A privacy-preserving tamper-evident revocation mechanism for verifiable credentials

Li Xu

Delft University of Technology

TU Delft

A privacy-preserving tamper-evident revocation mechanism for verifiable credentials

by

Li Xu

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday, June 28th, 2022 at 10:00 AM.

Student number:	5235952	
Project duration:	September 20, 2021 – June 28, 2022	
Thesis committee:	Dr. Zekeriya Erkin,	TU Delft, supervisor
	Dr. Zaid Al Ars,	TU Delft
	Dr. Ir. Oskar van Deventer,	TNO
	Dr. Ir. Johan Pouwelse,	TU Delft
	T. Li Msc,	TU Delft, daily supervisor

TUDelft

Preface

Working on this thesis turned out to be a long journey. Fortunately, I did not have to walk it alone but was helped by many others along the way. This is why I want to take this chance to thank those that contributed to this thesis.

First and foremost, I want to thank my supervisor Zeki and my daily supervisor Tianyu for sticking with me throughout the journey. I can not finish this work without their feedback, personal advice, and motivating words. Especially, I want to thank Zeki for teaching me the charisma of research. Zeki taught me the importance of critical thinking, communication and presentation. I always enjoy the cake time with "Zeki's gang". I also want to thank Tianyu for helping so much in writing and presentation; without Tianyu's help, I may still write paper-like Github Readme files.

I am very happy and grateful to meet Jelle, Ozzy, Asli, Armin, and Dieuwer during the journey. The lunchtime and coffee talks I had with you made this journey much more enjoyable. A special thanks to Jelle, who helped me a lot in implementation. And I want to thank Zaid, Oskar and Johan for taking the time to review my work as committee members.

Finally, I want to thank my family and friends for their support as I completed my thesis and master's degree.

To conclude the last nine months in short sentences is difficult, but I will remember the pains and gains in my mind forever. I am happy that with the help of so many nice guys, I finally finished the first research work by myself. And I hope this is the first work of my many research works in the future.

Li Xu
Delft, June 2022

Abstract

Third-party verified credentials (e.g. passports, diplomas) are essential in our daily life. The usage of third-party verified credentials bring us convenience in authentication. The Verifiable Credential (VC) data model is a new standard proposed by the W3C association to ease the expression and verification of third-party verified credentials on the Internet. The issuance and presentation of verifiable credentials are tamper-evident and privacy-preserving by design. However, the current verifiable credential data model lacks an explicit revocation design that guarantees the secure operations of the system. The lack of a revocation mechanism significantly limits the application of verifiable credentials.

This thesis studies the revocation mechanisms of existing verifiable credential implementations. The existing revocation mechanisms are either tamper-evident or privacy-preserving. None of them can achieve the two properties together. To evolve the revocation mechanism to be both tamper-evident and privacy-preserving by design, we propose a new method which combines the BBS+ signature, a cryptographic accumulator and the blockchain. Our design enables the verifier to verify the presented credential's revocation status without compromising the credentials holders' privacy.

We implement a proof-of-concept of our revocation mechanism to show it is practical in the real world. The experimental results show that after adding our revocation mechanism, the presentation time of a five-attribute credential changes from 22.22ms to 62.11ms (+39.89ms), and the verification time changes from 13.36ms to 44.56ms (+31.86ms). Moreover, the scalability analysis shows that our revocation mechanism can satisfy the need for revocation in the real world.

Contents

1 Introduction

1.1. Third-party verified credentials

Third-party verified credentials are certificates issued by trustworthy authorities. We can prove we satisfy specific requirements to pass authentications using a third-party verified credential. For example, we show driver's licenses to assert we can operate a vehicle, present our diplomas to state our education level, and use passports to travel across countries. Exploiting these credentials provides convenience to us when used in the physical world. As the Internet has become more indispensable, we also want to enjoy the same comfort on the Web. However, two factors make it challenging to use third-party verified credentials on the Internet.

The first factor is the difficulty in presenting third-party verified information on the Internet. The lack of a uniform presentation format and a consistent verification method makes employing third-party verified information on the Internet ineffective. To address it more clear, let's take the Spotify student discount as an example. If a student wants a student discount from Spotify, she must upload the photo of her student ID to SheerID[1] to verify that she is indeed a student. Because there are a significant number of schools worldwide, and their student IDs are in different formats, it is incredibly challenging to write a program to censor these uploaded student IDs. In other words, these uploaded student ID photos are not machine-verifiable. Thus, SheerID checks the validity of uploaded student IDs manually[2]. The participation of humans will increase the processing time and the risk of audit errors, which will make the verification unproductive.

The second factor is the concern of privacy leakage. Using paper-based credentials will expose all the information included in the credential to the verifier, although we only need to disclose part of that information in most cases selectively. Specifically, when we use our ID to buy alcohol, the seller only needs our age; nonetheless, the ID card exposes our name, date of birth, address and other private information to the seller. Malicious sellers may store this personal information and use it to correlate data from other platforms, which is a severe privacy concern. Thus, paper-based credentials are not privacy-preserving. What's worse, the persistence of digital data and the ease with which disparate digital data sources can be collected and correlated make the privacy concerns even more severe when using digital third-party verified credentials on the Internet.

1.2. Verifiable Credentials

To ease the presentation of third-party verified credentials on the Internet and relieve the privacy concerns, the W3C proposes the Verifiable Credential data model. The verifiable credential data model provides a mechinism to express third-party verified credentials in a way that is cryptographically secure, privacy respecting and machine-verifiable[3]. There are two components in the verifiable credential data model: the verifiable credentials and the verifiable presentations. Verifiable crendentials and verifiable presentations are encoded in the JSON-LD[4] format, providing a good machine-readability while ensuring global interoperability between a heterogeneous set of software systems.

The general scheme of verifiable credentials is shown in figure 1.1. A verifiable credential is composed of credential metadata, claim, and proof. Credential Metadata refers to the data used to describe properties of the credential, such as the issuer, the expiry date and time, a representative time and so on.

A claim is a statement about a subject. Claims are expressed using **subject-property-value** relationship. For example, the claim "Peter-studentOf-TU Delft" expresses that peter is a student of TU Delft. Proofs are the zero-knowledge proof that proves the issuer signs the claims. The addition of zero-knowledge proof techniques and cryptographic signature make the verifiable credentials more tamper-evident than their physical counterparts. Here, tamper-evident means that the verifiable credentials can resist forgery and cannot be modified after the issuance.

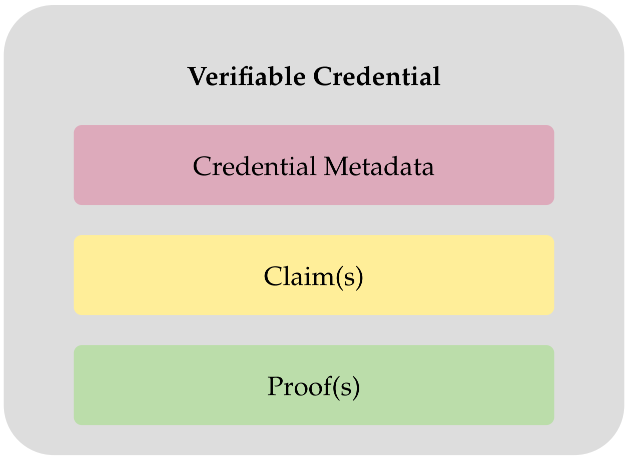

Figure 1.1: Basic components of verifiable credential

A verifiable presentation is the expression of a subset of one or more credentials. The basic scheme of verifiable presentation is shown in figure 1.2. The verifiable presentation has three components: presentation metadata, verifiable credential(s), and proof(s). Presentation metadata explains the usage of the presentation (e.g., the presentation is used to pass the age check when buying alcohol). The proofs in the verifiable presentation contain data required to check the validity of the presented claims. While using verifiable credentials to generate the verifiable presentation, the verifiable credential holder can choose to disclose part of the claims contained in the verifiable credentials. This is called the selective disclosure of verifiable credentials. The selective disclosure enables the holder to hide private information when presenting the verifiable credentials, enhancing the privacy of expressing credentials on the Internet.

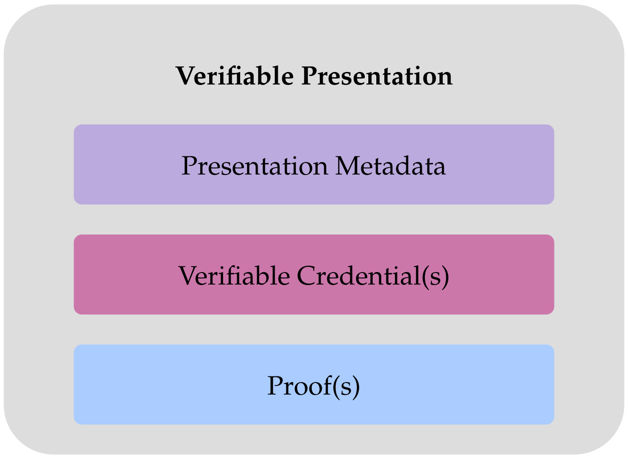

Figure 1.2: Basic components of verifiable presentation

Figure 1.3 shows the basic ecosystem of the verifiable credential data model. The ecosystem specifies the related actors in the verifiable credential data model and the supported operation. There are four actors and three operations in a verifiable credential ecosystem. The four actors are: issuer, holder, verifier and the verifiable data registry. The three operations are: issuance, presentation and verification. Issuance refers to the process that the issuer issues the verifiable credential to the holders. Presentation describes the procedure that holders use verifiable credentials to generate verifiable presenations. Verification indicates the verifiers check the validity of the verifiable presentations.

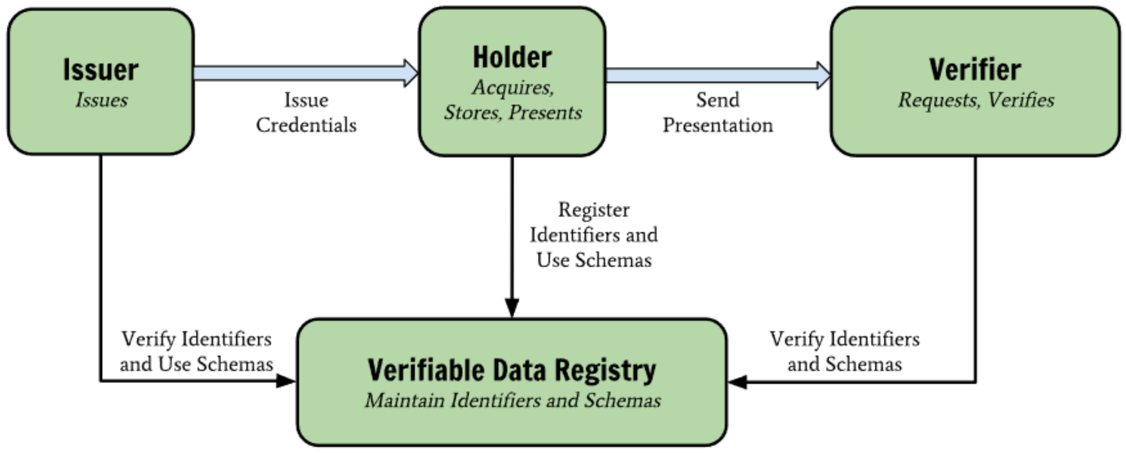
Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verify Identifiers and Schemas

Figure 1.3: Ecosystem of verifiable credentials

1.3. Revocation

Aside from issuance, presentation and verification, a credential system can also support the revocation operation. Revocation allows the issuer to rescind the validity of the issued credentials when the user misbehaves or the credentials are broken (e.g. stolen or lost). In a system with revocation, a credential will have two revocation statuses after issuance: revoked or non-revoked. Only non-revoked credentials can pass the verification. The term “Revocation mechanism" refers to the process how revocation is operated in the system. A revocation mechanism includes the following operations:

- Revoking a credential.
- Updating a credential’s revocation status
- Verifying a credential’s revocation status

The employment of revocation can be an enabler to guarantee the secure operation of credentials. Nowadays, credential fraud has become an unignorable problem in our society. Criminals and terrorists may use lost and stolen credentials to commit fraud or terrorist attacks. Statistics show that, by 2015, there were 34,085,965 lost and stolen passports in Europe, which caused a passport fraud “epidemic" around Europe[5]. According to Europol Director Rob Wainwright, this credential fraud facilitates cross-border terrorism[5], which has been a severe menace to public safety. Revocation is an ideal solution to credential fraud as the issuer can invalidate the stolen or lost credential to prevent further malicious behaviors. Therefore, adding a revocation mechanism to the credential system is vital.

1.4. Concerns on adding revocation mechanisms for Verifiable Credentials

Currently, the official verifiable credential data model does not provide a standard design for the revocation mechanism. The lack of a revocation mechanism limits the application of verifiable credentials in the scenario that needs a rigid validity check (e.g. applying loan from the bank). Nonetheless, adding a revocation mechanism to the verifiable credential data model is not an open-and-shut operation. Two concerns making the addition of revocation mechanisms challenging:

Privacy leakage. The risk of privacy leakage lies in the verification of revocation status. In the traditional revocation mechanism such as Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP), the verifier use identifiers such as the credential’s serial number to check the credential’s revocation status. The use of identifiers violates the selective disclosure of the verifiable credential data model.

Replacement Fraud. The problem of replacement fraud is a potential consequence of the anonymity of verifiable presentation. If the verifiable presentation reveals no personal information about the presented credentials, it is difficult to bind a credential with its associated revocation status. There are chances that malicious users can fool the verifier by replacing the revocation status of a revoked credential with the one from a non-revoked credential. Moreover, this is another form of replacement fraud, in which the attackers can hack in the central server to replace the revocation information with their own version.

To get over the abovementioned concerns, we need to design a revocation mechanism that can check whether an anonymous credential is revoked and avoid the replacement fraud. To put it another way, the revocation mechanism for verifiable credentials should be privacy-preserving and tamper-evident. Therefore, I structure my research question as:

> ***How to design a privacy-preserving tamper-evident revocation mechanism for verifiable credentials?***

1.5. Decomposition of Research Question

The first task of our research question is to find a privacy-preserving way to check the verifiable credential's revocation status and find a solution for the replacement fraud. Therefore, the first two sub-question are:

RQ1 How to achieve privacy-preserving in revocation status check?

RQ2 How to make the revocation status check tamper-evident?

After knowing the solutions, the third question is to discover how to combine the two good properties into our system;

RQ3 How to combine tamper-evident and privacy-preserving for revocation mechanism?

1.6. Contribution

To the best of our knowledge, this is the first academic work that proposes a privacy-preserving and tamper-evident revocation mechanism for verifiable credentials. Our contributions can be summarized as follows:

- We propose the first privacy-preserving and tamper-evident revocation mechanism for verifiable credentials. Our revocation mechanism can ensure: 1. the revocation and the revocation status check do not leak any identifying information about verifiable credentials and the holder of verifiable credentials; 2. malicious attackers can not pass the revocation status check by replacement fraud.
- We implement a proof of concept of our revocation mechanism in Rust. The efficient bilinear-pairing-based accumulator we use makes our revocation mechanism more storage-saving and faster than the related works by design. Our experimental results show that adding our revocation status check to the verifiable credential system only adds 40ms runtime overhead to presentation and 30ms runtime overhead to verification. We also test the scalability of our proof-of-conception, the transaction per second(TPS) is 70 transaction per second which is far larger than the real world need. Since the absolute figure of the runtime overhead is limited to tens of milliseconds and the system is scalable for read world application, our revocation mechanism is practical in the real world.

1.7. Outline

The rest of the thesis is organized as follows. Chapter 2 introduces the preliminaries needed to understand related works and our system. Chapter 3 gives a brief description of the related works. Chapter 4 describes the requirements of being tamper-evident and privacy-preserving. Chapter 5 presents the algorithm and protocols that constructs the revocation mechanism. Chapter 6 analyze the security and performance of the system. Chapter 7 shows the experiments result of our proof-of-concept implementation. Chapter 8 concludes this work and outlines future work.

2 Preliminaries

This chapter describes the necessary preliminaries to understand our solution and the building blocks of the proposed verifiable credential protocol (see Chapter 5). The building blocks consist of Signature Proof-of-Knowledge (SPK) protocols (see section 2.1), an elliptic-curve-based cryptographic commitment (see section 2.4). a pairing-based signature scheme (see section 2.5) and a pairing-based accumulator (see section 2.6).

Table 2.1 explains the notations we use in this chapter.

