovrin and IRMA was deemed unnecessary. Alternatively, the following section will briefly discuss how each SSI system implement Idemix.

5.2 Interpretation

In Idemix, ZKPs are exchanged during credential issuance and credential disclosure [13]. During the credential issuance phase, the prover presents an attribute ZKP while the issuer presents an issuance ZKP. Initially, the prover presents a ZKP certifying that their attributes (including their secret attribute) to be inserted in the credential are correct. Then, once the proof verifies, the issuer creates a credential with the provided attributes and signs the credential with their secret key. In addition to the credential, the issuer also presents a ZKP certifying that the issuer possesses knowledge of the correct issuer parameters to the prover. In the credential disclosure phase, the prover is the only participant that presents a ZKP. The prover sends a ZKP to the verifier displaying that they have valid issuer signatures on the hidden or revealed attributes.

The ZKP protocols implemented in Sovrin and IRMA follows the specifications described in the Idemix documentation [39, 25]. Both Sovrin and IRMA exchanges the same amount, and type of ZKP during the credential issuance and disclosure phase. In addition, nonces are used in both SSI systems to guarantee the freshness of each ZKP. The difference between the credential proving method of Sovrin and IRMA lies in the credential schema location. In both SSI systems, issuers and verifiers retrieve the credential schema and definition from their respective verifiable data registries. In Sovrin, credential schemas and definitions are appended on the blockchain [35]. In IRMA, scheme managers manage credential schema and definitions retrieved from the IRMA attribute index or a self-hosted attribute index [34]. In short, despite the different schema locations, the credential proving methods for Sovrin and IRMA are identical.

6 AMACs as Credential Proving Method for SSI Systems

In this chapter, the compatibility of AMACs as an alternative credential proving method for an open source SSI system will be investigated. Firstly, the methodologies of the tests conducted to investigate research question **RQ3** are elaborated. Then, the credential disclosure proving performance benchmarks of $\mathsf{AMAC}_{\mathsf{DDH}}$ and Idemix will be presented. The performance benchmarks will include the size of each disclosure proof and the time required to create them. In addition to the benchmarks, an interpretation of the results is also presented.

6.1 Methodology

In order to investigate **RQ3**, the size of the ZKPs and the execution times required to generate them during credential disclosure are compared between a system based on $\mathsf{AMAC}_{\mathsf{DDH}}$ and Idemix. A previous study suggests that the computation cost required to generate a ZKP during credential issuance and verification, is less than the cost of generating a ZKP when proving credentials [14]. Moreover, the need for an analysis on a credential system's proving cost is emphasized through proposals that require provers to use low-power equipment such as smart cards [4]. The disclosure proof size and calculation tests, will be done for a different number of attributes n and revealed attributes r.

The presented credential system based on $\mathsf{AMAC}_{\mathsf{DDH}}$ is implemented using Go. The implementation is created using Go to allow for a direct comparison when proving the same credential type with the Idemix system implemented in IRMA. The Idemix functions used in IRMA are defined in the Gabi library [4] which was also implemented in Go. Although not equivalent to comparing the proof creation performance of $\mathsf{AMAC}_{\mathsf{DDH}}$ in a full IRMA system, a comparison of the disclosure proof computation cost is still considered valid. This is because the same credential type with attributes represented as an array of BigInteger objects are used as input for the issuance and proving methods in both the Gabi library and the $\mathsf{AMAC}_{\mathsf{DDH}}$ implementation. The Gabi library will call the CreateDisclosureProof function, that contains the functionality required to create a credential disclosure proof in IRMA. This process can be seen in Figure 12 when the prover creates the credential presentation and disclosure proof. During the test, the nonce is randomly generated in the main file to simulate a prover that has received a nonce sent by verifier. Alternatively, the $\mathsf{AMAC}_{\mathsf{DDH}}$ implementation will create a credential disclosure proof by calling the implemented Show function as defined in section 2.4.7. This process is represented in Figure 16 when the prover creates the credential presentation and disclosure proof. In both Idemix and $\mathsf{AMAC}_{\mathsf{DDH}}$ tests, the emphasis is on measuring the disclosure proof creation time, hence the execution time for the credential presentation creation is disregarded.

