Verifiable Credentials with Privacy-Preserving Tamper-Evident Revocation Mechanism

Xu, Li; Li, Tianyu; Erkin, Zekeriya

DOI
10.1109/BCCA58897.2023.10338923

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)

Verifiable Credentials with Privacy-Preserving Tamper-Evident Revocation Mechanism

Li Xu
Cyber Security Group
Delft University of Technology
Delft, the Netherlands
L.Xu-11@student.tudelft.nl

Tianyu Li
Cyber Security Group
Delft University of Technology
Delft, the Netherlands
Tianyu.Li@tudelft.nl

Zekeriya Erkin
Cyber Security Group
Delft University of Technology
Delft, the Netherlands
Z.Erkin@tudelft.nl

2023 Fifth International Conference on Blockchain Computing and Applications (BCCA) | 979-8-3503-3923-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/BCCA58897.2023.10338923

***Abstract*—Verifiable Credential (VC) is a new standard proposed by the W3C association to facilitate the expression and verification of third-party-verified credentials on the Internet, such as passports or diplomas. However, the current VC data model lacks an explicit revocation design that guarantees the secure operations of the system, which limits its application. In this paper, we specify the requirements for a tamper-evident and privacy-preserving revocation mechanism, based on which we compare existing solutions and propose our revocation mechanism that satisfies all the requirements. Our design combines a cryptographic accumulator and a role-based blockchain. With zero-knowledge proof, the verifier can operate off-chain computation of the revocation status while ensuring the correctness of revocation information published on the blockchain. Our analysis shows that the proposed revocation mechanism can prevent fraud using forged and revoked credentials and relieve privacy concerns caused by the correlation of digital data. Our proof-of-concept implementation demonstrates that our revocation mechanism adds only 42.86 ms overhead in the presentation and 31.36 ms overhead in the verification of verifiable credentials. We also provide scalability analysis, which illustrates that the throughput of our blockchain can meet real-world needs.**

***Index Terms*—verifiable credential, revocation, blockchain, zero-knowledge proof, privacy**

I. Introduction

Verified third-party credentials are a part of our everyday lives. Driver licenses assert that someone can operate a vehicle, diplomas represent the level of education, and passports enable people to travel across countries. These credentials provide convenience to us when used in the physical world. As the Internet has become more indispensable, there is an increasing need for users to use verified third-party information on the Web. However, there lacks a uniform format, and a consistent verification method makes it challenging to represent third-party-verified credentials on the Web [1]. Besides, two factors comprise a severe privacy concern when using the verified third-party credential on the Internet: the persistence of digital data and the ease with which disparate digital data sources can be collected and correlated.

To ease using third-party-verified credentials on the Internet and relieve privacy concerns, the W3C proposes the Verifiable Credential data model. Verifiable credentials are designed to represent the information of physical credentials in a tamper-evident and privacy-preserving manner. Figure 1 shows the ecosystem of the verifiable credential data model. There are four actors in a verifiable credential system: issuer, holder, verifier, and verifiable data registry. The issuer issues the verifiable credentials to the holder. The holder stores verifiable credentials and uses them to generate verifiable presentations. The verifier requests a verifiable presentation from the holder and checks its validity when asked for service. The verifiable data registry maintains the auxiliary information for issuance.

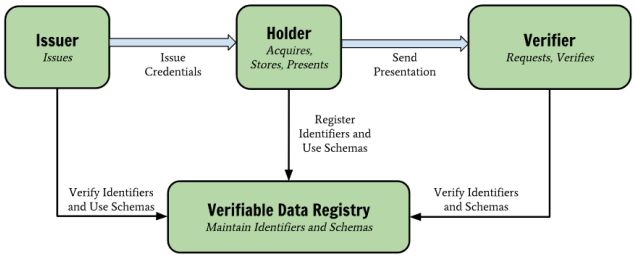


Fig. 1. Ecosystem of verifiable credentials.

Verifiable credentials represent users' attributes in a subject-property-value way called claims. For example, if Bob is 22 years old, it shows "Bob-age-22" on the credential. The issuer signs each claim on the verifiable credential and guarantees its authenticity. In addition, the signature of verifiable credentials allows the holder to selectively disclose the signed claims when presenting the credential to verifiers. Thus, the presentation of verifiable credentials is tamper-evident and privacy-preserving. In other words, the attackers can not modify the claims signed by the issuer. They can not track the usage of holders' verifiable credentials if the user chooses not to disclose identifying claims.