Table 2.1: Notations

Notation	Meaning
$\xleftarrow{\$}$	random sampling
$\leftarrow$	assign operation
l	The length of a value. For example, l_c means the length of c
$_R\{0,1\}^{l_c}$	The set of random bitstrings of length l_c
$\mathcal{H}_s$	cryptographic hash function
$\mathbb{Z}_q$	Integer group with order q
$\mathbb{Z}_q^*$	Non-zero integer group with order q

2.1. Zero Knowledge Proof

Zero-knowledge proof is a notion first introduced by Goldwasser et al.[6]. In general, a proof of a theory contains more knowledge than the fact that the theorem is true. For instance, we show a Hamiltonian tour to prove a graph is Hamiltonian. This Hamiltonian tour contains more knowledge than a single bit of Hamiltonian/non-Hamiltonian because the verifier can know about the graph's topology from the given Hamiltonian tour. The graph's topology learned from the Hamilton tour is called additional knowledge conveyed by the proof. In the definition of Goldwasser et al., the zero-knowledge proof is the proof that gives no additional knowledge other than the proof's correctness.

Goldwasser et al.'s definition of zero-knowledge proof is very intuitive; they didn't define the explicit properties of zero-knowledge proof. Goldreich et al.'s work[7] supplied the essential properties for zero-knowledge proof. Goldreich et al. assume that there are two actors in the zero-knowledge proof protocols. One is the actor who gives the proof, called the prover; the other is the actor who verifies the presented proof, called the verifier. Let L be a language, (P, V) are the predetermined program of prover and verifier, and x is a string. For a zero-knowledge proof system to prove $x \in L$, whenever the verifier is following the predetermined program, note as V, the following two conditions always hold:

- **Completeness** If the prover runs its predetermined program P, then, for every constant $c > 0$ and large enough $x \in L$, the verifier accepts the common input x with probability at least $1 - |x|^{-c}$. In other words, the honest prover can convince the verifier of $x \in L$.

- **Soundness** For every program P^*, run by the prover, for every constant $c > 0$ and large enough $x \in L$, the verifier rejects x with probability at least $1 - |x|^{-c}$. In other words, the malicious prover can not fool the verifier.

A protocol that satisfies completeness and soundness is called "proof of knowledge", which means using this protocol, the prover can prove a specific statement is true to the verifier. If the soundness of a protocol relies on a computational assumption, e.g. hardness of strong Diffie-Hellman assumption the protocol is referred to as *arguments of knowledge*. Σ-protocols are protocol constructions that yield efficient arguments of knowledge.

Definition 1. *Σ-protocol A three-move protocol <P,V> is said to be a Σ-protocol for relation R if P sends a message a,V sends a random challenge e and P responds with a message z and the protocol further satisfies*

- *Completeness If P,V follow the protocol on input x and P has a private input w where $(x, w) \in R$, the verifier always accepts*
- *Special Soundness From any x and any pair of accepting conversations on input x, $(a, e, z), (a, e', z')$, where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$*
- *Special Honest-Verifier Zero-Knowledge There exists a polynomial-time simulator M, which on input x and a random e, outputs an accepting conversation of the form (a, e, z), with a probability distribution indistinguishable from conversations between the honest P,V on input x.*

Schnorr identification protocol can be seen as an example of Σ-protocol. The Schnorr identification protocol proves knowledge of a discrete log $y = g^w \pmod p$ in a group of order p. The protocol use a subgroup of order p, and requires p to be prime and q divides $p - 1$. Both the prover and verifier know the group generator g, group order q, subgroup order q and the input y. The prover starts by choosing a random value r from the subgroup $\mathbb{Z}_q^*$, and computes $t \leftarrow g^r \mod p$. Then the prover sends t to the verifier, and the verifier randomly picks a challenge for the prover. Then the prover uses the received c to compute $s \leftarrow r + cw \mod q$ and sends s to the verifier. Finally, the verifier checks if $g^s \equiv ty^c \pmod p$ holds.

Schnorr Protocol

Prover		**Verifier**
g, q, p, y		g, q, p, y
w		
$r \leftarrow\$ \mathbb{Z}_q^*$		
$t \leftarrow g^r \pmod p$		
	$\xrightarrow{t}$	
		$c \leftarrow\$ \{0,1\}^{l_c}$
	$\xleftarrow{c}$	
$s \leftarrow r + cw \pmod q$		
	$\xrightarrow{s}$	
		Verify $g^s \overset{?}{\equiv} ty^c \pmod p$

Protocol 2.1: Schnorr protocol

We called it an interactive protocol because the verifier gives the challenge of Σ-protocol. Using Fiat-Shamir transformation, we can transform the Σ-protocols into a non-interactive signature scheme. The non-interactive signature scheme of Σ-protocol is called Signature Proof-of-Knowledge (SPK). 2.2 is the non-interactive version of the schnorr identification protocol. The only difference between the non-interactive and interactive versions is that the challenge c is generated by the prover using a cryptographic hash function. To simplify the description of a signature proof of knowledge, we use the notation introduced by Camenisch and Stadler. As an example, the signature proof-of-knowledge of the Schnorr identification protocol is noted as $SPK(w) : y \equiv g^w \pmod{q}(m)$, with w the secret value held by prover and m the messages that are signed. Using SPK can reduce the communication cost of the proof and reduce the attack surface for the protocols.

Non-interactive Schnorr Protocol

Prover		**Verifier**
g, q, p, y, m		g, q, p, y, m
w		
$r \leftarrow\$ \mathbb{Z}_q^*$		
$t \leftarrow g^r \pmod{p}$		
$c \leftarrow \mathcal{H}_s(g, q, p, y, t, m)$		
$s \leftarrow r + cw \pmod{q}$		
	$\xrightarrow{s, c}$	
		Verify $g^s \overset{?}{\equiv} ty^c \pmod{p}$

Protocol 2.2: Non-interactive Schnorr protocol

2.2. Bilinear Pairing

Bilinear pairing is a type of mathematical operation that are used to construct cryptographic schemes. For multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ with prime order p, a bilinear pairing is map e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, satisfying the following properties:

- *bilinearity:* $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2 : e(u^a, v^b) = e(u, v)^{ab}$ for $a, b \in \mathbb{Z}$
- *non-degeneracy:* For g_1 being a generator of $\mathbb{G}_1$, g_2 being a generator of $\mathbb{G}_2$ holds that: $e(g_1, g_2) \neq 1$

Galbraith et al.[8] distinguish three types of pairings: if $\mathbb{G}_1 = \mathbb{G}_2$, it is called type-1 pairing; if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficient isomorphism $\phi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$, it is called type-2 pairing; and type-3 pairing, in which $\mathbb{G}_1 \neq \mathbb{G}_2$ and no such isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$.

Type-3 pairings currently allow for the most efficient operations in $\mathbb{G}_1$ given a security level using BN curves with a high embedding. Therefore, it is desirable to describe a cryptographic scheme in a type-3 setting.

2.3. Strong Diffie Hellman Assumption

The computational hardness assumption of our work is the q-Strong Diffie Hellman (q-SDH) assumption. The q-SDH assumption has two versions. The first version by Boneh and Boyen is called the Eurocrypt version, and it is defined in a type-1 and type-2 pairing setting. This version states the following problem is computationally difficult for probabilistic polynomial time adversary:

> Given a q+2-tuple $(g_1, {g_2}^x, {g_2}^{(x^2)}, \dots, {g_2}^{(x^q)}) \in {\mathbb{G}_1}^{(q+1)} \times {\mathbb{G}_2}^2$ with $g_1 = \phi(g_2)$, output a pair $(c, {g_1}^{(1/(x+c))}) \in {\mathbb{Z}_p}^* \times \mathbb{G}_1$.

Boneh and Boyen created a new version of the q-SDH problem to support type-3 pairing settings. This version is called the JOC version. It states the following problem is computationally difficult for probabilistic polynomial time adversary:

> Given a q+2-tuple $(g_1, {g_1}^x, {g_1}^{(x^2)}, \dots, {g_1}^{(x^q), g_2, {g_2}^x}) \in {\mathbb{G}_1}^{(q+1)\times{\mathbb{G}_2}^2}$, output a pair $(c, {g_1}^{(1/(x+c))} \in$ $\mathbb{Z}_p \; \{-X\} \times \mathbb{G}_1$.

Boneh et al. present a protocol for proving possession of a solution to an SDH problem in zero-knowledge. In the protocol, the public values are $g_1, u, v, h \in \mathbb{G}_1$, and $g_2, w \in \mathbb{G}_2$. Here u, v, h are random values in $\mathbb{G}_1$, g_2 is a random generator of $\mathbb{G}_2$, w equals g_2^{γ} for secret value $\gamma \in \mathbb{Z}_p$. The protocol proves possesion of a pair (A,x), where $A \in \mathbb{G}_1$ and $x \in \mathbb{Z}_p$, such that $A^{x+\gamma} = g_1$. Such ad pair satifies $e(A, wg_2^x) = e(g_1, g_2)$. The protocol is a standard generalization of Schnorr's protocol for proving knowledge of discrete logarithm in a group of prime order. Figure 2.3 shows the protocol. In this

A zero-knowledge protocol for SDH

Prover	**Verifier**	
$g_1, u, v, h \in \mathbb{G}_1 \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$	g, q, p, y, m	c
$T_1 \leftarrow u^{\alpha}, T_2 \leftarrow v^{\beta}$		
$T_3 \leftarrow A h^{\alpha+\beta}$		
$\delta_1 \leftarrow x\alpha, \delta_2 \leftarrow x\beta$		
$r_\alpha, r_\beta, r_{\delta_1}, r_{\delta_2} \xleftarrow{\$} \mathbb{Z}_p$		
$R_1 \leftarrow u^{r_\alpha}, R_2 \leftarrow v^{r_\beta}$		
$R_3 \leftarrow e(T_3, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}$		
$R_4 \leftarrow T_1^{r_x} \cdot u^{-r_{\delta_1}}, R_5 \leftarrow T_2^{r_X} \cdot v^{-r_{\delta_2}}$		
	$\xrightarrow{(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)}$	
	$\xleftarrow{c}$	
$s_\alpha \leftarrow r_\alpha + c\alpha, s_\beta \leftarrow r_\beta + c\beta$		
$s_x \leftarrow r_x + cx, s_{\delta_1} \leftarrow r_{\delta_1} + c\delta_1$		
$s_{\delta_2} \leftarrow r_{\delta 2} + c\delta_2$		
	$\xrightarrow{(s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})}$	
		$u^{s_\alpha} \overset{?}{\equiv} T_1^c \cdot R_1$
		$v^{s_\beta} \overset{?}{\equiv} T_2^c \cdot R_2$
		$e(T_3, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha - s_\beta} \cdot e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}}$
		$\overset{?}{\equiv} (e(g_1, g_2)/e(T_3, w))^c \cdot R_3$
		$T_1^{s_x} \cdot u^{-s_{\delta_1}} \overset{?}{\equiv} R_4 \cdot R_1$
		$T_2^{s_x} \cdot v^{-s_{\delta_2}} \overset{?}{\equiv} R_5 \cdot R_1$

Protocol 2.3: ZKP for SDH

protocol, Alice, the prover, selects exponents $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$, and computes a Linear encryption of A. She also computes two helper values δ_1 and δ_2. Alice and Bob then undertake a proof of knowledge of values $(\alpha, \beta, x, \delta_1, \delta_2)$ satisfying the following five relations:

$$
\begin{aligned}
u^{\alpha} &= T_1 \\
v^{\beta} &= T_2 \\
e(T_3, g_2)^x \cdot e(h, w)^{-\alpha-\beta} \cdot e(h, g_2)^{-\delta_1 - \delta_2} &= e(g_1, g_2)/e(T_3, w) \\
T_1^x \cdot u^{-\delta_1} &= 1 \\
T_2^x \cdot v^{-\delta_2} &= 1
\end{aligned}
$$

The proof of knowledge of $(\alpha, \beta, x, \delta_1, \delta_2)$ proceeds as follows. Alices picks random blinding values $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2}$ and computes five values $(R_1, R_2, R_3, R_4, R_5)$ based on the blindings. She then sends $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$ to the verifier. Bob, the verifier, sends a challenge value c to Alice. Alice computes $(s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ and sends them back to Bob. Finally, Bob verifies the values received from Alice. He accepts the proof of knowledge if all five equations in the protocol hold.

2.4. Cryptographic Commitment

The cryptographic commitment scheme is a method to keep the input value hidden and unchangeable before revealing it. Committing means that the user chooses a value to submit, and she can no longer change the submitted value after the commitment. Later, the user can choose to reveal her secret input by sending some auxiliary information to the verifier. A cryptographic commitment provides binding and hiding for the committed value. Binding means that two different committed values are impossible to generate the same commitment value. Hiding means given two commitment value, we can not tell if they are from the same committed value or different committed value. Binding and hiding enable commitment to a suitable tool to seal a secret value.

Pedersen commitment[9] is an elliptic-curve based commitment scheme which supports efficient zero-knowledge proof. The Pedersen scheme works in the following steps:

Setup Let g and h be elements of $\mathbb{G}_q$ such that nobody knows $log_g h$. g, h are the public key of Pedersen commitment.

Commit Let $x \in \mathbb{Z}_q$ be the value to be committed, $r \in \mathbb{Z}_q$ is a randomly chosen value. The commitment is generate by the algorithm 2.1. The committer stores the x, r and send the commitment c to the receiver.

```
$Commit(x, r)$
1 :  $c \leftarrow g^x h^r$
2 :  **return** $c$
```

Algorithm 2.1: Issuer Key Generation

Open To open the commiment c, the committer sends the paired x, r to the receiver. The receiver verifies that $c \overset{?}{\equiv} g^x h^r$

Using Pedersen commitment, the prover can prove that she knows the committed value of a commitment to the verifier without opening the commitment. Protocol 2.4 shows the zero knowledge proof of a committed value. Frist, the prover randomly choose two value $a, b \in \mathbb{Z}_q$, and then use a, b to generate a new commitment c'. After that, using Fiat-shamir transformation to generate the challenge t, and use t to compute a', b'. Then the prover sends a', b', c', t to the verifier. Finally, if $g^{a'} h^{b'} \equiv c' + tc$ holds, the verifier accepts the proof of knowledge.

ZKP for Pedersen

Prover	**Public**	**Verifier**
x, r	g, h, c	
$a \leftarrow \$ \mathbb{Z}_q^*$		
$b \leftarrow \$ \mathbb{Z}_q^*$		
$c' \leftarrow g^a h^b$		
$t \leftarrow \mathcal{H}_s(g, h, c')$		
$a' \leftarrow a + tx$		
$b' \leftarrow b + tr$		
	$\xrightarrow{a', b', c', t}$	
		Verify$g^{a'} h^{b'} \overset{?}{\equiv} c' + tc$

Protocol 2.4: Zero-knowledge proof for Pedersen Commitment

2.5. BBS+ Signature

BBS+ signature is a pairing-based signature that support selective disclosure of signed messages. The name BBS comes from the original authors' family name: Boneh, Boyen, Shacham. Au et al. improved the signature, so it is called BBS+ signature. The basic scheme of the signature is like:

Key Generation Take $(h_0, ..., h_L) \overset{\$}{\leftarrow} \mathbb{G}_1^{L+1}, x \overset{\$}{\leftarrow} \mathbb{Z}_p^*, w \leftarrow g_2^x$, and set $sk = x$ and $pk = (w, h_0, ..., h_L)$.

Sign On input message $(m_1, ..., m_L) \in \mathbb{Z}_p^L$ and secret key x, pick $e, s \overset{\$}{\leftarrow} \mathbb{Z}_p$ and compute $A \leftarrow (g_1 h_0^s \sum_{i=1}^L h_i m_i)^{(\frac{1}{e+x})}$.

Verify On input a public key $(w, h_0, ..., h_L) \in \mathbb{G}_2 \times \mathbb{G}_1^{L+1}$, message $(m_1, ..., m_L) \in \mathbb{Z}_p^L$, and signature $(A, e, s) \in G_1 \times \mathbb{Z}_p^2$, check $e(A, wg_2^e) = e(g_1 h_0^s \sum_{i=1}^L h_i^{m_i}, g_2)$.

Camenisch et al. prsent a proof of knowledge for BBS+ signature. Using their protocol, the prover can prove knowledge of a BBS+ signature while selectively disclosing the signed messages.