[4] https://github.com/privacybydesign/gabi

The $\mathsf{AMAC}_{\mathsf{DDH}}$ implementation uses Go's built-in elliptic curve library based on elliptic curve P-256 as defined in FIPS 186-3[42]. In contrast, Idemix uses 2048-bit RSA modulus, with system parameters defined in [13]. The size of the disclosure proof is determined by summing up the bit lengths of the variables contained in the disclosure proofs. The variables in the disclosure proof includes the challenge value and responses used to represent the randomized credential values in the sigma protocol. In addition, the Idemix disclosure proof created using the Gabi library also contains the list of revealed attribute values. The $AMAC_{DDH}$ specification in section 2.4.7 does not explicitly combine the proof with the revealed attributes. The disclosure proof size of $AMAC_{DDH}$ presented in the findings includes the bit length of the revealed attributes as well. This will provide an equivalent comparison with the Idemix proof generated using the Gabi library. The sizes of the disclosure proof can be used with the disclosure proving time data, to roughly estimate the elapsed time to compute and send the proof to a verifier in a wireless network environment. The execution times for each proving method will be measured in milliseconds (ms). The disclosure proving time of each implementation will be measured for a number of attributes n with the values 1, 10, 20, 30, 40, and 50. Other than the first value which only measured the proving time for 1 hidden and revealed attribute, the others will measure the elapsed proving time for when all attributes are hidden, half revealed, and fully revealed. In addition, the disclosure proving times for a linearly incremented number of total attributes n with values between 0 and 50 will be measured for when all attributes are fully hidden and fully revealed. For each proving time test, the average execution time and standard deviation are taken from 100 iterations. The mean value and standard deviation are used to summarize quantitative data [43]. All the tests were executed on a computer with 2,6 GHz Intel I7-9750H CPU and 16GB of RAM.

6.2 Findings

(n,r)	Proof size		Proof time	
	$\mathsf{AMAC}_{\mathsf{DDH}}$	Idemix	$\mathsf{AMAC}_{\mathsf{DDH}}$	Idemix
(1,0)	1280 bits	7870 bits	1.14 ms	2.87 ms
(1,1)	1024 bits	7670 bits	0.6 ms	2.51ms

TABLE 5: Mean disclosure proof size and calculation time for a single attribute

The size of the disclosure proof for a single attribute produced in $AMAC_{DDH}$ and Idemix can be seen in Table 5. When the attribute is hidden, the size of the $AMAC_{DDH}$ proof is 1280 bits, which is smaller than the Idemix proof that is 7870 bits. The Idemix proof is larger because it contains a randomized CL-signature and challenge variable that has a bit length of 2048 bits each, along with three sigma protocol responses with a bit length of 594 bits, 2724 bits, and 456 bits. In comparison, the $AMAC_{DDH}$ proof consisted of a challenge variable, and four sigma protocol response variables, each with a bit length of 256 bits. When the single attribute is revealed, the size of each disclosure proof decreases because it omits the response variables related to the committed attributes presented in the proof. However, an additional 256 bit number is substituted to represent the revealed attribute. In an $AMAC_{DDH}$ proof, two variables related the committed value responses are subtracted. When added with the revealed attribute, it produced a disclosure proof size of 1024 bits. In the Idemix disclosure proof, the 456 bit number representing the committed attribute response is subtracted. Afterwards, the revealed attribute is added to the proof causing it to have a size of 7670 bits.

The disclosure proof calculation time for a single attribute in $\mathsf{AMAC}_{\mathsf{DDH}}$ and Idemix are also presented in Table 5. The presented benchmark results for a single attribute proof disclosure show that $\mathsf{AMAC}_{\mathsf{DDH}}$ computed a proof faster than Idemix. Idemix computed a disclosure proof in 2.87 milliseconds when no attributes are revealed, while $\mathsf{AMAC}_{\mathsf{DDH}}$ computed a proof in 1.14

milliseconds. When the number of revealed attributes are equal to the number of issued attributes, both proving methods performed faster than when attributes are hidden. When the attribute is revealed, the $\mathsf{AMAC_{DDH}}$ implementation is faster by over 0.5 milliseconds than $\mathsf{AMAC_{DDH}}$ when attributes are hidden. In contrast, Idemix with revealed attributes is faster by over 0.3 milliseconds than Idemix with hidden attributes.