Aside from issuance and presentation, revocation is important in the credential system. Revocation allows the issuer to rescind the validity of the issued credentials when the user misbehaves or the credentials are broken. The employment of revocation can significantly reduce the frauds caused by stolen and lost credentials [2]. Meanwhile, the appropriate revocation can relieve the concern that terrorists use stolen credentials to commit cross-board crimes [3]. Thus, it is necessary to check the revocation status of a given credential. The process of checking revocation status is called revocation status.

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 a scenario that needs a rigid validity check, such as applying for a loan from banks or health allowance). Some existing works [4]–[9] proposed a revocation mechanism for verifiable credentials. Nevertheless, the existing revocation mechanisms suffer from two types of threats: privacy leakage caused by revocation status and the issue of centralization. The designs of [4]–[6], [8], [9] leave a space for the attackers to correlate verifiable credentials with revocation status, which violates the selective disclosure for verifiable credential data models. The design of [7] suffers from the threat of the central server's breakdown. In conclusion, the existing revocation mechanisms for verifiable credentials fail to achieve privacy-preserving and tamper-evident simultaneously.

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 specify the requirements for a temper-evident and privacy-preserving revocation mechanism. Based on that, we analyze and compare existing solutions. Also, we propose a revocation mechanism that satisfies all the requirements by combining a blockchain and an efficient bilinear pairing-based accumulator. The revocation and the revocation status check do not leak any identifying information about verifiable credentials and the holder of verifiable credentials. Adversaries can not pass the revocation status check with forged or revoked credentials.
- We implement a proof of concept of our revocation mechanism in *Rust*. Our results show that adding our revocation mechanism to a verifiable credential system without a revocation mechanism adds 40ms runtime overhead to presentation and 30ms runtime overhead to verification. Since the absolute figure of the runtime overhead is limited to tens of milliseconds, our revocation mechanism is practical in real use.

The rest of the paper is organized as follows: Section II gives a brief description of the background knowledge for the related works and our design. Section III introduces and analyzes related works. Section IV describes our proposed revocation mechanism. Section V analyzes the security properties and performance of the proposed system. Section VI shows the experimental measurements of our design. The discussion and conclusion are provided in section VII.

II. Preliminaries

A **verifiable credential** is a set of claims that describes the attributes of the holder. The user can use the verifiable credentials he holds to generate a verifiable presentation and send the verifiable presentation to the verifier for authentication. Figure 2 shows the basic structure of the verifiable credential and verifiable presentation. A Verifiable Credential (verifiable credential) consists of credential metadata, claims, and proofs. Credential metadata describes properties of the credential, such as the issuer, the expiry date, and the public keys to verify proofs. Claims are subject-property-value relationships (e.g."Alice"-"is an alumnus of"-"Example University"). Proofs are the signatures of the claims. A Verifiable Presentation (VP) contains one or more verifiable credentials and has metadata and proofs. The metadata describes its usage, and the proofs are used to prove the verifiable credentials embedded in the VP belong to the same holder. The verifiable credentials contained in a VP may only have selective claims. For instance, if a user generates a VP for his ID card to prove he is an adult, the verifiable credential in the VP only shows the user's age; other claims are hidden to protect his privacy.

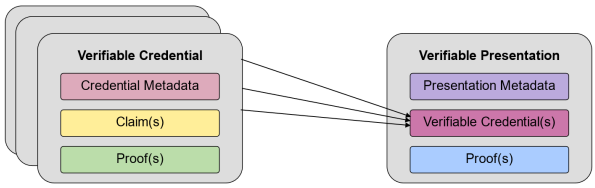


Fig. 2. Basic structure of Verifiable credentials and Verifiable presentations

Signature Proof-of-Knowledge (SPK) is a non-interactive Zero-Knowledge Proof (ZKP) technique. Traditional zero-knowledge protocols like Σ-protocols [10] are interactive, which means the prover needs to solve the challenge given by the verifier to prove he knows specific knowledge. SPK applies Fiat-Shamir transformation [11] to enable the prover can generate the challenge herself. Therefore, the interactive proof protocols become a non-interactive signature. The verifier can check the validity of the proof by verifying the signature. SPK can be utilized as proof of claims for verifiable credentials.