A zero-knowledge protocol for BBS+ protocol

Prover		**Verifier**
$g_1, g_2, (h_0, ..., h_L)$		$g_1, g_2, (h_0, ..., h_L)$
(A, e, s)		
$r_1, r_2 \leftarrow\$ \mathbb{Z}_p^*$		
$A' \leftarrow A^{r_1}$		
$r_3 \leftarrow \frac{1}{r_1}$		
$b \leftarrow g_1 h_0^s \prod_{i=1}^{L} h_i^{m_i}$		
$\bar{A} \leftarrow A'^{-e} \cdot b^{r_1}$		
$d \leftarrow (g_1 h_0^s \prod_{i=1}^{L} h_i^{m_i})_1^r \cdot h_0^{-r_2}$		
$s' \leftarrow s - r_2 \cdot r_3$		
$s \leftarrow r + cw \mod q$		
$\pi \leftarrow SPK\{(m_{i i \notin D}, e, r_2, r_3, s') :$		
$\bar{A}/d = A'^{-e} \cdot h_0^{r_2} \wedge g_1 \prod i \in D h_i^{m_i} = d_3^r h_0^{-s'} \prod i \notin D h_i^{-m_i}\}$		
	$\xrightarrow{(A', \bar{A}, d, \pi)}$	
		$e(A', X) \overset{?}{\equiv} e(\bar{A}, g_2)$
		Verify π
		Verify $A' \neq 1_{\mathbb{G}_1}$

Protocol 2.5: ZKP for BBS+ signature

Figure 2.5 shows the zero-knowledge proof of BBS+ signature. The protocol is used to prove that the prover knows the BBS+ signature while selectively disclosing messages $\{m_i\}$ with $i \in D$. The signature is randomized by taking r_1, r_3 and set $A' \leftarrow A^{r_1}$. The key step of the protocol is to prove signature proof of knowledge $\pi = SPK\{(m_{i i \notin D}, e, r_2, r_3, s') : \bar{A}/d = A'^{-e} \cdot h_0^{r_2} \wedge g_1 \prod i \in D h_i^{m_i} = d_3^r h_0^{-s'} \prod i \notin D h_i^{-m_i}\}$. To verify the proof, the verifier checks $A' \neq 1_{\mathbb{G}_1}, e(A', X) = e(\bar{A}, g_2)$ and verifies π. The algorithm of genearating and verifying π are given in 2.2 and 2.3 respectively. The generation of π takes the messages $\{m_i\}$, the indexes of revealed messgaes $RevealedIndexes$, the signature (A, e, s) and the public key of BBS+ signature pk_{BBS} as inputs. First, the prover computes the indexes of unrevealed messages $(j_1, j_2, ..., j_u)$. Then the prover randomly pick blindings $r_1, r_2, e^{\sim}, r_2^{\sim}, r_3^{\sim}, s^{\sim}$ and $\{m_j^{\sim}\}$. Using these blindings, the prover can compute A', D, s', C_1, C_2. Then the prover applies the Fiat-Shamir transformation to generate the challenge value c herself. Finally the prover computes $\hat{e}, \hat{r_2}, \hat{r_3}, \hat{s}$ and send them to the verifier with $\bar{A}, A', D, C_1, C_2$ together. As for the verification of pi, the verifier computes the challenge value c by using the values received from the prover. Then the verifier computes C_1' and C_2', the verifier accepts π if $C_1 = C_1'$ and $C_2 = C_2'$.

```
$BBS.SpkGen(\{m_i\} \in \mathbb{Z}_p^L, RevealedIndexes, A, e, s, pk_{BBS})$
1 :  $(i_1, i_2, ..., i_r) \leftarrow RevealedIndexes$
2 :  $(j_1, j_2, ..., j_u) \leftarrow [L] \backslash RevealedIndexes$
3 :  $(g_1, h_0, h_1, ..h_l) \leftarrow pk_{BBS}$
4 :  $r_1 \leftarrow \$ \mathbb{Z}_p^*$
5 :  $r_2 \leftarrow \$ \mathbb{Z}_p^*$
6 :  $e^{\sim} \leftarrow \$ \mathbb{Z}_p^*$
7 :  $r_2^{\sim} \leftarrow \$ \mathbb{Z}_p^*$
8 :  $r_3^{\sim} \leftarrow \$ \mathbb{Z}_p^*$
9 :  $s^{\sim} \leftarrow \$ \mathbb{Z}_p^*$
10 : for $j_i$ in $(j_1, j_2, ..., j_u)$ do
11 :     $m_j^{\sim} \leftarrow \$ \mathbb{Z}_p^*$
12 : endfor $b \leftarrow g_1 h_0^s \prod_{i=1}^{L} h_i^{m_i}$
13 : $r_3 = r_1^{-1}$
14 : $A' = A^{r_1} \bar{A} = A'^{(-e)} b^{r_1}$
15 : $D = b^{r_1} h_0^{r_2}$
16 : $s' = s + r_2 * r_3$
17 : $C_1 = A' * e^{\sim} + h_0 r_2^{\sim}$
18 : $C_2 = D * (-r_3^{\sim}) h_0^{s^{\sim}} \prod_{j=0}^{u} h_j^{m_j^{\sim}}$
19 : $c = \mathcal{H}_s(pk_{BBS} || \bar{A} || A' || D || C_1 || C_2)$
20 : $\hat{e} \leftarrow e^{\sim} + c * e$
21 : $\hat{r_2} \leftarrow r_2^{\sim} + c * r_2$
22 : $\hat{r_3} \leftarrow r_3^{\sim} + c * r_3$
23 : $\hat{s} \leftarrow s^{\sim} + c * s'$
24 : for $j_i$ in $(j_1, j_2, ..., j_u)$ do
25 :     $\hat{m_j} = m_j^{\sim} + c * m_j$
26 : endfor
27 : $\pi = (\bar{A}, A', D, C_1, C_2, \hat{e}, \hat{r_2}, \hat{r_3}, \hat{s})$
28 : return $\pi$
```

Algorithm 2.2: The generation of π

```
$BBS.SpkVerify(\pi)$
1 : $(\bar{A}, A', D, C_1, C_2\hat{e}, r_2\hat{BBS}, \hat{r_3}, \hat{s}) \leftarrow \pi$
2 : $c = Hash(pk_{BBS}||\bar{A}||A'||D||C_1||C_2)$
3 : $C'_1 = (\hat{A} - D)^c A'^{\hat{e}} h_0^{r_2\hat{BBS}}$
4 : $T = h_0^{s^\sim} \prod_{j=0}^{R} h_i R^{m_i R^\sim}$
5 : $C'_2 = T^c D^{-\hat{r_3}} \prod_{j=0}^{U} h_j U^{\hat{m_j U}}$
6 : return $(C'_1 \overset{?}{=} C_1 \wedge C'_2 \overset{?}{=} C_2)$
```

Algorithm 2.3: The verification of π

BBS+ signature has two significant advantages over the RSA-based CL-signature[10] used by most existing verifiable credential implementations: the key generation and signing are faster, and the size of keys and claims are smaller[11]. Furthermore, combining BBS+ signature and JSON-LD grant good interoperability to verifiable credentials[11]. Therefore, the BBS+ signature is now the most suitable signature for verifiable credentials.

2.6. Cryptographic Accumulator

The cryptographic accumulator scheme is first introduced by Benaloh and de Mare[12]. A cryptographic accumulator allows the user to hash a large set of elements into one short, constant value. Each element in the set has a corresponding value called the witness to prove its (non-)membership. Dynamic cryptographic accumulators refer to accumulators that allow addition and deletion from the given set. Known dynamic accumulators can be categorized as Merkle-tree-based[13], RSA-based[10], [14] and pairing-based[15]–[18]. Each time an element is added (or removed), the accumulator should update witnesses for other elements in a dynamic accumulator. Cryptographic accumulators allow users to prove membership for an element while keeping the value and associated witness hid using cryptographic commitment. Among the proposed accumulators, pairing-based accumulators are the most efficient. They can reach the same level of security as RSA-based accumulators with shorter key-size and faster computation[19].

The pairing-based accumulator proposed by Karantaidou and Baldimtsi[17] is the current state of art pairing-based accumulator. For reference convenience, we called it KB accumulator in this thesis. There are three actors in the KB accumulator: the accumulator manager, the third party, the witness holder. The accumulator manager possesses the secret key sk of the accumulator. The witness holder is responsible for elements accumulated in the accumulator. The third parity is an entity that is interested in verifying membership of the elements. The basic scheme of the KB accumulator contains six parts: the generation algorithm $Gen(1^\lambda, S = \emptyset$, the addition algorithm $Add(sk = a, v_t, x)$, the deletion algorithm $Del(sk = a, v_t, x)$, the witness updation algorithm $MemWitUpOnDel(v_{t+1}, x, w_t^x, upmsg = y)$, the membership verification algorithm for the accumulator manager $VerMemA(sk = a, v_t, x, w_t^x)$ and the membership verification algorithm for the third party $VermMemT(v_t, x, w_t^x)$. These algorithms are performed by different actors . $Gen, Add, DelVerMemA$ are performed by the accumulator manager, $MemWitUpOnDel$ is performed by the witness holder, and $VerMemT$ is performed by the third party.

Figure 2.6 shows the details of the algorithms. $Gen(1^\lambda, S = \emptyset)$ takes the security parameter λ as input and outputs a pairing instance $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $|p| = \lambda$ and p prime, the accumulator's domain $D = \mathbb{Z}_p - \{a\}$, the initial empty accumulator's value v_0 and the manager's secret key sk. The manager use $Add(sk = a, v_t, x)$ to add a new element x into the accumulator. $Add(sk = a, v_t, x)$ takes the secret key, the accumulator value at the time t and the element x as input and outputs x's membership witness at the time t, w_t^x. $Add(sk = a, v_t, x)$ does not change the accumulator value. $Del(sk = a, v_t, y)$ takes similar input as $Add(sk = a, v_t, x)$, however, the deletion changes the accumulator value and sends update message $upmsg = y$ to all witness holders. $MemWitUpOnDel(v_{t+1}, x, w_t^x, upmsg = y)$

Basic Scheme of KB accumulator

```
1 : $Gen(1^\lambda, S = \emptyset)$
    1 : Generate a pairing instance $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $|p| = \lambda$ and $p$ prime
    2 : $a, u_0 \leftarrow\$ \mathbb{Z}_p^*$
    3 : Set the initial accumulator value as $v_0 = g^{u_0} \in \mathbb{G}, where$u$_0$ is a random value instead of the identity element.
    4 : Set the domain $D = \mathbb{Z}_p - \{a\}$
    5 : return $sk = a$, the initial accumulator value $v_0$, public params = $((p, \mathbb{G}, \mathbb{G}_T, e, g), g^a)$
2 : $Add(sk = a, v_t, x)$
    1 : if x $\notin D$, FAIL
    2 : Let $w_{t+1}^x = v^{\frac{1}{x+a}}$ and keep the same accumulator value $v_{t+1} = v_t$
    3 : return w$_{t+1}^x$
3 : $Del(sk = a, v_t, y)$
    1 : if y $\notin D$, FAIL
    2 : Let $v_{t+1} = v_t^{\frac{1}{y+a}}$
    3 : return v$_{t+1}$, $upmsg = y$
4 : $MemWitUpOnDel(v_{t+1}, x, w_t^x, upmsg = y)$
    1 : Compute $w_{t+1}^x = w_t^{x(\frac{1}{y-x})} \times v_{t+1}^{\frac{1}{x-y}}$
    2 : return w$_{t+1}^x$
5 : $VerMemA(sk = a, v_t, x, w_t^x)$
    1 : if v$_t$ $= (w_t^x)^{(x+a)}$, 1
    2 : else return 0
6 : $VerMemT(v_t, x, w_t^x)$
    1 : if e(g,v$_t$) $= e(g^x g^a, w_t^x)$, return 1
    2 : else return 0
```

Protocol 2.6: Basic scheme of KB accumulator describes the process that the witness holders update their membership witnesses according to the received update messages. $VerMemA(sk = a, v_t, x, w_t^x)$ shows the membership verification process of the accumulator manager who knows the secret key sk, whereas $VerMemT$ shows the membership verification process of the third parities.

The writer of KB accumulator also present a zero-knowledge protocol which allows the witness holder to prove his element is accumulated to the third party without revealing the element and associated witness. Let $D = \mathbb{G}_q =< h_i >, i = 0, 1, 2, \mathbb{G} =< g_i >, i = 0, 1$ be publicly known generators to be used for commitments. The protocol wants to prove the following relations in zero-knowledge.

$$ZKP = \{(w, r, \hat{r})\, e(w, g_0^r, g_0^a) = e(v, g_0) \wedge C = g_0^r g_1^{\hat{r}}\}$$

In the above proof, $e(w, g_0^r, g_0^a) = e(v, g_0)$ means the element r is accumulated, whereas $C = g_0^r g_1^{\hat{r}}$ indicates the element r is hid by commitment. The above ZKP still exposes the membership witness value w, to hide w, the above ZKP can be converted into the below form:

$$\begin{aligned} ZKP' = \{(r_1, r_2, r, \delta_1, \delta_2) : w_1 &= g_0^{r_1} g_1^{r_2} \wedge w_1^r = g_0^{\delta_1} g_1^{\delta_2} \\ \wedge \frac{e(w_2, g_0^a)}{e(v_0, g)} &= e(w_2, g_0)^{-r} e(g_1, g_0)_1^{\delta} e(g_1, g_0^a)^{r_1} \\ \wedge C &= g_0^r g_1^{\hat{r}}\} \end{aligned}$$

where $r_1, r_2 \leftarrow\!\$\, \mathbb{Z}_p, w_1 = g_0^{r_1} g_1^{r_2}, w_2 = wg^{r1}, \delta_1 = rr_1, \delta_2 = rr_2$.

ZKP' can been as a variant of the ZKP for SDH (see section 2.3),

2.7. Blockchain

A blockchain is a list of records, called blocks. These blocks are sequentially linked in the order of time. In 2008, Nakatomo Satoshi proposed the bitcoin[20] which signs the advent of blockchain. Although there are many different blockchains now, they share similar block structure with the bitcoin. Figure 2.1 shows the structure of the bitcoin. In the bitcoin, each block contains a cryptographic hash of the previous block, a timestamp, a nonce, and transaction data (generally represented as a Merkle tree, where only the leaf nodes store the transactions). The timestamp proves that the transaction data existed when the block was published to get into its hash. As blocks each contain information about the block previous to it, they form a chain, with each additional block reinforcing the ones before it. Therefore, blockchains are tamper-resistant because once recorded, the data in any given block cannot be altered retroactively without altering all subsequent blocks.

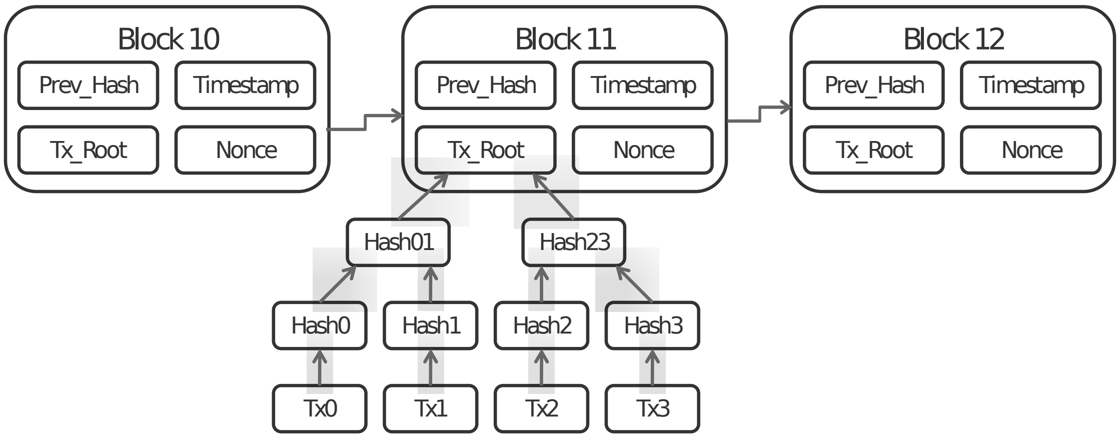

Figure 2.1: Blockchain structure[21]

Blockchains are typically managed by a peer-to-peer network for use as a publicly distributed ledger, where nodes collectively adhere to a protocol to communicate and validate new blocks called consensus.

As blockchains are distributed, they can prevent negative influence caused by centralization (e.g. service lost caused the crashdown of central server).

Figure 2.2 shows that according to different access and validation permissions, blockchains can be classified as public-permissionless, public-permissioned,private-permissionless, and private-permissioned. Here public means that everyone can participant the blockchain, whereas private indicates that only pre-approved users can participate the blockchain. Permissionless means everyone can modify the blockchain, while permissioned means only pre-approved users can modify the blockchain. Therefore, public-permissioned means that everyone can read the blockchain's contents, but only pre-approved participants can validate transactions. Public-permissioned blockchains are ideal tools to apply in authentication[22]. Because in the scenario of authentication, only the issuer needs to modify the credential data, while the credenital holders and verifiers can finish their operations by just querying the blockchain.