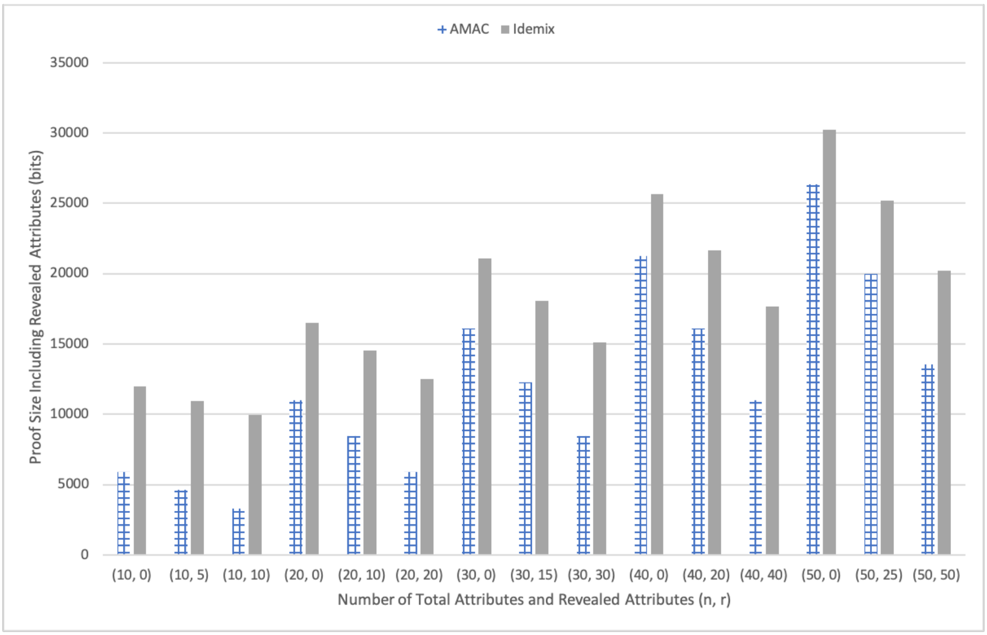


FIGURE 18: Disclosure proof size including revealed attributes for $\mathsf{AMAC_{DDH}}$ and Idemix with different number of attributes n and revealed attributes

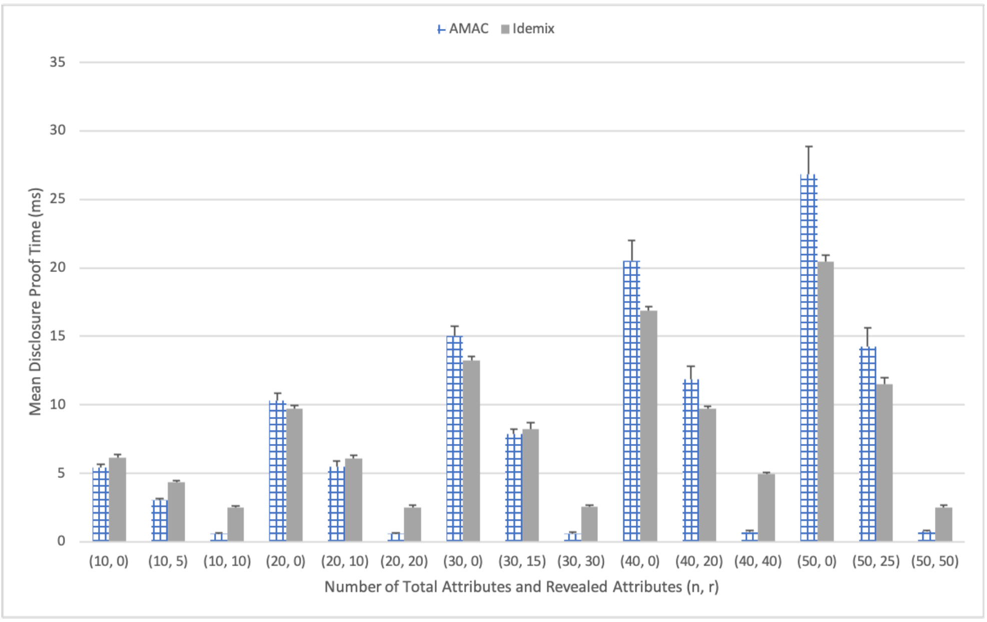


FIGURE 19: Mean disclosure proving times for $\mathsf{AMAC_{DDH}}$ and Idemix with different number of attributes n and revealed attributes r

Figure 18 shows the disclosure proof size produced in $AMAC_{DDH}$ and Idemix for various numbers of attributes with different ratios of revealed attributes. The trend of Idemix having a larger disclosure

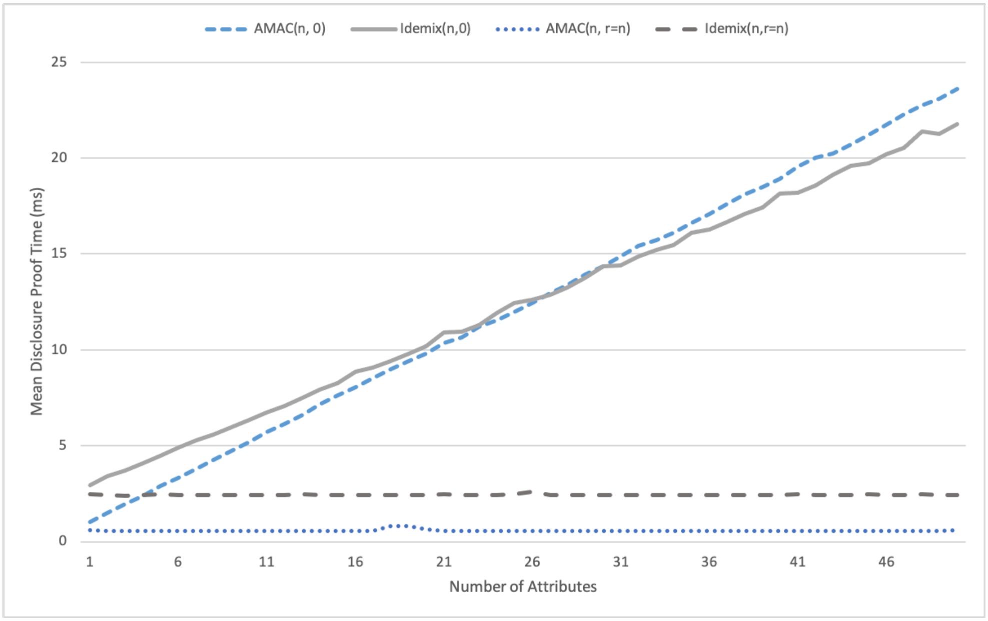


FIGURE 20: Mean disclosure proving times for $\mathsf{AMAC}_{\mathsf{DDH}}$ and Idemix

proof size seen in Table 5, is also present in Figure 18. In all cases, $AMAC_{DDH}$ produced a smaller discosure proof size than Idemix. For each number of total attributes, the credential size decreases when more attributes are revealed. The proof size increases as the number of attributes grow, because the number of response variables related to the hidden attribute also increases. As the number of total attributes increases, the difference between the proof size of $AMAC_{DDH}$ and Idemix diminishes. For example, the difference between the proof size of $AMAC_{DDH}$ and Idemix for when all 10 attributes are hidden is 6086. The difference becomes 3846 bits for 50 hidden attributes. The decrease in proof size can also be observed when half of the attributes and all of the attributes are revealed.

The results of the credential proving times can be seen in Figure 19. The results in Figure 19 show that when no attributes are revealed, $\mathsf{AMAC}_{\mathsf{DDH}}$ is faster than Idemix when generating proofs for 10 issued attributes. When half of the attributes are revealed, $\mathsf{AMAC}_{\mathsf{DDH}}$ is faster than Idemix until the number of issued attributes is 20. The $\mathsf{AMAC}_{\mathsf{DDH}}$ system performs better than Idemix when all attributes are revealed, displaying similar results to Table 5. The standard deviation of the $\mathsf{AMAC}_{\mathsf{DDH}}$ results in Figure 19 grows as the number of parameters n increases. On the other hand, the Idemix results have relatively constant standard deviations.

The credential proving times for $\mathsf{AMAC}_{\mathsf{DDH}}$ and Idemix in linear incremented number of attributes can be observed in Figure 20. When no attributes are revealed, $\mathsf{AMAC}_{\mathsf{DDH}}$ is faster during credential proving than Idemix. But Idemix begins to surpass $\mathsf{AMAC}_{\mathsf{DDH}}$ when the number of issued attributes reaches 26. Afterwards, $\mathsf{AMAC}_{\mathsf{DDH}}$ performed slower than Idemix for larger numbers of issued attributes. When the number of hidden and revealed attributes are equivalent, $\mathsf{AMAC}_{\mathsf{DDH}}$ is about 4 times faster than Idemix when computing a disclosure proof. Some lines in Figure 20 showed peaks in execution time due to unforeseen overhead in the Go program used for testing.

6.3 Interpretation

6.3.1 Disclosure Proof Size

As shown in Table 5 and Figure 18, $AMAC_{DDH}$ possesses a smaller disclosure proof size than Idemix. Besides the amount of variables in the produced proof, the size of the disclosure proof is mainly affected by the bit length of the parameters. The $AMAC_{DDH}$ implementation uses 256 bit numbers for addition and multiplication operations in the P-256 elliptic curve. The Idemix system uses 2048 RSA parameters and other system parameters defined in the documentation [13], that has larger bit lengths than the $AMAC_{DDH}$ elliptic curve parameters. Despite having a smaller proof size than Idemix, the results in Figure 18 show that the difference between the disclosure proof size of $AMAC_{DDH}$ and Idemix decreases as the number of total attributes grow. It is hypothesized that the smaller disclosure proof size of $AMAC_{DDH}$ would become negligible for a large number of attributes when $n > 50$, because the proof size would be similar to an Idemix proof. Nevertheless, for a small number of attributes such as $n = 10$ or $n = 1$, the efficient disclosure proof size of $AMAC_{DDH}$ would benefit an SSI system because it requires less storage resources.