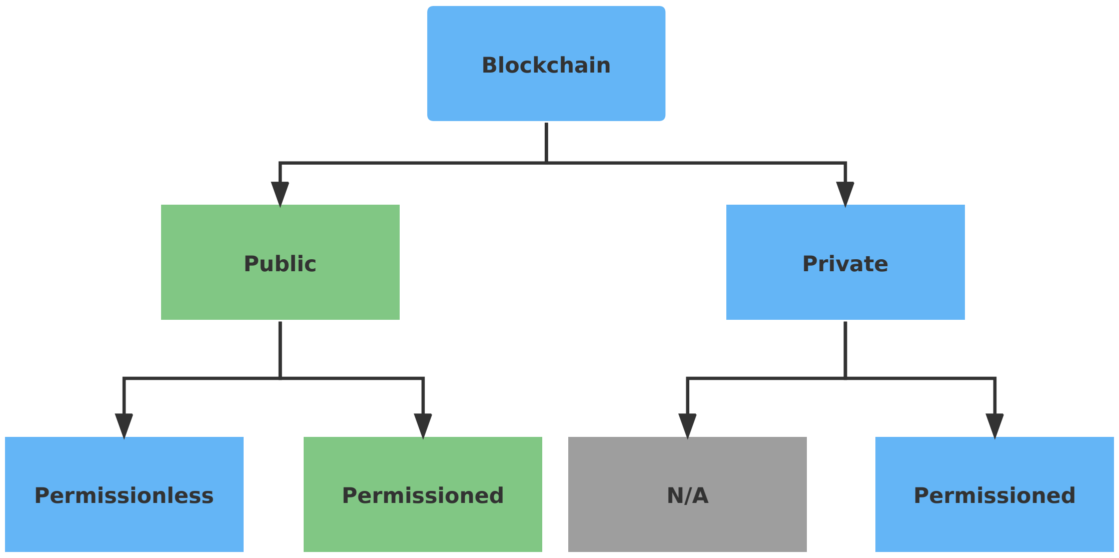

Figure 2.2: Blockchain classification

3 Related Work

In the past few years, several implementations of verifiable credential data model have been proposed, but many of them have not support revocation mechanism yet[23]–[26]. In this paper, we focus on the verifiable credential implementations that already supported revocation mechanisms. We evaluate the existing works in three dimensions: 1. privacy-preserving performance; 2. tamper-evident performance; 3. revocation information storage cost. In this chapter, privacy-preserving means the revocation mechanism should not leak identifying information about the verifiable credential, and tamper-evident means the system should resist the influence of the central server's breakdown.

3.1. OpenAttestation

OpenAttestation (OA)[27] is an open-sourced framework to endorse and verify verifiable credentials based on Ethereum Smart Contracts. OpenAttestation provides two ways to revoke verifiable credentials: using a document store or using the Online Certificate Status Protocol (OCSP).

The document store is a smart contract on the Ethereum network that records the issuance and revocation status of the OpenAttestation documents. The appliance of Ethereum block ensure that OA's revocation mechanism is tamper-evident. The issuer can revoke an OA credential by operating the document store of that credential. The revocation by the document store requires the issuer to know the address of the credential's document store.

As for OCSP, a tool called the OCSP responder is designed to assist the revocation. The OCSP responder records the revocation statuses of the issued verifiable credentials. The verifier can check the revocation status of a verifiable credential by querying the hash of the verifiable credential to the OCSP responder.

The address of the document store used by the document store and the hash of the verifiable credentials are unique identifiers for the verifiable credentials. The verifier can use these identifiers to correlate different presented verifiable credentials. For example, suppose a user disclosed different claims in two presentations. In that case, the attacker can use the identifiers to correlate the different claims to the same credential, violating the verifiable credential data model's selective disclosure.

The document store and OCSP need to record the revocation status for each credential. Thus their storage cost is linear with the number of credentials.

3.2. Veramo

Veramo[28] is the open-source version of uPort project[29]. Veramo aims to enable the user to create and manage verifiable credentials without worrying about interactive operations and vendor lock-in. Veramo is based on the Ethereum network, and it uses a smart contract called ethr-status-registry to carry out the revocation mechanism. Same as OA, the employment of smart contracts ensures the revocation to be tamper-evident. The ethr-status-registry takes the hash value of a credential as input for the revocation status check. Since the hash value of a credential is static, attackers can employ it to correlate verifiable credentials. Therefore the revocation mechanism of veramo can cause leakage of user's personal information, which is to say it is not privacy-preserving. Veramo also needs to maintain revocation status for all credentials it issues. Therefore, the storage cost is linear with the number of credentials.

3.3. Sorvin

Sorvin[22] is a Self-Sovereign Identity (SSI)[30] solution based on HyperLedger Indy[31] public permissioned blockchain. Sorvin supports the user to use of verifiable credentials as the authentication method. The revocation of Sovrin's verifiable credentials is done by using the CKS accumulator[16] and the Indy blockchain. The CKS accumulator indexes the accumulated values and publishes a list that records the current valid indices. Each time a new member joins or an old member leaves, the accumulator should add/delete an index. Thus, the CKS accumulator suffers from join-revoke linkability[18], which means others can determine that the revoked credential is the same credential that was issued before. In particular, the verifier can compare the current index list to stored previous index lists to determine whether the credential revoked just now was issued at a certain previous point. For that reason, the revocation mechanism of Sovrin is not privacy-preserving. Sorvin's revocation mechanism is tamper-evident as it uses CL-signature and the blockchain to guarantee the authenticity of published information. As the result of using the CKS accumulator, the revocation information of Sorvin contains the valid index list and the accumulator value. The storage cost is linear with the valid credentials.

3.4. IRMA

I Reveal My Attributes (IRMA)[32] is a verifiable credential project developed and supported by the Dutch government. IRMA project develops a mobile application to interact with the credentials. All operations about verifiable credentials (issuance, store, presentation) are carried out with the IRMA app, which means the user can use one single app to control her identity. The revocation of IRMA is done based on the CL-RSA-B accumulator[18]. The CL-RSA-B accumulator only requires updating the accumulator value and witnesses for users when revocation happens. It has reached the optimal communication complexity of accumulators[13]. In addition, updating witnesses of CL-RSA-B accumulator does not need the knowledge about current valid credentials, preventing the joint-revoke attack. In addition, the update of witnesses in CL-RSA-B accumulator does not require the knowledge about the indices of valid credentials

IRMA successfully achieved protecting users' privacy. However, the IRMA project uses a central server to process and store the data and once it crashes - which could happen through malicious censorship, a hacking or natural calamity. Although IRMA says they evolve their service stateless by employing Redis as the data storage, a single Redis datastore still suffers the tampers above.

3.5. VCJ

Verifiable-Credential-Java (VCJ)[33] is an implementation of the verifiable credential data model based on Java. VCJ employs Revocation List 2020 as its revocation mechanism. In Revocation List 2020, the issuer keeps a bitstring list of all its issued verifiable credentials. Each verifiable credential is associated with a position in the list, called the revocation list index. If the binary value of the position in the list is 1, the verifiable credential is revoked, if it is 0 it is not revoked. One significant benefit of Revocation List 2020 is the bitstring they use can be compressed to save storage. However, the revocation list index is a kind of identifier that can correlate the verifiable credentials. Therefore the revocation mechanism of VCJ is not privacy-preserving. VCJ employs a blockchain to guarantee the proofs are tamper-evident. The storage cost of VCJ is linear with the number of credentials as it records all credentials in the revocation list.

3.6. Gravity

Gravity[34] is a decentralized cloud platform through which individuals can easily receive, store, and share verifiable credentials in a secure wallet that they fully control. The decentralization of Gravity achieves tamper-evident for revocation, but Gravity suffers from correlation since it employs Revocation List 2020 as its revocation mechanism. The storage cost of Gravity is linear with the number of credentials as a result of using Revocation List 2020.

Table 3.1: Revocation comparison with related works

Project	Methods	Privacy-preserving	Tamper-evident	Storage complexity
OpenAttestation	smart contracts and OCSP	×	✓	O(n)
Veramo	smart contracts	×	✓	O(n)
Sorvin	CKS accumulator	×	✓	O(v)
IRMA	CL-RSA-B accumulator	✓	×	O(1)
Verifiable-Credential-Java	Revocation List 2020	×	✓	O(n)
Gravity	Revocation List 2020	×	✓	O(n)

Table 3.1 lists the summary of the related works. In this table, n means the number of issued credentials, and v implies the number of valid (unrevoked) credentials. From the table, we can see that the existing revocation mechanisms for verifiable credentials fail to achieve privacy-preserving and tamper-evident simultaneously. How to establish a privacy-preserving and tamper-evident revocation mechanism remains an open question.

4 Requirements

This section presents the requirements of our solution. We first introduces the privacy and security considerations of the verifiable credential data model, and verifiable credential's trust model. Based on these considerations and the trust model, we propose the requirements for our revocation mechanism. In this section, we use the following terms: "revocation handler" refers to the value accumulated in the accumulator; "revocation information" represents the auxiliary information required by the revocation mechanism; "proof of unrevocation" indicates the proof generated by the verifiable credential's holder to state the verifiable credential is not revoked.

4.1. Privacy consideration for verifiable credentials

The verifiable credential data model has proposed a spectrum of privacy from pseudonymous to strongly identified. Figure 4.1 shows the privacy spectrum. According to this spectrum, information can be roughly classified to three categories: highly correlatable, correlatable via collision and non-correlatable. Highly correlatable information often refers to global IDs, such as Government ID, shipping address and credit card info. These Global IDs are generally globally unique and are used repeatedly, allowing attackers to pinpoint the user's identity. The information which is correlatable via collusion is also called personally identifiable info. Typically, name, birthday and zipcode are common personally identifiable info. This information is not required to be globally unique as two persons may share the same name or birthday. However, if the attackers group this information, they may be able to recognize the user's identity. An easy example is to group names with birthdays. Although two people sharing the same name is ordinary, two people sharing the same birthday and name is rare. Therefore, grouping names with birthdays can help attackers correlate the user's information. Non-correlatable, also known as pseudonyms, is the most privacy-preserving type of information. Pseudonyms are randomly generated one-time-use information. Since pseudonyms are only used once, they have limited contribution to correlation.

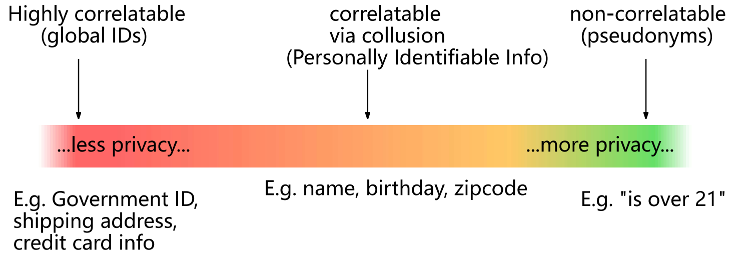
Highly correlatable (global IDs)
correlatable via collusion (Personally Identifiable Info)
non-correlatable (pseudonyms)
...less privacy...
...more privacy...
E.g. Government ID, shipping address, credit card info
E.g. name, birthday, zipcode
E.g. "is over 21"

Figure 4.1: Privacy spectrum ranging from pseduonymous to fully identified[3]

Privacy considerations are use case specific. Depending on the use case, people have different comfort levels about what inforamtion they are willing to provide and what information can be derived from the provided information. For revocation, the verifiable credential data model suggests that the issuers are urged to not use mechanisms, such as credential revocation lists that are unique per credential, during the verification process that could to lead to privacy leakage.

4.2. Security consideration for verifiable credentials

The verifiable credential data model also proposed several security considerations. Among the proposed security consideration, there is no security consideration designed for revocation mechanism specifically. But the security consideration for "bundling dependent claims" and "Device Theft and Impersonation" can be transferred to the scenario of revocation.

"Bundling dependent claims" describes the problem that issuer may group different claims to fool the verifier. In the verifiable credential data model, it is considered best practice for issuers to atomize information in a credential, or use a signature scheme that supports selective disclosure. In this case of atomization, the holder might bundle together diffirent credentials in a way that was not intended by the issuer to fool the verifier. For example, a university might issue two verifiable credentials to a person, each containing two properties, which must be taken together to to designate the "role" of that person in a given "department", such as "Staff Member" in the "Department of Computing", or "Post Graduate Student" in the "Department of Economics". If these verifiable credentials are atomized to put only one of these properties into each credential , then the university would issue four credentials to the person, each containing one of the following designations: "Staff Member", "Post Graduate Student", "Department of Computing", and "Department of Economics". The holder might then transfer the "Staff Member" and "Department of Economics" verifiable credentials to a verifier, which together would comprise a false claim. The bundle dependent claims attack is similar with the replacement fraud we mentioned in section 1.4. As the issuer atomizes revocation status as a single claim, the attacker may bundle a valid revocation status with a revoked credential to pass the validity check.

"Device Theft and Impersonation" indicates that the case that when verifiable credentials are stored on a device and that device is lost or stolen, it might be possible for an attacker to commit malicious behaviours by using the victim's verifiable credentials. As for the case of revocation, "Device Theft" refers to the situation that the central server is hacked and the attacker can modify the revocation information. This may cause the following attacks:

- The first possible attack we call it **revocation information pollution**. In this situation, the attackers add fake information to the published information. For example, the attacker's credential is revoked but they replace the published revocation information with a version which his credential is not revoked. He can still pass the authentication.
- The second possible attack we call it **revocation information deletion**. The attackers will delete the revocation information of the credential system, making the system break down. The deletion may also caused by natural calamity like flood or earthquake.

4.3. Trust model

The trust model identifies the specific requirements that are necessary to run the verifiable credential system correctly. The verifiable credentials trust model is as follows:

- The verifier trusts the issuer to issue the credential that it received. To establish this trust, a credential is expected to either: Include a proof establishing that the issuer generated the credential (that is, it is a verifiable credential), or have been transmitted in a way clearly establishing that the issuer generated the verifiable credential and that the verifiable credential was not tampered with in transit or storage. This trust could be weakened depending on the risk assessment of the verifier.
- All entities trust the verifiable data registry to be tamper-evident and to be a correct record of which data is controlled by which entities.
- The holder and verifier trust the issuer to issue true (that is, not false) credentials about the subject, and to revoke them quickly when appropriate.

- The holder trusts the repository to store credentials securely, to not release them to anyone other than the holder, and to not corrupt or lose them while they are in its care. Here repository refers to a program, such as a storage vault or personal verifiable credential wallet, that stores and protects access to holders' verifiable credentials.

This trust model differentiates itself from other trust models by ensuring the:

- Issuer and the verifier do not need to trust the repository.
- Issuer does not need to know or trust the verifier.

4.4. Our requirements

Our revocation mechanism follows the trust model, the privacy considerations and the security considerations of the verifiable credential data model. Based on principles mentioned above, we give our definition of tamper-evident and privacy-preserving:

Definition 2 (Tamper-evident)**.** *A revocation mechanism is tamper-evident if only the issuer can generate the verifiable credentials' revocation handler and only the issuer can publish the revocation information. Furthermore, being tamper-evident also requires the revocation mechanism can resist certain degree of hacking and disaster.*

Definition 3 (Privacy-preserving)**.** *A revocation mechanism is privacy-preserving if the revocation status check does not leak any identifying information about the verifiable credentials and verifiable credentials' holders.*

To achieve the properties we define, we set the following requirements:
Requirements of tamper-evident:

(1) A verifiable credential's proof of unrevocation should include proof that the issuer generates the revocation handler and the revocation handler is associated with this verifiable credential. This property is called unforgeability.
(2) Only the Issuer can publish the revocation information to the verifiable data registry.

Requirement (1) is to convince the verifier that the revocation status of a given verifiable credential is trustworthy. Requirement (2) ensure that revocation information publishing and updating is correct. Combining requirements (1) and (2), our revocation mechanism achieves tamper-evidence.
Requirements of privacy-preserving:

(3) The proof of unrevocation of a verifiable credential should be unlinkable for the verifier and untraceable for the issuer when the verifiable credential is presented multiple times.
(4) The revocation information should be join-revoke unlinkable.

Requirement (3) gurantees the proof of unrevocation is unlinkable. The holder repeatedly presents the verifiable credential; thus, she needs to generate multiple proofs of unrevocation for the same verifiable credential. Multi-show unlinkable here means the verifier can not use the proof of unrevocation to link verifiable credentials. In other words, given two different proofs of unrevocation, the verifier can not determine if they are generated from the same verifiable credential or two different verifiable credentials. Because there are chances that the verifier collides with the issuer, the revocation mechanism should also avoid the issuer distinguishing the verifiable credential from the proofs of unrevocation, which is to say the proof of unrevocation should be untraceable for the issuer. Additionally, since revocation information may suffer from the join-revoke linkability, we set the requirement (4) to prevent it.