6.3.2 Disclosure Proof Computation Time

In their proposal, the authors of $\mathsf{AMAC_{DDH}}$ provided benchmarks that shows how their method performed better than Idemix [14]. In their benchmarks, the number of issued attributes is fixed to 10, with varying numbers of revealed attributes. Findings presented in this thesis show that when the number of total issued attributes is 10, $\mathsf{AMAC_{DDH}}$ is faster than Idemix when all attributes are hidden, half revealed, and fully revealed. Thus, the results supports the performance boost mentioned in the aforementioned AMAC proposal. However, when the number of issued attributes is larger than 10, there is a point where the performance of $\mathsf{AMAC_{DDH}}$ begins to drop in comparison to Idemix. Hence, anonymous credential systems that require provers to prove a small number of attributes can harness the performance benefits offered by $\mathsf{AMAC_{DDH}}$. The threshold for the number of recommended attributes can be seen in Figure 20 where the $\mathsf{AMAC_{DDH}}$ line begins to dip under the Idemix line, which is when the number of attributes $n = 26$.

Despite using elliptic curves when implementing the $\mathsf{AMAC_{DDH}}$ credential system, the results show how the computation time for disclosure proofs can be slower than the finite field operations implemented in Idemix. Operations within an elliptic curve have lower computation costs when compared with operations in a finite field [7]. However, in some cases elliptic curve operations can be slower than their finite field counterparts. For example, the execution time for signing algorithms of Elliptic Curve Digital Signature Algorithm (ECDSA) is slower than RSA signatures, but its verification algorithm is faster [44]. Therefore, the performance implications of elliptic curve algorithms seems to be affected by the choice of the elliptic curve parameters and the implemented algorithm.

6.3.3 Disclosure Proving in a Network Environment

The findings related to the proof computation time are taken from an isolated test in a single machine. But the proving time for a single attribute in a network environment can be crudely estimated from the proof size and creation time results in Table 5. For example, assume that an SSI system is deployed using a 3G wireless network environment that has a maximum throughput of about 2Mbps $\approx 2 \times 10^7$bps [45]. The time required to transport an $AMAC_{DDH}$ disclosure proof for a single hidden attribute is $\frac{1280}{2\times10^7} = 6.4 \times 10^{-5}s \approx 6.4 \times 10^{-8}ms$. Afterwards, the transport time required for the $AMAC_{DDH}$ proof can be added to the existing proving time of 1.14ms shown in Table 5 to produce the total proving time. However, the previously mentioned calculation for the total proving time did not take into account the time required to initiate the network connection and wait for the acknowledgement response from the verifier to complete the connection.

6.3.4 AMAC as an SSI Proving Method

The findings showed that $\mathsf{AMAC}_{\mathsf{DDH}}$ is compatible for use with SSI systems. However, since it uses the same issuer private key to issue and verify credentials, issuer private keys must be shared with verifiers to allow credential validation in the SSI system. Results of the previous experiments demonstrated how a credential with its attributes represented as an array of BigInteger objects following IRMA's credential format, is compatible with the disclosure proving method implemented in the $\mathsf{AMAC}_{\mathsf{DDH}}$ credential system that also represents attributes as an array of BigInteger. SSI systems such as IRMA that use Idemix key pairs do not require issuers to share their private keys with verifiers, since the public keys attached in credential definitions can be used to verify a credential. However, since verification in an AMAC credential system requires the issuer private key, key-sharing protocols to distribute private keys with verifiers should be considered if IRMA intends to implement the $\mathsf{AMAC}_{\mathsf{DDH}}$ credential system. A one-time key-sharing protocol can be executed between the issuer and verifier during the CredKeygen phase described in section 2.4.7, when MAC private keys for a prover's attributes are generated by the issuer. Thus, there are no additional computation and network cost introduced during the $\mathsf{AMAC}_{\mathsf{DDH}}$ credential verification phase since verifiers would have already possessed the issuer's private key.

7 Discussions

This chapter will reiterate the research questions posited at the introduction of this thesis and recall the findings discussed in chapters 4, 5 and 6 to examine whether each research question has been addressed. Afterwards, the limitations of this study and recommendations for future studies will be discussed.

7.1 Addressing Research Questions

This section will contain a discussion addressing each research question formulated in the introduction.

- **RQ1:** What is the difference between Sovrin and IRMA's SSI implementation?
  - **SQ1.1:** What are the benefits and downsides of a blockchain SSI system?
  - **SQ1.2:** What are the benefits and downsides of non-blockchain SSI system?

After evaluating Sovrin and IRMA using the SSI system evaluation framework in chapter 4, two distinctions between them can be made.

The first difference between Sovrin and IRMA's SSI implementation is the location of credential schemas and credential definitions. Sovrin stores schemas in a blockchain shared between each validator and observer nodes, while IRMA schemas are located in a specified scheme manager. Blockchain SSI systems can benefit from the decentralized schema ledger as it provides additional assurance against colluding schema custodians. However, managing the architecture of a blockchain SSI system introduces challenges that are not present in non-blockchain systems. Non-blockchain SSI systems have less deployment requirements, making it a viable choice for rapid prototyping. But, if credential schemas rely on attribute indexes managed by a single entity like in IRMA, a threat arises where organizations who have access to attribute indexes can manipulate credential schemas to disrupt the SSI system.

Another difference between Sovrin and IRMA is in terms of their usability. The SSI system evaluation framework [15] requires an SSI system to have user-friendly client applications. IRMA has a mobile application which can be used by provers to store and share their credentials. On the other hand, Sovrin does not have a publicly available client application. Both SSI systems provide client libraries in their respective code repositories, allowing developers to create client applications.

- **RQ2:** What is the difference between Sovrin's and IRMA's ZKP protocol? Are different ZKP protocols interchangeable between them?

In chapter 5, it was discovered that there were no apparent differences between Sovrin and IRMA in terms of their ZKP protocols. Both SSI systems are based on Idemix. Idemix anonymous credential systems exchange ZKPs of valid CL-signatures on attributes in the credential [13]. Since Sovrin and IRMA both share the same anonymous credential system, there was no need to exchange credential proving protocols between them.

- **RQ3:** Can keyed verification credentials based on algebraic MACs be implemented as proving mechanism for Sovrin or IRMA? What are the differences in performance between their current ZKP protocols and the newly implemented alternative method?

This study showed how anonymous credential systems based on AMACs is a compatible credential proving method for IRMA. Findings show that valid disclosure proofs can be computed from credentials with attributes represented as an array of BigInteger objects using IRMA's Idemix implementation, and the $\mathsf{AMAC}_{\mathsf{DDH}}$ implementation used in this study. The size of the disclosure proof produced by the $AMAC_{DDH}$ implementation is smaller than the disclosure proof produced using Idemix. The smaller bit lengths of the $AMAC_{DDH}$ system allows the production of more efficient-sized disclosure proofs than Idemix. However, the results presented in Figure 18 suggest that the size of the disclosure proof produced using $AMAC_{DDH}$ might be as big as an Idemix disclosure proof for larger number of attributes. In terms of the proving time, $\mathsf{AMAC}_{\mathsf{DDH}}$ was able to produce a proof faster than Idemix for a small number of attributes when all attributes are hidden. The performance gains of $\mathsf{AMAC}_{\mathsf{DDH}}$ become insubstantial when larger amount of hidden attributes are proved. The $\mathsf{AMAC}_{\mathsf{DDH}}$ credential system produced a disclosure proof faster than Idemix when the number of hidden and revealed attributes are equal. However, this means that the SSI system enforce provers to always disclose their attributes when presenting their ZKPs to verifiers.

Why do Sovrin and IRMA use Idemix instead of keyed verification credentials based on AMACs ?

The reason why Idemix is used for Sovrin and IRMA is because it aligns with their requirements. The anonymous credential systems discussed so far have different cryptography foundations. Idemix is based on CL-signatures, and AMACs are based on MACs. Signature schemes use asymmetric keys, while MACs use a symmetric key [7]. The lead developer of the IRMA project explained how the development team prefers asymmetric keys than symmetric approaches (Ringers, S., personal communication, January 17 2022). SSI systems like Sovrin and IRMA publishes credential definitions containing the issuer's public key. This allows verifiers to validate the ZKP in a credential using the public key attached in the credential definition. If these SSI systems use anonymous credentials based on AMACs instead of idemix, then secure key-sharing protocols are required to allow the distribution of issuer secret keys with verifiers. The security of key-sharing protocols must be ensured, because sharing issuer private keys introduces the threat of non-issuers producing an existential or strong existential forgery of signatures / MACs on a credential. Alternatively, AMACs can be used without a key-sharing protocol in an SSI system setting where the issuer and verifier are the same entity.

7.2 Limitations and Recommendations

The SSI system evaluation framework [15] used in this study expected researchers to conduct a qualitative comparison on SSI systems. A framework containing quantitative scoring rubrics would be better for such comparisons because it can provide definitive grades for each aspect. Furthermore, some of the evaluation points in the framework could have been extended to cover other aspects of an SSI system. As discussed in chapter 4, the evaluation criterion concerning an SSI system's non-reliance to a centralized authority specified that credentials must not be stored by a centralized entity. This definition does not extend to credential schemas. This criterion can introduce ambiguity since an argument can be made to classify both SSI systems for being reliant to a centralized authority, by mentioning how credential schemas are governed by the issuer and custodians of attribute indexes. If the schema is manipulated after credentials are issued, there is the possibility of verifiers rejecting credentials due to it not containing the attributes defined in the new schema. Another example in chapter 4 is how the ambiguous nature of some of the evaluation criterion can be used to invalidate IRMA's compliance to the SSI criterion regarding non-reliance with a centralized authority.