4.5. Security game

This section describes the security games we propose to satisfy the requirements we propose. The security of a cryptographic algorithm can be phrased as a game played between an adversary (or attacker) and a hypothetical entity called the challenger. Both adversary and challenger are probabilistic processes that communicate with each other, and so we can model the game as a probability space. Typically, the definition of security is tied to some particular event S. In this context, security means that for every efficient adversary, the probability that event S occurs is very close to some specified target probability: typically, either 0, $1/2$, or the probability of some event in some other game in which the same adversary is interacting with a different challenger[35]. For the requirements of being privacy-preserving, we propose the security game for multi-show unlinkability and issuer untraceability. For the requirements of being tamper-evident, we first introduces two possible attacks of forgery, then we define the security games to mitigate the attacks. The notations we use in this section is shown in 4.1.

Table 4.1: Notations

Notation	Meaning
$\mathcal{A}$	Probabilistic Polynomial Time (PPT) adversary
VC	verifiable credential
VP	verifiable presentation
PU	proof of unrevocation
b	The correct result of a security game
b'	The guess result of the PPT adversary
st	stored information
$SPGen$	The generation algorithm of system parameters
$spar$	system parameters
λ	security parameter

Multi-show Unlinkability
Multi-show unlinkability means that it is impossible to determine if two given proof of unrevocation are generated from the same VC or two different VC. Here we give a game-based definition of multi-show unlinkability. The security game means that given a security parameter λ, algorithm $SPGen$ use λ to generate the system parameters $spar$. The probabilistic polynomial time attacker $\mathcal{A}$ has already got two verifiable credential VC_0 and VC_1 in his stored information st. $\mathcal{A}$ generates two proof of unrevocation PU_0 and PU_1. Assume a honest user randomly choose a number b from $\{0,1\}$, and use the corresponding verifiable credenital VC_b to generate a new proof of unrevocation PU_b. The attacker $\mathcal{A}$ use the known information $(PU_0, PU_1, PU_b, VC_0, VC_1, st)$ to guess the value of b. The guess result of $\mathcal{A}$ is noted as b'. If $b' = b$, then $\mathcal{A}$ wins the game. The probability of $\mathcal{A}$ wining the game should less than $\frac{1}{2} + negl(\lambda)$, which means if λ is big enough, the honest user always has advantage over the attacker $\mathcal{A}$ for this game. Advantage here means that even though the adversary knows the auxiliary information, the chance of winning this game is same as random guess.

Definition 4. *Revocation token unlinkability: Revocation tokens are multi-show unlinkable, if for every efficient adversary $\mathcal{A}$ there exists a negligible function $negl(\lambda)$ such that the following holds*

$$\begin{aligned}
&Pr[b' = b : \\
&\quad spar \leftarrow SPGen(1^\lambda), \\
&\quad (VC_0, VC_1) \leftarrow \mathcal{A}(st), \\
&\quad PU_0 \leftarrow PUGeneration(VC_0, spar), \\
&\quad PU_1 \leftarrow PUGeneration(VC_1, spar), \\
&\quad b \leftarrow_R \{0,1\}, PU_b \leftarrow PUGeneration(VC_b, spar), \\
&\quad b' \leftarrow \mathcal{A}(PU_0, PU_1, PU_b, VC_0, VC_1, st)] \leq \frac{1}{2} + negl(\lambda)
\end{aligned}$$

Issuer Untraceability
Issuer untraceability means that given a proof of unrevocation and a verifiable credential, it is impossible for the issuer to determine if the proof of unrevocation is generated by the given verifiable credential. We also give the security game about issuer untraceability. The security game also employs $SPGen$ and $PUGeneration$ to compute system parameters $spar$ and proof of unrevocation PU. The adversary $\mathcal{A}$ already knows a credetnial VC_0 and a proof of unrevocation PU_0. The adversary wants to check if PU_0 is generated by VC_0, and he is able use VC_0 as the input of $PUGeneration$ to generate a new proof of

conception PU'_0. b is a bit that represents the fact if PU_0 is generated by VC_0, b' is the bit generated by $\mathcal{A}$ by using all the auxiliary information he gets. Given a big enough security parameter λ, the chance for $\mathcal{A}$ to win the game is equal to random guess.

Definition 5. *Issuer untraceability: Proofs of unrevocations are untraceable for the issuer, if for every efficient adversary $\mathcal{A}$ there exists a negligible function $negl(\lambda)$ such that the following holds.*

$$\begin{aligned}
&Pr[b' = b : \\
&\quad spar \leftarrow SPGen(1^\lambda), \\
&\quad (VC_0, PU_0) \leftarrow \mathcal{A}(st), \\
&\quad PU_0 \leftarrow PUGeneration(VC_0, spar), \\
&\quad b' \leftarrow \mathcal{A}(PU_0, PU_0^p rime, VC_0, spar)] \leq \frac{1}{2} + negl(\lambda)
\end{aligned}$$

4.5.1. Unforgeability

There are two possible way to forge the proof of unrevocation:

- The first possible attack is forge-1 attack, which means the attackers can counterfeit a proof of unrevocation to pass the revocation mechanism for a revoked credential or a fake credential that is not issued by the issuer. For example, if the terrorist can forge a fake passport that can pass the authentication of the boarder, it is a successful forge-1 attack.
- The second possible attack we call it forge-2 attack, which means the attackers use a valid PU from another credential or a previous valid PU pass the revocation status check for a revoked credential. Forge-2 attack is very similar with the replay attack in authentication, the attackers do not change the revocation status of their credentials but they try to use valid proof of unrevocation from other source to fool the verifier.

To ensure the unforgeability, the proof of unrevocation we generate should resist the two forge attacks we mentioned above. Here we give the definition of forge-1 attack resistancy and the definition of forge-2 attack resistancy based on the computational complexity theory.

The negligible function in definiton 6 means that given a big enough security parameter λ , using the verifiable presentation VP^* and the proof of unrevocation PU^* generatecd by a invalid verifiable credential VC^* to pass the verification is impossible.

Definition 6. *Forge-1 attack resistancy: The proof of unrevocation can resist forge-1 attack, if for every efficient adversary $\mathcal{A}$ there exists a negligible function $negl(\lambda)$ such that the following holds.*

$$\begin{aligned}
&Pr[spar \leftarrow SPGen(1^\lambda), \\
&\quad PU^* \leftarrow PUGeneration(VC^*, spar), \\
&\quad VP^* \leftarrow VPGeneration(VC^*, spar), \\
&\quad Verify(VP^*, PU^*, RI) = pass] \leq negl(\lambda)
\end{aligned}$$

The negligible function in definition 7 means that given a big enough security parameter λ , using the verifiable presentation VP^* generatecd by a invalid verifiable credential VC^* and the proof of unrevocation PU_0 generated by a valid verifiable credential VC_0 to pass the verification is impossible.

Definition 7. *Forge-2 attack resistancy: The proof of unrevocation can resist forge-2 attack, if for every efficient adversary $\mathcal{A}$ there exists a negligible function $negl(\lambda)$ such that the following holds.*

$$\begin{aligned}
&Pr[spar \leftarrow SPGen(1^\lambda), \\
&\quad PU_0 \leftarrow PUGeneration(VC_0, spar), \\
&\quad VP^* \leftarrow VPGeneration(VC^*, spar), \\
&\quad Verify(VP^*, PU_0, RI) = pass] \leq negl(\lambda)
\end{aligned}$$

5
System Design

This section introduces the design of our Verifiable Credential system. Our design is based on the BBS+ signature, KB accumulator and a role-based permissioned blockchain. The BBS+ signature ensures the selective disclosure and unforgeability of claims. The KB accumulator enables the prover to prove a committed value is in a specific set. We use a permissioned blockchain to guarantee only the issuer can publish data on the blockchain and the published data are tamper-evident. Integrating the above three building blocks empowers our revocation mechanism to prove a committed revocation handler is signed and accumulated by the issuer in a tamper-evident and privacy-preserving manner.

An overview of the system is showed in figure 5.1. Our system has five components: accumulator, issuer, holder, verifier and permissioned blockchain. In practice, like the original ecosystem of verifiable credentials (figure 1.3), there should be only four actors. The accumulator is part of the issuer. We put it on the system overview because we want to explicitly show that the accumulator is a module that actually performs the function of recording the revocation status of verifiable credentials. The functions of the actors are:

- The issuer can issue verifiable credentials to the holder and revoke issued credentials. Only issuer can operate the accumulator and the permissoned blockchain.
- The holder can use obtained VCs to generate a verifiable presentation and send it to the verifier for authentication. The holder can query the permissoned blockchain to update the witnesses of their credentials.
- The verifier can query the permissioned blockchain to get the newest revocation information and use it to check the validity of the proof of unrevocation given by the holder.
- The accumulator deletes a verifiable credential's revocation handler and updates the accumulator's value when that verifiable credential is revoked.
- The permissioned blockchain performs the same role as the verifiable data registry in figure 1.3. It works as a tamper-evident ledger of the revocation information. Only the issuer can write. Holders and verifiers can merely read the ledger.

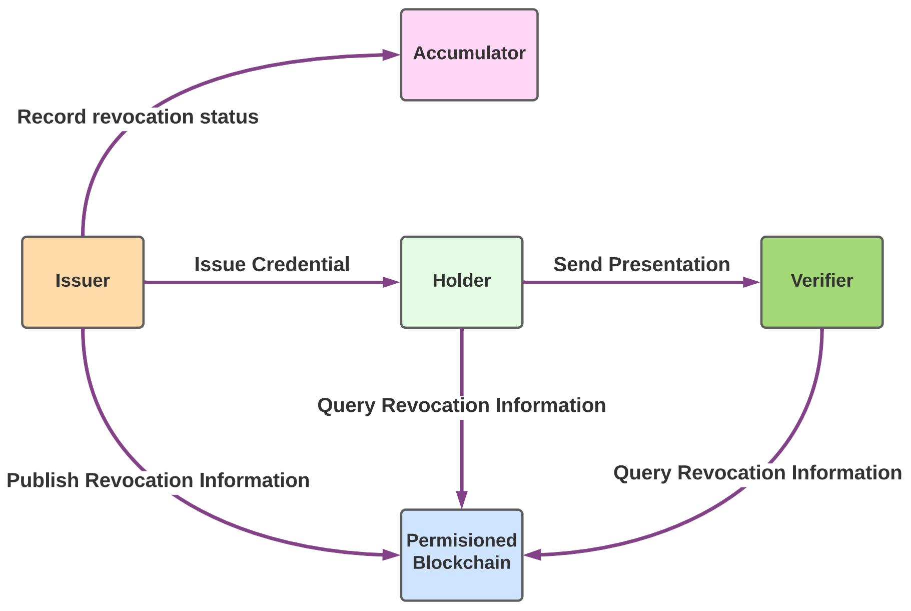

Figure 5.1: The overview of our system

To run our revocation mechanism, the issuer should first set up the KB accumulator and publish the initial accumulator value on the blockchain at the system initialization step. Each time the issuer issues a verifiable credential, the issuer assigns a revocation handler to the verifiable credential. The issuer signs the revocation handler with other claims contained in that verifiable credentials and accumulates it in the KB accumulator. If the issuer wants to revoke an issued verifiable credential, that verifiable credential's revocation handler will be deleted from the KB accumulator, and a new accumulator value will be published. Whenever the holder wants to present a verifiable credential, the holder does not reveal the revocation handler. Instead, the holder sends the committed revocation handler to the verifier. With the help of the revocation information published on the blockchain, the verifier can determine if the committed revocation handler is bound with other given claims and accumulated by the issuer.

The detailed algorithms and protocols for the abovementioned operations are shown in the rest of the chapter. Section 5.1 introduces the algorithm for initializing the system. Section 5.3 presents the issuance protocol. Section 5.2 describes the procedure of revocation. Section 5.4 and section 5.5 explains the algorithms of presentation and verification respectively.

5.1. System initialization

The system initialization is composed of three parts: the generation of BBS+ signature key pairs, the setup of the KB accumulator, and the setup of the Pedersen commitment. Algorithm 5.1 describes the algorithm $SPGen(1^\lambda)$ generates the necessary keys with length of λ bits. $SPGen(1^\lambda)$ consists of three sub-algorithm, namely $BBS.KeyGen(1^\lambda, L)$, $Acc.SetUp(1^\lambda, par_{pairing})$ and $Commitment.SetUp(1^\lambda)$.

The Issuer applies the $BBS.KeyGen(1^\lambda, L)$ algorithm to generate BBS+ signature key pairs according to the credential scheme. A credential scheme contains the description of claims contained in the credential. $BBS.KeyGen(1^\lambda, L)$ takes the security parameter λ and the number of claims in the credential scheme L as input. First, the issuer generate a type-3 pairing instance called $par_{pairing}$. Using the pairing instance, the issuer can compute the public key pk_{BBS} and secret key sk_{BBS}. The pairing instance $par_{pairing}$ will be used as input of the other two algorithms.

The original KB accumulator works in the type-1 setting. We transform it to the type-3 setting to obtain better security property while facilitating the implementation. The $Acc.SetUp(1^\lambda, par_{pairing})$ algorithm takes the security paramete λ and the pairing instance $par_{pairing}$ as input. The following two points are the differences between our $Acc.SetUp$ algorithm and the original generation algorithm of the KB accumulator (see section 2.7):

- The public key changes from $((p, \mathbb{G}, \mathbb{G}_T, e, g), g^a)$ to $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), g_1^a)$
- The initial accumulator value changes from g^{u_0} to $g_2^{u_0}$.

The commitment scheme generation algorithm $Commitment.SetUp(1^\lambda, par_{pairing})$ also takes $par_{pairing}$ as input. Because the commitment scheme should use a group with the same order as the BBS+ signature and the KB accumulator, we use the $\mathbb{G}_1$ in $par_{pairing}$ for implementation convenience.

```
SPGen(1^λ)
1 : Publish the credential scheme $CredScheme$ to the ledger
2 : Generate the key for BBS+ signature
3 : $BBS.KeyGen(1^\lambda, L)$
    1 : Generate a pairing instance $par_{pairirng} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $|p| = \lambda$ and $p$ prime
    2 : $(h_0, ..., h_L) \leftarrow\$ \mathbb{G}_1^{L+1}, x \leftarrow\$ \mathbb{Z}_p^*, w \leftarrow g_2^x,$
    3 : $sk_{BBS} = x, pk_{BBS} = (w, h_0, ..., h_L).$
    4 : return $sk_{BBS}, pk_{BBS}, par_{pairing}$
4 : Set up the accumulator
5 : $Acc.SetUp(1^\lambda, par_{pairing})$
    1 : Pick random $a, u_0 \in \mathbb{Z}_p^*$
    2 : set the initial accumulator value as $v_0 = g_2^{u_0} \in \mathbb{G}_2$, where $u_0$ is a random value instead of the identity element
    3 : Set the domain $D = \mathbb{Z}_p - \{a\}$
    4 : Let $sk_{Acc} = a, pk_{Acc} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), g_1^a)$
    5 : return $sk_{Acc}, pk_{Acc}$
6 : Set up the commitment scheme
7 : $Commitment.SetUp(1^\lambda, par_{pairing})$
    1 : $\mathbb{G}_1 \leftarrow par$
    2 : $g_{com}, h_{com} \in \mathbb{G}_1$
    3 : return $g_{com}, h_{com}$
8 : let $sk_{Issuer} = (sk_{BBS}, sk_{Acc}), pk_{Issuer} = (pk_{BBS}, g_{com}, h_{com}, pk_{Acc})$
9 : return $(sk_{Issuer}, pk_{Issuer})$
```

Algorithm 5.1: Issuer Key Generation

5.2. Revocation

The revocation of a verifiable credential is a two-step process. Firstly, the issuer needs to remove that verifiable credential's revocation handler and update the accumulator value. Secondly, the issuer needs to publish the revoked revocation handler and the newest accumulator value, which are required to update the witnesses for unrevoked verifiable credentials. Note that the holder should always use the latest revocation to update her verifiable credentials' witnesses. Otherwise, she can not generate valid proof of unrevocation. To ensure the user always gets the latest revocation information, we add a version number to the revocation information. Thus, the revocation information published on the blockchain is in this format: (acc, rh, v), here acc means the accumulator value,rh means the revocation handler of the recently revoked verifiable credential, v means the version number of the revocation handler. Algorithm 5.2 shows the process of revocation. The issuer uses the $Acc.Del$ algorithm (see section) to delete the verifiable credential's revocation handler rh and get the newest accumualtor value acc_{new}. After getting the new accumulator value, the issuer updates the verision number and publish the newest revocation information to the blockchain.

To update the witnesses, the holder stores the last revocation information she used. When she needs to generate a revocation handler, she compares the version number of stored revocation information and the version number of the latest revocation information on the data registry. If the two version number is the same, there is no need for the holder to update the witness for the verifiable credential she wants to present. Algorithm 5.3 shows the process of using revocation information to update the witness. $RI_{local}, RI_{latest}, \{RI_{blockchain}\}$ here refer to the local revocation information, the latest revocation information and the set of revocation information stored on the blockchain. The detail of

$Acc.MemWitUpOnDel(witness_{old}, acc, rh)$ can be found in section.

```
$Revoke(rh, RI_{old}, sk_{Acc})$
1 : $acc_{old} \leftarrow RI_{old}$
2 : $acc_{new} \leftarrow Acc.Del(sk_{Acc}, rh, v_{old})$
3 : $v_{new} = v_{old} + 1$
4 : $RI_{new} = (v_{new}, acc_{t+1}, rh)$
5 : Publish $RI_{new}$ to the blockchain
```

Algorithm 5.2: Revoke a verifiable credenital

```
$MemWitUpOnDel(RI_{local}, RI_{latest}, witness_{old}, \{RI_{blockchain}\})$
1 : $v_{local} \leftarrow RI_{local}$
2 : $v_{latest} \leftarrow RI_{latest}$
3 : if $v_{local} == v_{latest}$ then
4 :     $witness_{new} = witness_{old}$
5 : else for $v = v_{local} + 1..v_{latest}$ do
6 :     Find the RI with version $v$ in $\{RI_{blockchain}\}$
7 :     $(acc, rh) \leftarrow RI$
8 :     $witness_{new} = Acc.MemWitUpOnDel(witness_{old}, acc, rh)$
9 : endfor
10 : endif
11 : return $witness_{new}$
```

Algorithm 5.3: Update the witness for a verifiable credential

5.3. Credential Issuance

The credential issuance happens between the user and the issuer. The user sends the set of claims $\{m_i\}$ to the issuer. The issuer generate a revocation handler r and accumulates r in the accumulator. Then the issuer signs the claims $\{m_i\}$ and the revocation handler r together. The issuer puts the signature and witness in the proof part of the credential and embeds the associated witness in the credential. Finally, the issuer sends back the verifiable credential to the user. The holder accepts the verifiable credential if the signature and the witness are valid. Protocol 5.1 shows the credential issuance process. We assume that the communication between the issuer and the user is operated through a secure channel. Thus, we don't consider man-in-the-middle and replay attacks when designing this protocol.

Credential Issuance

Holder		**Issuer**
$\{m_i\}$		sk_{Issuer}
	$\xrightarrow{\{m_i\}}$	
		$r \leftarrow\$ D_{Acc}$
		$witness \leftarrow Acc.add(r, sk_{Issuer})$
		$(A, e, s) \leftarrow BBS.Sign(\{m_i\}, r, sk_{Issuer})$
$VC = (\{m_i\}, (A, e, s), (witness, r))$	$\xleftarrow{VC}$	
Accept if both $BBS.Verify(A, e, s, pk_{Issuer})$ and $Acc.Verify(r, witness, pk_{Issuer})$ are true		

Protocol 5.1: Issuance of supreme verifiable credential

5.4. Credential Presentation

Credential presentation refers to the process of generating a verifiable presentation. When developing the verifiable presentation, the holder selectively discloses the claims required by the verifier, whereas the rest claims and the revocation handler are anonymous. Algorithm 5.4 depicts the procedure of presenting a single verifiable credential. The issuer first updates the witness for the verifiable credential. Then the issuer computes the signature proof of knowledge for the BBS+ signature SPK_{BBS} to prove the issuer signs the revealed claims, and the signature proof of knowledge for the KB accumulator SPK_{Acc} to prove the credential is not revoked. The SPK_{Acc} is the proof of unrevocation we used in Chapter 4. The generation algorithm of SPK_{BBS} (see algorithm 5.5) is similar to the original $BBS.SpkGen$ algorithm described in section 2.5 and the generation algorithm of SPK_{Acc} (see algorithm 5.6) is identical to the proof-of-knowledge of the KB accumulator(see section 2.6). The major difference between our and original algorithms is that our algorithm only computes the intermediate blindings. We don't compute the challenge value separately for the two signature proof of knowledge. Instead, we merge the blindings used in the generation of SPK_{BBS} and the blindings used in the generation of SPK_{Acc} to compute a mutual challenge value c. This is because both the SPK_{BBS} and SPK_{Acc} need a challenge value and blinding factor to compute the commitment value for the revocation handler ($\hat{r}$ is the revocation handler's commitment value in SPK_{Acc}, $\hat{m}_0$ is 's counterpart in SPK_{BBS}). If we use a mutual challenge value c and a mutual blinding factor $\tilde{m_0}$, the revocation handler's commitment value in the SPK_{BBS} is the same as the revocation handler's commitment in the SPK_{Acc}. Thus, the holder can prove the valid revocation status is bundled with the presented verifiable credential. The presentation algorithm is shown in algorithm 5.4. Notice that the $\{m_i\}$ here contains the revocation handler. For convenience, we assume m_0 is the revocation handler, which means $m_0 = r$. Thus, $\tilde{m_0}$ is the blinding factor for the revocation handler, and $\hat{m}_0$ is the revocation handler's commitment value in SPK_{BBS}.

```
$VPGeneration(1^{\lambda}, pk_{Issuer}, r, witness, RevealedIndex, A, e, s, \{m_i\})$
1 : $(pk_{BBS}, pk_{Acc}, g_{com}, h_{com}) \leftarrow pk_{Issuer}$
2 : $MemWitUpOnDel(RI_{local}, RI_{latest}, witness_{old}, \{RI_{blockchain}\}$
3 : $(\bar{A}, A', D, C_1, C_2, blindings_{BBS}) \leftarrow BBS.SpkGen(\{m_i\}, RevealedIndex, A, e, s, pk_{BBS})$
4 : $(R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C_r, blindings_{Acc}) \leftarrow Acc.SpkGen^*(witness, r, g_{com}, h_{com}, pk_{Acc}, m_{\tilde{0}})$
5 : $revealedClaims = \emptyset$
6 : for i in revealedIndex do
7 :     $m_i \cup revealedClaims$
8 : endfor
9 : $c = \mathcal{H}_S(\bar{A}, A', D, C_1, C_2, R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C_r, pk_{Issuer}, revealedClaims)$
10 : $(e^{\sim}, r_{\widetilde{2BBS}}, r_{\tilde{3}}, r_{2BBS}, s^{\sim}, s', \{m_j\}^{\sim}) \leftarrow blinding_{BBS}$
11 : $\hat{e} \leftarrow e^{\sim} + c * e$
12 : $r_{2\hat{BBS}} \leftarrow r_{\widetilde{2BBS}} + c * r_{2BBS}$
13 : $\hat{r_3} \leftarrow r_{\tilde{3}} + c * r_3$
14 : $\hat{s} \leftarrow s^{\sim} + c * s'$
15 : for $j_i$ in $(j_1, j_2, ..., j_u)$ do
16 :     $\hat{m_j} = m_{\tilde{j}} + c * m_j$
17 : endfor
18 : $SPK_{BBS} = (\bar{A}, A', D, C_1, C_2\hat{e}, r_{2\hat{BBS}}, \hat{r_3}, \hat{s}, \{\hat{m_j}\})$
19 : $(r_1, r_2, \delta_1, \delta_2, r_{com}, r_{\tilde{1}}, r_{\tilde{2}}, \delta_{\tilde{1}}, \delta_{\tilde{2}}, r^{\sim}, r_{\widetilde{com}}) \leftarrow blindings_{Acc}$
20 : $\hat{r_1} = r_{\tilde{1}} + c * r_1$
21 : $\hat{r_2} = r_{\tilde{2}} + c * r_2$
22 : $\hat{\delta_1} = \delta_{\tilde{1}} + c * \delta_1$
23 : $\hat{\delta_2} = \delta_{\tilde{2}} + c * \delta_2$
24 : $\hat{r} = r^{\sim} + c * r$
25 : $r_{\hat{com}} = r_{\widetilde{com}} + c * r_{com}$
26 : $SPK_{Acc} = (R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C, \hat{r_1}, \hat{r_2}, \hat{\delta_1}, \hat{\delta_2}, \hat{r}, r_{\hat{com}})$
27 : $VP = (revealedClaims, SPK_{BBS}, SPK_{Acc})$
28 : return $VP$
```

Algorithm 5.4: The generation algorithm of verifiable presentation

```
$BBS.SpkGen(\{m_i\} \in \mathbb{Z}_p^L, RevealedIndex, A, e, s, pk_{Issuer})$
1 :  $(i_1, i_2, ..., i_r) \leftarrow RevealedIndex$
2 :  $(j_1, j_2, ..., j_u) \leftarrow [L] \backslash RevealedIndex$
3 :  $(g_1, h_0, h_1, ..h_l) \leftarrow pk_{Issuer}$
4 :  $r_1 \leftarrow\$ \mathbb{Z}_p^*$
5 :  $r_2 \leftarrow\$ \mathbb{Z}_p^*$
6 :  $e^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
7 :  $r_2^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
8 :  $r_3^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
9 :  $s^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
10 : **for** $j_i$ in $(j_1, j_2, ..., j_u)$ **do**
11 :     $m_j^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
12 : **endfor**
13 : $b \leftarrow g_1 h_0^s \prod_{i=1}^{L} h_i^{m_i}$
14 : $r_3 = r_1^{-1}$
15 : $A' = A^{r_1} \bar{A} = A'^{(-e)} b^{r_1}$
16 : $D = b^{r_1} h_0^{r_2}$
17 : $s' = s + r_2 * r_3$
18 : $C_1 = A' * e^{\sim} + h_0 r_2^{\sim} C_2 = D * (-r_3^{\sim}) h_0^{s^{\sim}} \prod_{j=0}^{u} h_j^{m_j^{\sim}}$
19 : $blindings_{BBS} = (r_2, r_3, r_2^{\sim}, r_3^{\sim}, e^{\sim}, s^{\sim}, s', \{m_j^{\sim}\})$
20 : **return** $(\bar{A}, A', D, C_1, C_2, blindings_{BBS})$
```

Algorithm 5.5: Generate blinding factors for SPK_{BBS}

```
$Acc.SpkGen(witness, r, g_{com}, h_{com}, pk_{Acc}, m_0^{\sim})$
1 : $g_1, g_2, g_a \leftarrow pk_{Acc}$
2 : $r_1, r_2, r_{com} \leftarrow\$ \mathbb{Z}_p^*$
3 : $w_1 \leftarrow g_{com}^{r_1} h_{com}^{r_2}$
4 : $w_2 \leftarrow w h_{com}^{r_1}$
5 : $C_r \leftarrow g_{com}^{r} h_{com}^{r_{com}}$
6 : $\delta_1 \leftarrow r_1 r$
7 : $\delta_2 \leftarrow r_2 r$
8 : $r_1^{\sim}, r_2^{\sim}, \delta_1^{\sim}, \delta_2^{\sim}, r_{com}^{\sim} \leftarrow\$ \mathbb{Z}_p^*$
9 : $r^{\sim} \leftarrow m_0^{\sim}$
10 : $R_1 \leftarrow g_{com}^{r_1^{\sim}} h_{com}^{r_2^{\sim}}$
11 : $R_2 \leftarrow g_{com}^{\delta_1^{\sim}} h_{com}^{\delta_2^{\sim}}$
12 : $R_3 \leftarrow e(w_2, g_2)^{-r^{\sim}} e(g_{com}, g_2)^{\delta_1^{\sim}} e(g, g_a)^{r_1^{\sim}}$
13 : $R_4 \leftarrow g_{com}^{r^{\sim}} h_{com}^{r_{com}^{\sim}}$
14 : $blindings_{Acc} = (r_1, r_2, \delta_1, \delta_2, r_{com}, r_1^{\sim}, r_2^{\sim}, \delta_1^{\sim}, \delta_2^{\sim}, r^{\sim}, r_{com}^{\sim})$
15 : return $(R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C_r, blindings_{Acc})$
```

Algorithm 5.6: Generate blinding factors for SPK_{Acc}

5.5. Credential Verification

Credential verification depicts the process by which the verifier checks the given verifiable presentation. The verifier verifies the given SPK_{BBS} and SPK_{Acc}, and checks if the revocation handler's commitment is the same in the two signature proof of knowledge. If the verifiable presentation passes the three validity checks mentioned above, the verifier accepts it. Algorithm 5.7 shows the steps of verification.

```
$Verification(VP, RI, pk_{Issuer}))$
1 :  $(SPK_{BBS}, SPK_{Acc}, revealedClaims) \leftarrow VP$
2 :  $\bar{A}, A', D, C_1, C_2 \leftarrow SPK_{BBS}$
3 :  $R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C_r \leftarrow SPK_{Acc}$
4 :  $c = \mathcal{H}_s(\bar{A}, A', D, C_1, C_2, R_1, R_2, R_3, R_4, w_1, w_2, w_1^r, C_r, pk_{Issuer}, revealedClaims)$
5 :  Verify the $SPK_{BBS}$
6 :  $BBS.SpkVerify(c, SPK_{VP})$
        1 :  $C_1' = (\hat{A} - D)^c A'^{\hat{e}} h_0^{r2\hat{B}BS}$
        2 :  $T = h_0^{s\sim} \prod_{j=0}^{R} h_i R^{m_i R^{\sim}}$
        3 :  $C_2' = T^c D^{-\hat{r_3}} \prod_{j=0}^{U} h_j U^{m_j \hat{U}}$
        4 :  **return** $(C_1' \overset{?}{=} C_1 \wedge C_2' \overset{?}{=} C_2)$
7 :  Verify the $SPK_{Acc}$
8 :  $Acc.SpkVerify(c, SPK_{Acc})$
        1 :  $R_1' = g_{com}^{\hat{r1}} h_{com}^{\hat{r2}}$
        2 :  $R_2' = g_{com}^{\hat{\delta_1}} h_{com}^{\hat{\delta_2}}$
        3 :  $R_3' = e(w_2, g_a) e(v_t, g_2)^{-1} R_3$
        4 :  $\hat{R_3} = e(w_2, g_2)^{-\hat{r}} e(g_{com}, g_2)^{\hat{\delta_2}} e(g_{com}, g_a)^{\hat{r_1}}$
        5 :  $R_4' = g_{com}^{\hat{r}} h_{com}^{r_c \hat{o} m}$
        6 :  **return** $(R_1' \overset{?}{=} R_1 + c * w_1 \wedge R_2' \overset{?}{=} R_2 + c * w_1^r \wedge R_3' \overset{?}{=} \hat{R_3} \wedge R_4' \overset{?}{=} R_4 + c * C_r)$
9 :  $\hat{r} \overset{?}{=} \hat{m}_0$
10 : **return** $True$
```

Algorithm 5.7: The verification of a verifiable presentation

6 Analysis

This section analyze our revocation mechanism in three dimensions: security and privacy, performance, and storage. For security, we analyse the unforgeablility of our revocation mechanism; for privacy, we analyse the multi-show unlinkability and issuer untraceability our solution. As for performance, we analyse the impact of adding our revocation mechanism to the performance of a verifiable credential system.

6.1. Security analysis

This section illustrate how can our design satisfy the requirements of being tamper-evident we set in Chapter 4. The first requirement is to ensure the revocation status is unforegable. We give the following proof to prove our revocation mechanism provides unforgeabilitiy to the revocation status.

Theorem 6.1.1 (Unforgeability)**.** *Our revocation mechanism provides unforgeability to the revocation status; a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to derive a proof of unrevocation for revoked credentials or use other credentials' proof of unrevocation to pass the revocation status check.*

Proof. To derive a proof of unrevocation for a revoked credential, the adversary $\mathcal{A}$ needs to prove a revoked revocation handler is accumulated in the accumulator. This is proved to be impossible under the Strong Diffie-Hellman assumption in [18]. To pass the authentication with other valid credential's proof of unrevocation, the adversary $\mathcal{A}$ needs to prove the commitment of other credentials' revocation handler C_r' is signed with the claims contained in this verifiable credential. The unforgeability of the BBS+ signature ensures this is impossible under the Strong Diffie-Hellman assumption. For complete proof, please read [36]. □

Aside from ensuring the unforgeability of revocation status. Our system should also guarantee the unforgeability of the revocation information. We achieve the unforgeability of revocation information by using a role-based blockchain to limit the write permission to the issuer. The tamper-resistance of blockchain makes sure the attackers can not forge the revocation information.