The discussions posited throughout this thesis have explored the different ZKP protocols used during credential disclosure in SSI systems. However, it has not studied the effect of implementing a different anonymous credential system such as $\mathsf{AMAC}_{\mathsf{DDH}}$ in a complete deployment of an SSI system. Future studies are urged to explore the performance implications of fully migrating the Idemix functions in Sovrin or IRMA with another anonymous credential system. Furthermore, the experiments conducted in chapter 6 were done in a single machine without considering the performance implications of a network setting. A crude method to estimate the proving time of each method was discussed to show how the total disclosure proving time can be calculated using the data presented in chapter 6. But the method was not pursued further because the preliminary result showed that the result's difference was negligible with the proving times presented in chapter 6. A more sophisticated effort in the future can involve measuring the throughput and bandwidth of multiple provers generating a disclosure proof to a verifier in the network. More data concerning the performance benchmarks of the ZKP performance is required. Another possibility of future study is to explore how alternative elliptic curves might affect the performance of $\mathsf{AMAC}_{\mathsf{DDH}}$. One of the authors of the AMAC proposal suggested that using optimized elliptic curves such as Curve25519 and FourQ, or optimizing the programming logic of AMACs in the same P-256 curve, would show improvements in the disclosure times mentioned in [14] (Zaverucha, G., personal communication, January 10 2022).

8 Conclusion

In conclusion, the best ZKP protocols used for open source SSI systems would depend on the use-case requirement. The findings in this thesis showed that digital signatures were adopted by some SSI systems because it allowed verifiers to conduct verification using the issuer's public key. Moreover, SSI Systems based on digital signatures have an extra layer of security because private keys are not shared with non-issuers. However, alternative ZKP protocols based on AMACs with symmetric keys can also be used if existential forgeries are not a problem, such as when the issuer and verifier entities are the same, or if a secure key-sharing protocol is already in place.

The evaluation posited in this thesis have compared how two practical open source SSI systems differ from each other in terms of their SSI properties. Blockchain and non-blockchain SSI systems essentially function the same way, but differ in how credential schemas are managed. Then, it is revealed that Sovrin and IRMA use the same Idemix anonymous credential system that exchanges

ZKPs of CL-signatures contained in a credential. Afterwards, despite having a larger proof size than $AMAC_{DDH}$, the proving performance of Idemix is faster than a system based on $AMAC_{DDH}$ when larger number of fully hidden attributes are involved.

References

[1] J. Greig, “Passport info and healthcare data leaked from indonesias covid-19 test and trace app for travellers.” [Online]. Available: https://www.zdnet.com/article/passport-info-and-healthcare-data-leaked-from-indonesias-covid-19-test-and-trace-app-for-travellers/

[2] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, “Decentralized identity: Where did it come from and where is it going?” *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[3] C. Allen, “The path to self-sovereign identity.” [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[4] G. Alpár and B. Jacobs, “Credential design in attribute-based identity management,” 2013.

[5] M. Koning, P. Korenhof, G. Alpár, and J.-H. Hoepman, “The abc of abc: an analysis of attribute-based credentials in the light of data protection, privacy and identity,” 2014.

[6] D. Baars, “Towards self-sovereign identity using blockchain technology,” Master’s thesis, University of Twente, 2016.

[7] N. Smart, *Cryptography Made Simple*, ser. Information Security and Cryptography. Springer, 2016.

[8] V. castaños, “Case study report: e-estonia,” European Commission Directorate-General for Research and Innovation, Tech. Rep., 02 2018.

[9] N. Naik and P. Jenkins, “uport open-source identity management system: an assessment of self-sovereign identity and user-centric data platform built on blockchain,” in *2020 IEEE International Symposium on Systems Engineering (ISSE)*, ser. Proceedings of the 2020 IEEE International Symposium on Systems Engineering (ISSE). Institute of Electrical and Electronics Engineers, Dec. 2020.

[10] A. Tobin and D. Reed, “Inevitable rise of self-sovereign identity,” The Sovrin Foundation, Tech. Rep., 09 2016.