6.2. Privacy analysis

This section analyse the privacy property of our solution. Our revocation mechanism can provide multi-show unlinkability, issuer untraceability and join-revoke unlinkability. Below are the proofs:

Theorem 6.2.1 (Multi-show Unlinkability)**.** *Our revocation mechanism provides multi-show unlinkablity; a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to determine if two given proof of unrevocation are generated from the same verifiable credential or two different verifiable credentials.*

Proof. The multi-show unlinkability is achieved by employing the BBS+ signature's signature proof of knowledge SPK_{BBS} and the KB accumulator's signature proof of knowledge SPK_{Acc}. SPK_{BBS} and SPK_{Acc} use commitment to hide the identifying information (e.g. the revocation handler). The perfect hiding property of Pedersen commitment ensures that the attacker can not derive the revocation handler or other hidden identifiers from the commitment under the Strong Diffie-Hellman assumption. Thus, the attacker can not use the proof of unrevocation to link a verifiable credential even though that verifiable credential is presented multiple times. □

Theorem 6.2.2 (Issuer Untraceability)**.** *Our revocation mechanism provides issuer untraceability; given a verifiable credential and a proof of unrevocation, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to determine if the given verifiable credential generates the proof of unrevocation.*

Proof. Similar to the proof of multi-show unlinkability, the perfect hiding property of Pedersen commitment guarantees no identifying information leakage under the Strong Diffie-Hellman assumption. Thus, the issuer cannot use proof of unrevocation to derive the original verifiable credential from a verifiable presentation. That is to say the issuer cannot track the use of a verifiable credential through its verifiable presentations. □

Theorem 6.2.3 (Join-revoke Unlinkability)**.** *Our revocation mechanism provides join-revoke unlinkability; given a verifiable credential and a proof of unrevocation, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to determine when did the given credential is issued.*

Proof. The KB accumulator achieves the join-revoke unlinkability for our revocation mechanism. Using the KB accumulator, the holders do not have the knowledge about the current valid members when updating the witnesses for their verifiable credentials. Therefore, the attacker can not predict the join time of a certain verifiable credential and cannot commit the join-revoke attack. For the detailed description, we refer the readers to read the appendix of [18]. □

6.3. Performance analysis

Here we give the performance analysis of our solution. Figure 6.1 shows the lifecycle of a verifiable credential. The figure shows that repeatable operations are the presentation of a verifiable credential, the verification of presented verifiable credentials, and the revocation status check. In the credential system with revocation system, revocation status check and the verification of presented verifiable credentials can be merged. Thus, the system's performance with the revocation mechanism is dominated by two operations: verifiable credential presentation and verifiable credential verification. Our analysis mainly focuses on the impact on the two operations mentioned above.

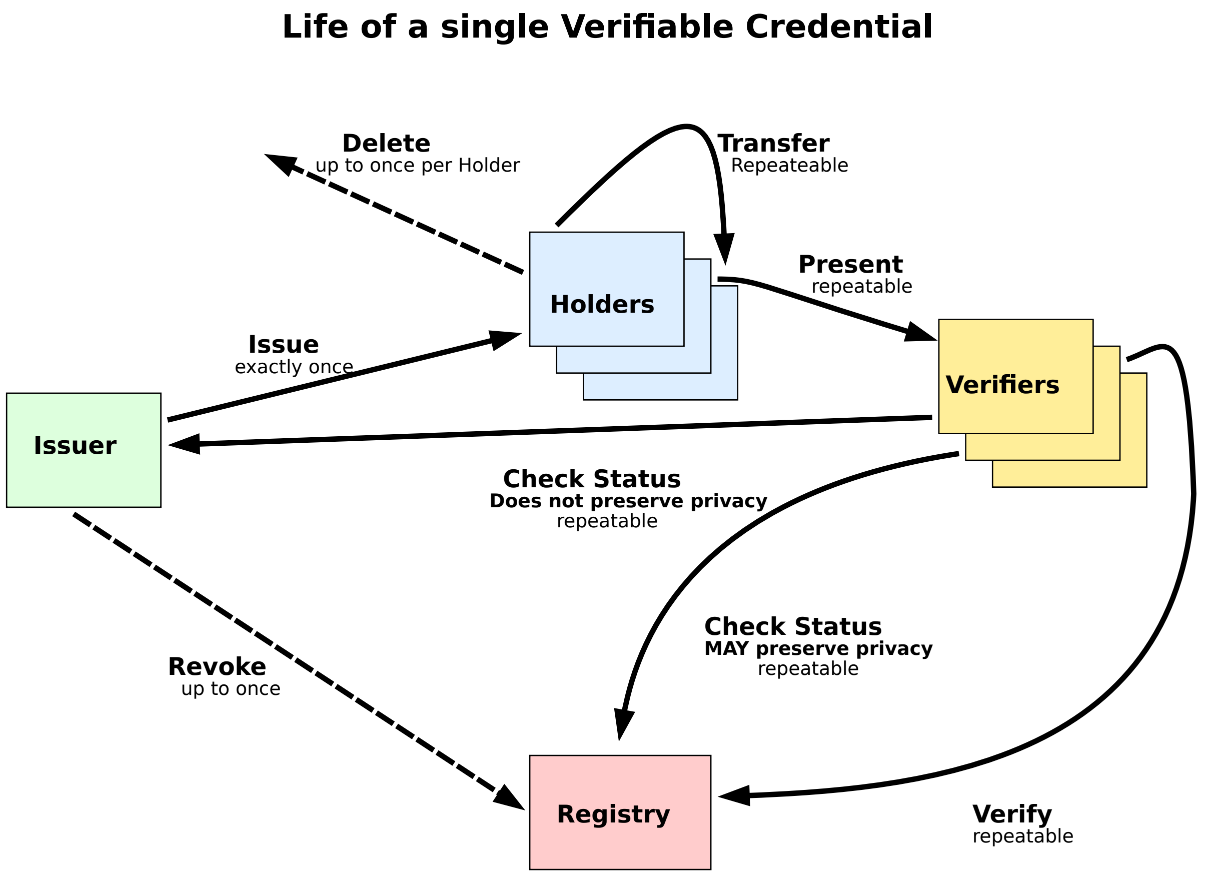

Figure 6.1: The lifecycle of a verifiable credentials and the verification of verifiable presentations. [3]

6.3.1. Computation complexity

The verifiable credential presentation consists of generating the SPK_{BBS} and generating the SPK_{ACC}. Compared to the system without a revocation mechanism, our design adds the generation of SPK_{ACC}. Thus, the runtime overhead equals to the generation time of the SPK_{ACC}. [17] shows that generating the SPK_{ACC} takes a fixed number of parameters which means the computation complexity is linear with the size of the parameters. In our design, the size of the parameters is fixed. Therefore, the runtime overhead caused by adding our revocation mechanism is under constant complexity $O(1)$.

The verifiable credential verification is to verify the SPK_{ACC}. Similar to the analysis of the presentation, we can conclude that the computation complexity of the runtime overhead for the status check is also $O(1)$. Besides, since the computation complexity of verifying SPK_{BBS} is constant if we use parameters of fixed size, the overall computation complexity of verifiable credential presentation is also $O(1)$

6.3.2. Storage

The revocation information of our revocation mechanism contains three parts: the accumulator value, the revoked revocation handler and the version number. Thus the storage cost is a constant value $l_{acc} + l_{rh} + l_v$. Here l_{acc}, l_{rh}, l_v represents the length of the accumulator value, the length of the revoked revocation handler and the length of the version number. The storage complexity is $O(1)$.

6.4. Discussion

The existing works use revocation mechanisms such as smart contracts, OCSP, and revocation list 2020 to manipulate identifiers to check the revocation status, which violates the verifiable credential's requirement of being privacy-preserving. Sorvin uses the CKS accumulator as the revocation mechanism, which avoids the use of identifiers, but adversaries can still correlate the credentials through a join-revoke attack. The appliance of CL-RSA-B makes IRMA outperforms other existing works on the enhancement of privacy; however, the use of a central server makes it lose in the performance of tamper-evident. To achieve tamper-evident and privacy-preserving together, we propose a new revocation mechanism. Table 6.1 compares our revocation mechanism with the existing works. From the table we can see that, our solution is only revocation mechanism that achieve tamper-evident and privacy-preserving simultaneously. Besides our revocation mechanism also reaches the state of art storage complexity O(1).

Table 6.1: Revocation comparison with related works

Project	Methods	Privacy-preserving	Tamper-evident	Storage complexity
OpenAttestation	smart contracts and OCSP	×	✓	O(n)
Veramo	smart contracts	×	✓	O(n)
Sorvin	CKS accumulator	×	✓	O(v)
IRMA	CL-RSA-B accumulator	✓	×	O(1)
Verifiable-Credential-Java	Revocation List 2020	×	✓	O(n)
Gravity	Revocation List 2020	×	✓	O(n)
Our work(Chapter 5)	Efficient bilinear pairing-based accumulator	✓	✓	O(1)

7
Experiments result

Here we analyse the impact of adding our revocation mechanism to the performance of a verifiable credential system. Figure 6.1 shows the lifecycle of a verifiable credential. The figure shows that repeatable operations are the presentation of a verifiable credential, the verification of presented verifiable credentials, and the revocation status check. In the credential system with revocation system, revocation status check and the verification of presented verifiable credentials can be merged. Thus, the system's performance with the revocation mechanism is dominated by two operations: verifiable credential presentation and verifiable credential verification. Our analysis mainly focuses on the impact on the two operations mentioned above.

The verifiable credential presentation consists of generating the SPK_{BBS} and generating the SPK_{ACC}. Compared to the system without a revocation mechanism, our design adds the generation of SPK_{ACC}. Thus, the runtime overhead equals to the generation time of the SPK_{ACC}. [17] shows that generating the SPK_{ACC} takes a fixed number of parameters which means the computation complexity is linear with the size of the parameters. In our design, the size of the parameters is fixed. Therefore, the runtime overhead caused by adding our revocation mechanism is under constant complexity $O(1)$.

The verifiable credential verification is to verify the SPK_{ACC}. Similar to the analysis of the presentation, we can conclude that the computation complexity of the runtime overhead for the status check is also $O(1)$. Besides, since the computation complexity of verifying SPK_{BBS} is constant if we use parameters of fixed size, the overall computation complexity of verifiable credential presentation is also $O(1)$

7.1. Experimental results

7.1.1. Runtime of off-chain zero-knowledge proof

This section measures the experimental runtime of our revocation mechanism. The runtime is measured by comparing the credential system with our revocation mechanism (noted as "w" in the figures) against the credential system without a revocation mechanism (noted as"w/o" in the figures). The proof-of-concept implementation used in the experiments is written in Rust 1.59.0. The pairing-friendly curve we choose is Bls12-381, the random generator we use is the thread_rng(). The runtime measurements were performed on a Macbook Air laptop with an Apple M1 chip and 8GB memory under macOS Monterey 12.2.1.

Same as the performance analysis in section 5, we compare the runtime of the two credential systems for the following algorithms: verifiable credential presentation and verifiable credential verification. We use credentials with 5, 10, 15, and 20 attributes to evaluate the runtime.

Figure 7.1 shows the runtime measurements of verifiable credential presentation. For the system without a revocation mechanism, the runtime is the generation of SPK_{BBS}. For the system with our revocation mechanism, the runtime equals the sum of generating SPK_{BBS} and SPK_{ACC}. Our revocation mechanism's average runtime overhead brought on verifiable credential presentation is 42.86ms. And according to our analysis in section 6.3.1, the runtime overhead has a constant computation complexity. Thus, the runtime overhead will not fluctuate significantly with the change of number claims in a credential. Because the generation of SPK_{BBS} is linear with the number of claims in the credentials[36], with the number of claims in the credential increases, the generation of SPK_{BBS} will gradually dominate the runtime of verifiable credential presentation. In other words, the impact of our revocation mechanism on verifiable credential presentation will decrease with the increase of the claims contained in a verifiable credential. We believe that as the application scenarios of verifiable credentials consistently extend, there will be a need to contain more complex claims in the verifiable credential. Thus, having an increasing number of claims in the variable credential is realistic. For this reason, the runtime overhead on verifiable credential presentation caused by our design is acceptable in practice.

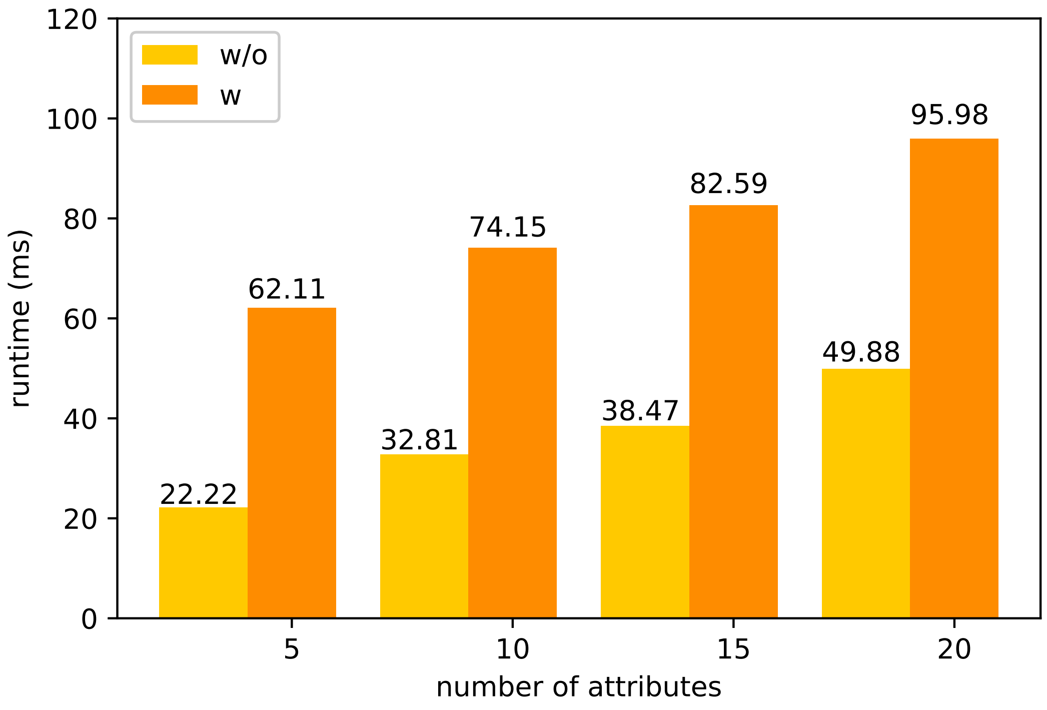

Figure 7.1: Runtime measurements of verifiable credential presentation

Figure 7.2 shows the runtime measurements of verifiable credential verification. The runtime for the system without a revocation mechanism is the verification of SPK_{BBS}, and the runtime for the system with our revocation mechanism is the overall measurement of verifying SPK_{BBS} and SPK_{ACC}. Our revocation mechanism brings 31.36ms average runtime overhead to the system, and the average overall verification runtime is 45.84ms. As we analysed in section 6.3, the computation complexity of verifiable credential verification is $O(1)$. Therefore, the runtime of verifiable credentials verifiable should slightly fluctuate around the average overall runtime. Since the runtime of verifiable credential verification is limited to tens of milliseconds and the time will not change significantly with increasing claims, we believe the 31.36ms runtime overhead is not a practical problem.

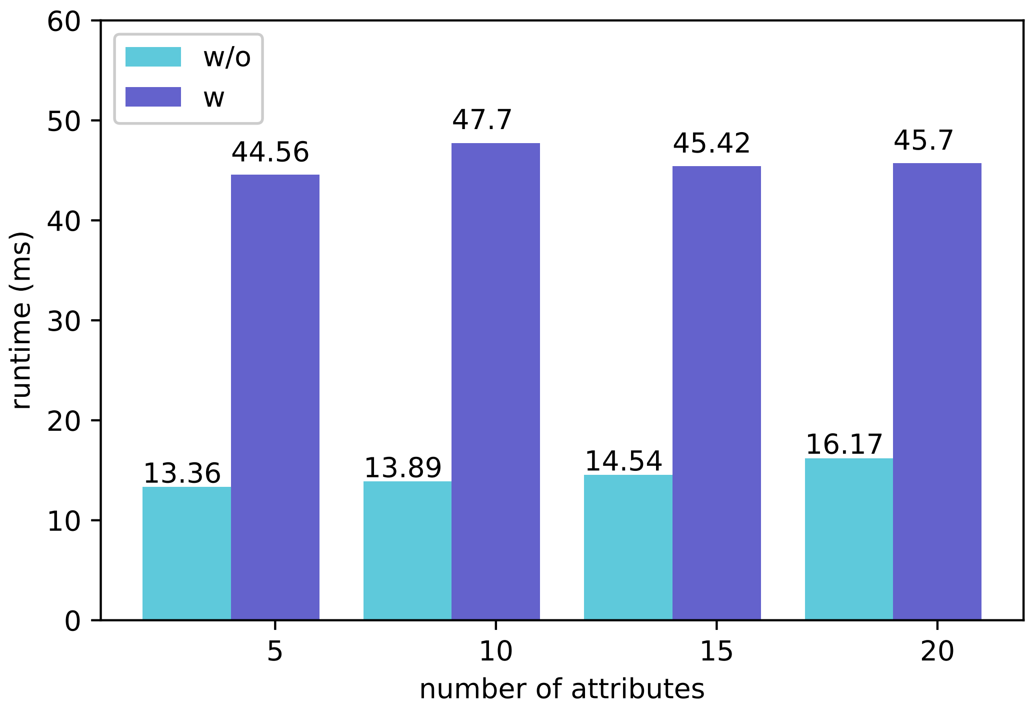

Figure 7.2: Runtime measurements of verifiable credential verification

7.1.2. On-chain scalability

We also combine our proof of concept implementation with hyperledger Iroha[37] to test the scalability of our system. We test the transaction per second (TPS) of our mechanism using the test tools provided by Iroha, and the test is run on the same machine we employ in the runtime measurements of off-chain computation. The transaction per second (TPS) result of our revocation mechanism with four nodes is 70 transaction/second (6048000 transaction/day), which means there can be 6048000 revocations per day in our system. We believe the TPS is big enough for practical use because revocation is a rare operation in the real world. For example, from 2015 to 2016, even in the American state with the highest number of revoked driver's licenses, the annual revocation rate was only 7.57%, and the actual number was only 41263[38]. In other words, there are only, on average, 113 revoke credentials per day. The throughput of our revocation mechanism is far more extensive than 113 transaction/day. Therefore, our revocation mechanism is scalable enough in the real world.