[11] P. by design foundation, “Irma technical overview.” [Online]. Available: https://irma.app/docs/overview/

[12] M. Isaakidis, H. Halpin, and G. Danezis, “Unlimitid: Privacy-preserving federated identity management using algebraic macs,” in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, ser. WPES ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 139–142. [Online]. Available: https://doi.org/10.1145/2994620.2994637

[13] IBM, “Specification of the identity mixer cryptographic library (revised version 2.3.0),” 2010. [Online]. Available: https://github.com/isislovecruft/library--/blob/master/anonymity%20%26%20circumvention/Specification%20of%20the%20Identity%20Mixer%20Cryptographic%20Library.pdf

[14] M. Chase, S. Meiklejohn, and G. M. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[15] A. Satybaldy, M. Nowostawski, and J. Ellingsen, *Self-Sovereign Identity Systems: Evaluation Framework.* Springer, 03 2020, pp. 447–461.

[16] J. Nauta and R. Joosten, "Self-sovereign identity: A comparison of irma and sovrin," The Netherlands Organization for Applied Scientific Research, Tech. Rep., 07 2019.

[17] N. Naik and P. Jenkins, "Self-sovereign identity specifications: Govern your identity through your digital wallet using blockchain technology," in *Proceedings - 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2020*, ser. Proceedings - 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2020. United States: IEEE, Jul. 2020, pp. 90–95.

[18] K. Cameron, "The laws of identity," 2005. [Online]. Available: https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf

[19] "Verifiable credentials data model 1.1." [Online]. Available: https://www.w3.org/TR/vc-data-model/

[20] D. Hardman, "How dids, keys, credentials, and agents work in sovrin," 2018. [Online]. Available: https://sovrin.org/wp-content/uploads/2019/01/How-DIDs-Keys-Credentials-and-Agents-Work-Together-in-Sovrin-131118.pdf

[21] "[editor's draft] verifiable claims working group frequently asked questions." [Online]. Available: http://w3c.github.io/webpayments-ig/VCTF/charter/faq.html

[22] "Decentralized identifiers (dids) v1.0." [Online]. Available: https://www.w3.org/TR/did-core/

[23] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, p. 1030–1044, Oct. 1985. [Online]. Available: https://doi.org/10.1145/4372.4373

[24] M. Drijvers, "Efficient delegation of idemix credentials," Master's thesis, Radboud University Nijmegen, 2014.

[25] P. by design foundation, "Irma zero-knowledge proofs." [Online]. Available: https://irma.app/docs/zkp/

[26] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2576, 01 2002, pp. 268–289.

[27] P. Dunphy and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security Privacy*, vol. 16, no. 4, pp. 20–29, 2018.

[28] NIST, "Blockchain technology overview," U.S. Department of Commerce, Washington, D.C., Tech. Rep. National Institute of Standards and Technology Internal Report 8202, 2018.

[29] K. Nyante, "Secure identity management on the blockchain," Master's thesis, University of Twente, 2018.

[30] P. J. Windley, “An overview of self-sovereign identity: the use case at the core of hyperledger indy,” 2019. [Online]. Available: https://www.hyperledger.org/blog/2019/05/01/an-overview-of-self-sovereign-identity-the-use-case-at-the-core-of-hyperledger-indy

[31] “uport developer portal.” [Online]. Available: https://developer.uport.me/

[32] “Irma technical documentation.” [Online]. Available: https://credentials.github.io/docs/irma.html

[33] P. by design foundation, “Getting started.” [Online]. Available: https://irma.app/docs/getting-started/

[34] ——, “Irma schemes.” [Online]. Available: https://irma.app/docs/schemes/

[35] A. Tobin, “Sovrin: What goes on the ledger?” 2017. [Online]. Available: https://sovrin.org/wp-content/uploads/2017/04/What-Goes-On-The-Ledger.pdf

[36] P. by design foundation, “Irma keyshare protocol.” [Online]. Available: https://irma.app/docs/keyshare-protocol/

[37] I. Derksen, “Backup and recovery of irma credentials,” Master’s thesis, Radboud University Nijmegen, 2019.

[38] “Stewards.” [Online]. Available: https://sovrin.org/stewards/

[39] D. Reed, J. Law, and D. Hardman, “The technical foundations of sovrin,” The Sovrin Foundation, Tech. Rep., 09 2016.

[40] P. by design foundation, “What is irma?” [Online]. Available: https://irma.app/docs/what-is-irma/

[41] “Hyperledger aries.” [Online]. Available: https://github.com/hyperledger/aries

[42] NIST, “Digital signature standard (dss),” U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publication 186-3, 2009. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf

[43] R. D. D. Veaux, P. Velleman, and D. E. Bock, *Stats: Data and Models, Global Edition, 4th Edition*. Pearson, 2016.

[44] N. J. G. Saho and E. C. Ezin, “Comparative study on the performance of elliptic curve cryptography algorithms with cryptography through rsa algorithm,” 2020.

[45] C. Beard and W. Stallings, *Wireless Communication Networks and Systems, eBook, Global Edition*. Pearson Education, 2016. [Online]. Available: https://books.google.nl/books?id=LJ5VCwAAQBAJ