8 Conclusion and future work

This work presents a revocation mechanism based on the BBS+ signature, the KB accumulator and a permissioned blockchain. To the best of our knowledge, this is the first privacy-preserving and tamper-evident revocation mechanism for verifiable credentials. Our design shows that it is possible to check the revocation status of a verifiable credntial while keeping the revocation handler anonymous and tamper-evident. In this conclusion, we revisit each question and give an answer based on the result of this work. Before returning to the questions, we first discuss the results and limitations of this work chapter by chapter.

8.1. Discussion

This thesis started with the question of how a verifiable credential's revocation mechanism can be privacy-preserving and tamper-evident. In chapter 3, we describe why existing revocation mechanisms cannot provide tamper-resistance and privacy. Two factors cause the previous works to fail to provide privacy: the use of identifiers and the join-revoke attack. For the evaluation of tamper-resistance, we assume centralized solutions are not tamper-evident as they may suffer from malicious hacking or natural calamity. In this setting, the project IRMA does not provide an ideal revocation mechanism, although it successfully prevents join-revoke attacks and the usage of identifiers (see Section 3.4).

To better define tamper-evident and privacy-preserving for a revocation mechanism, we propose four requirements based on the verifiable data model's trust model, security considerations and privacy considerations in Chapter 4. We also give the security game for the requirements to address how to satisfy these requirements with cryptography.

We present the design of our revocation mechanism in chapter 5. Our work's most influential design decision is to combine the BBS+ signature and the KB accumulator. This decision endows our system with the following properties:

- The holder of a verifiable credential can prove the claims she presented is not revoked to the verifier without disclosing the revocation handler. This way, our revocation mechanism prevents privacy leakage through a revocation status check.
- The attackers cannot pass the revocation status checks if they use a revoked or forged credential.

Notice that our design in Chapter 5 is a prototype revocation mechanism. It can not be put into use in the practice directly. To employment in practice, more work needs to be done in integrating other standards proposed in the official verifiable data model (e.g. the standard of data preprocessing and data encoding). Given this thesis's limited time and scope, we cannot consider all these standards. For simplicity, We assume the data we used in the protocol are well-processed, and the other procedures are operated correctly.

In Chapter 6, we prove a verifiable credential's revocation status is unforgeable, and the revocation status check is privacy-preserving under the Strong Diffie-Hellman assumption. We cannot give concrete proof for the unforgeability of the revocation information. Because we assume the revocation information is unforgeable under the belief that the blockchain we use is tamper-evident. As the tamper resistance of the blockchain differs from the consensus protocols and node settings, the tamper resistance of our revocation mechanism is case-specific.

We show the experimental results of our proof-of-concept implementation in Chapter 7. Our work's performance can not be compared directly with the related works as the testing environment, and security assumptions differ. Besides, our proof-of-concept implementation lacks performance optimizations, preventing us from reliably computing the additional overhead.

8.2. The answer to research questions

After discussing the results and limitations of this work, we return to the research questions.

RQ1 How to achieve privacy-preserving in revocation status check?

To achieve privacy-preserving in revocation status check, we should avoid two types of privacy leakage: 1) the privacy leakage caused by the revocation handler and 2) the privacy leakage caused by the revocation information. The first type of privacy leakage is a consequence of using an identifier as the revocation handler; the second type refers to join-revoke attack specifically. We employ the KB accumulator to ensure the revocation status check is privacy-preserving. The KB accumulator empowers us to prove a credential with an anonymous revocation handler is unrevoked and prevents the join-revoke attack. The proof is given in Chapter 6.

RQ2 How to make the revocation status check tamper-evident?

This question equals to the question "how to avoid replacement fraud". There are two types of replacement fraud. The first type is to replace the revoked credentials' proof of unrevocation with a valid proof of unrevocation. This fraud can be solved by guaranteeing the credential's integrity with a cryptographic signature. The second type of replacement fraud is to replace the revocation information. Our revocation mechanism prevents this by publishing the revocation information on the blockchain—the tamper resistance of the blockchain guarantees that revocation information can not be replaced.

RQ3 How to combine tamper-evident and privacy-preserving for revocation mechanism?

As we mentioned in the introduction, the anonymity of the revocation handler might give the attacker chance to commit replacement fraud. Although we use a cryptographic signature to ensure the integrity of the credential, the signature is a fixed value, which means it might be used as an identifier to correlate the credential. Thus, we should avoid showing the signature directly. Combining tamper-evident and privacy-preserving means we should find a way to prove the anonymous revocation handler is signed and accumulated by the issuer. We achieve this goal by integrating the BBS+ signature and the KB accumulator. Our design has two signature proof of knowledge SPK_{BBS} and SPK_{Acc}. Both SPK_{BBS} and SPK_{Acc} contain the revocation handler's commitment value. When receiving a verifiable presentation, the verifier checks if the revocation handler's commitment in the two signature proof of knowledge is the same. Thus, the attacker can not fool the verifier with other credential's SPK_{Acc} as the revocation handler's commitment in that SPK_{Acc} is different from the revocation handler's commitment contained in the revoked credential's SPK_{BBS}. And the perfect hiding of commitment ensures the revocation handler remains anonymous to avoid privacy leakage.

How to design a privacy-preserving tamper-evident revocation mechanism for verifiable credentials?

Considering the answers we gave to the subquestions, we now return to the main research question. In this work, we researched how a revocation mechanism for verifiable credentials can be privacy-preserving and tamper-evident. We analysed how existing works failed to achieve these two properties and proposed four requirements that a revocation mechanism must fulfil to become tamper-evident and privacy-preserving. Integrating the BBS+ signature, the KB accumulator and the blockchain, we construct a tamper-evident and privacy-preserving revocation mechanism for the verifiable credential. Our revocation mechanism allows the verifier to verify the validity of the presented claims without compromising the privacy of the credentials holders.

8.3. Future work

As discussed in conclusion, due to this thesis's limited time and scope, this work has several limitations. Further research can be done based on our work. Here we suggest two directions for future research:

- **Allow holder to revoke the verifiable credential** The right of revocation of the verifiable credentials belongs to the holder for now. However, in some cases, the issuer also needs the right of revocation. For example, if the holder finds the device storing verifiable credentials is lost or stolen. She should be able to revoke the verifiable credential instantly instead of informing the issuer and waiting for the issuer to react. Currently, our revocation mechanism does not support the revocation for the holder. How to allow the holder to revoke the verifiable credential is open to research.
- **Selective revocation.** Our revocation mechanism currently only considers the case of revoking the credential, which means all the credential claims will become invalid after revocation. However, sometimes, we only need to revoke part of the claims. Typically, selective revocation is important to flexible access control. In the case of access control, the issuer can issue a verifiable credential that specifies the permissions of the credential holder. For example, Alice owns a verifiable credential that allows her to read, write and delete a specific database. If the issuer wants to prohibit Alice's delete permission while maintaining Alice's read and write permissions, the issuer can selectively revoke the claim that allows Alice to delete the database. Our revocation mechanism can achieve selective disclosure if we assign revocation handlers for all revocable claims. However, assigning revocation handlers for all revocable claims is inefficient because the increase of the revocable claims might lead the blockchain to become a heavy storage burden to the system. Designing a revocation mechanism that supports efficient selective revocation remains an open question.

Bibliography

[1] *Student faq*. [Online]. Available: https://verify.sheerid.com/student-faq-b/#who-is-sheerid.

[2] *How it works - sheerid developer center*. [Online]. Available: https://developer.sheerid.com/concepts#success.

[3] M. Sporny, D. Longley, and D. Chadwick, *Verifiable credentials data model v1.1*, M. Sporny, G. Noble, D. Longley, D. C. Burnett, and K. D. Hartog, Eds., Nov. 2021. [Online]. Available: https://www.w3.org/TR/vc-data-model/.

[4] *Json-ld 1.1*. [Online]. Available: https://www.w3.org/TR/json-ld11/.

[5] G. Paravicini, *Eu's passport fraud 'epidemic'*, Jan. 2017. [Online]. Available: https://www.politico.eu/article/europes-fake-forged-stolen-passport-epidemic-visa-free-travel-rights/.

[6] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems (extended abstract)," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, R. Sedgewick, Ed., ACM, 1985, pp. 291–304. DOI: 10.1145/22145.22178. [Online]. Available: https://doi.org/10.1145/22145.22178.

[7] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, 1994. DOI: 10.1007/BF00195207. [Online]. Available: https://doi.org/10.1007/BF00195207.

[8] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discret. Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, 2008. DOI: 10.1016/j.dam.2007.12.010. [Online]. Available: https://doi.org/10.1016/j.dam.2007.12.010.

[9] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, J. Feigenbaum, Ed., ser. Lecture Notes in Computer Science, vol. 576, Springer, 1991, pp. 129–140. DOI: 10.1007/3-540-46766-1_9. [Online]. Available: https://doi.org/10.1007/3-540-46766-1%5C_9.

[10] 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*, S. Cimato, C. Galdi, and G. Persiano, Eds., ser. Lecture Notes in Computer Science, vol. 2576, Springer, 2002, pp. 268–289. DOI: 10.1007/3-540-36413-7_20. [Online]. Available: https://doi.org/10.1007/3-540-36413-7%5C_20.

[11] B. Zundel, *Why the verifiable credentials community should converge on bbs+*, Mar. 2021. [Online]. Available: https://www.evernym.com/blog/bbs-verifiable-credentials/.

[12] J. C. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital sinatures (extended abstract)," in *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, T. Helleseth, Ed., ser. Lecture Notes in Computer Science, vol. 765, Springer, 1993, pp. 274–285. DOI: 10.1007/3-540-48285-7_24. [Online]. Available: https://doi.org/10.1007/3-540-48285-7%5C_24.

[13] P. Camacho, A. Hevia, M. A. Kiwi, and R. Opazo, "Strong accumulators from collision-resistant hashing," *Int. J. Inf. Sec.*, vol. 11, no. 5, pp. 349–363, 2012. DOI: 10.1007/s10207-012-0169-2. [Online]. Available: https://doi.org/10.1007/s10207-012-0169-2.

[14] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, J. Katz and M. Yung, Eds., ser. Lecture Notes in Computer Science, vol. 4521, Springer, 2007, pp. 253–269. DOI: 10.1007/978-3-540-72738-5_17. [Online]. Available: https://doi.org/10.1007/978-3-540-72738-5%5C_17.

[15] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, A. Menezes, Ed., ser. Lecture Notes in Computer Science, vol. 3376, Springer, 2005, pp. 275–292. DOI: 10.1007/978-3-540-30574-3_19. [Online]. Available: https://doi.org/10.1007/978-3-540-30574-3%5C_19.

[16] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," in *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, S. Jarecki and G. Tsudik, Eds., ser. Lecture Notes in Computer Science, vol. 5443, Springer, 2009, pp. 481–500. DOI: 10.1007/978-3-642-00468-1_27. [Online]. Available: https://doi.org/10.1007/978-3-642-00468-1%5C_27.

[17] I. Karantaidou and F. Baldimtsi, "Efficient constructions of pairing based accumulators," *IACR Cryptol. ePrint Arch.*, p. 638, 2021. [Online]. Available: https://eprint.iacr.org/2021/638.

[18] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, *et al.*, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, IEEE, 2017, pp. 301–315. DOI: 10.1109/EuroSP.2017.13. [Online]. Available: https://doi.org/10.1109/EuroSP.2017.13.

[19] I. Surname, I. Surname, and I. Surname, "The title of the article," *The Title of the Journal*, vol. 1, no. 2, pp. 123–456, 2000.

[20] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21 260, 2008.

[21] *Blockchain*, Jun. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Blockchain.

[22] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, 2016.

[23] *Blwasp*. [Online]. Available: https://github.com/BlWasp/Fido2VC%5C_FirefoxExtension.

[24] *Spruce*. [Online]. Available: https://spruceid.com/.

[25] D. W. Chadwick, R. Laborde, A. Oglaza, R. Venant, A. S. Wazan, and M. Nijjar, "Improved identity management with verifiable credentials and FIDO," *IEEE Commun. Stand. Mag.*, vol. 3, no. 4, pp. 14–20, 2019. DOI: 10.1109/MCOMSTD.001.1900020. [Online]. Available: https://doi.org/10.1109/MCOMSTD.001.1900020.

[26] *Vc-js*. [Online]. Available: https://github.com/digitalbazaar/vc-js.

[27] *Openattestation*. [Online]. Available: https://www.openattestation.com/docs/docs-section/how-does-it-work/comparison.

[28] *Veramo*. [Online]. Available: https://veramo.io/docs/basics/introduction.

[29] *Uport*. [Online]. Available: https://www.uport.me/.

[30] C. Allen. "The path to self-sovereign identity." (2016), [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[31] *Hyperledger indy*. [Online]. Available: https://github.com/hyperledger/indy-sdk/blob/master/docs/getting-started/indy-walkthrough.md.

[32] *Irma*. [Online]. Available: https://irma.app/.

[33] *Verifiable-credential-java*. [Online]. Available: https://github.com/danubetech/verifiable-credentials-java.

[34] *Gravity*. [Online]. Available: https://docs.gravity.earth/.

[35] V. Shoup, "Sequences of games: A tool for taming complexity in security proofs," *IACR Cryptol. ePrint Arch.*, p. 332, 2004. [Online]. Available: http://eprint.iacr.org/2004/332.

[36] J. Camenisch, M. Drijvers, and A. Lehmann, “Anonymous attestation using the strong diffie hellman assumption revisited,” *IACR Cryptol. ePrint Arch.*, p. 663, 2016. [Online]. Available: `http://eprint.iacr.org/2016/663`.

[37] *Hyperledger iroha*. [Online]. Available: `https://www.hyperledger.org/use/iroha`.

[38] I. Insights, *The 10 states with the most suspended/revoked licenses*. [Online]. Available: `https://insurify.com/insights/the-10-states-with-the-most-suspended-revoked-licenses/`.

A Appendix

Nomenclature

Abbreviations

Abbreviation	Definition
VC	Verifiable credential. See Introduction.
VP	Verifiable Presentation. See Introduction.
PU	Proof of unrevocation. See Chapter 4.
CRL	Certificate Revocation List. A list of revoked credential serial numbers (revocation handles), used in the WebPKI.
OCSP	Online Certificate Status Protocol (OCSP). A protocol to retrieve the revocation status of a X.509 certificate from the issuing Certificate Authority, as used in the WebPKI.
PPT	Probabilistic Polynomial-Time (PPT). The complexity class of algorithms that run in Probabilistic Polynomial Time with regards to the input size.
rh	Revocation handler. See Chapter 4.
RI	Revocation Information. See Chapter 4.
SPK	Signature Proof of Knowledge (SPK). See section 2.1

Symbols

Symbol	Definition
$\leftarrow\$$	random sampling
$\leftarrow$	assign operation
l	The length of a value. For example, l_c means the length of c
$_R\{0,1\}^{l_c}$	The set of random bitstrings of length l_c
$\mathcal{H}_s$	cryptographic hash function
$\mathbb{Z}_q$	Integer group with order q
$\mathbb{Z}_q^*$	Non-zero integer group with order q
$\mathcal{A}$	Probabilistic Polynomial Time (PPT) adversary
b	The correct result of a security game
b'	The guess result of the PPT adversary
st	stored information
$SPGen$	The generation algorithm of system parameters
$spar$	system parameters
λ	security parameter

List of Figures

List of Tables

List of Protocols

List of Algorithms