Pairing, also known as bilinear maps, indicates the operation that maps pairs of points on two elliptic curves into a finite field. According to the elliptic curves used, Galbraith et al. [12] distinguish three types of pairings: type-1, which use two identical elliptic curves; type-2, which uses two different curves and there exists an efficient isomorphism between the two curves; and type-3, which uses two different curves, but no isomorphism exists. Currently, type-3 pairings are the most desirable choice to describe a cryptographic scheme as it provides the most efficient operations on the curves [13].

BBS+ signature [13] is a type-3 pairing-based signature which support selectively disclosure of signed values. BBS+ signature has two advantages over the RSA-based CL-signature [14] used by most existing verifiable credential implementations: the key generation and signing are faster, and the size of keys and claims are smaller [15]. Combining BBS+ signature and JSON-LD grants good interoperability to verifiable credentials [15]. Therefore, the BBS+ signature is now the most suitable signature for verifiable credentials.

A **cryptographic accumulator** allows the user to represent 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 [16], RSA-based [14],

[17], [18] and pairing-based [19]–[21]. 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 [22].

III. Related Works

In the past few years, several implementations of verifiable credential data models have been proposed, but many of them do not support revocation mechanisms [23]–[26]. In this paper, we focus on verifiable credential implementations with revocation mechanisms. We evaluate the existing works in three dimensions: (1) privacy-preserving performance, (2) tamper-evident performance, and (3) revocation information storage cost. In this section, 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.

OpenAttestation (OA) [4] is an open-sourced framework to endorse and verify verifiable credentials based on Ethereum Smart Contracts. OpenAttestation provides two ways to revoke a verifiable credential: using a document store or an Online Certificate Status Protocol (OCSP). A document store is a smart contract on the Ethereum network that records the issuance and revocation status of OA documents. 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. OCSP checks the revocation status of a verifiable credential by querying its certificate ID from an OCSP responder. The appliance of smart contracts made the revocation tamper-evident, but OA suffers privacy leakage issues. Both the document store and OCSP require identifiers to check the revocation status. The verifier can use the identifiers used in the revocation status check to correlate different presented verifiable credentials. For example, suppose a user disclosed different claims in two presentations. The attacker can use the identifiers to correlate the different claims to the same credential, violating the verifiable credential data model's selective disclosure. Besides, 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.

Veramo [5] is the open-source version of uPort project [27]. 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 uses a smart contract called ethr-status-registry to carry out the revocation mechanism. Similar to OA, smart contracts ensure the revocation is tamper-evident. The ethr-status-registry takes the hash value of a credential as input for the revocation status check. Since the hash value is static, attackers can employ it to correlate verifiable credentials. Therefore, the revocation mechanism can cause leakage of users' personal information, which is not privacy-preserving. Veramo also needs to maintain revocation status for all credentials it issues, so the storage cost is linear with the number of credentials.

Sorvin [28] is a Self-Sovereign Identity (SSI) [29] solution based on HyperLedger Indy [30] public permissioned blockchain. Sorvin supports the user to use of verifiable credentials as the authentication method. The revocation of Sovrin is done using the CKS accumulator [20] and the Indy blockchain. The CKS accumulator indexes the accumulated values and publishes an index list that records the current valid indices. Each time a new member joins or an old member leaves, the accumulator needs to add/delete an index from the index list. 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 since CL-signature and blockchain guarantee the authenticity of published information. As a 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 valid credentials.

I Reveal My Attributes (IRMA) [7] 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 his identity. The revocation of IRMA is done based on the CL-RSA-B accumulator [18], which only requires updating the accumulator value and witnesses for users when revocation happens. It reaches the optimal communication complexity of accumulators. In addition, updating witnesses of the CL-RSA-B accumulator does not need knowledge about current valid credentials, preventing the joint-revoke attack. However, IRMA uses a central server to process and store the data. It suffers from possible crashes, which could happen through malicious censorship, hacking, or natural calamity. Although IRMA employs Redis as the data storage, a single Redis datastore still suffers from possible tampers.

Verifiable-Credential-Java (VCJ) [9] 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 (and 0 if not revoked). A benefit of Revocation List 2020 is the bitstring can be compressed to save storage. However, the revocation list index is a kind of identifier that can correlate 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.

Gravity [8] is a decentralized cloud platform where individuals can receive, store, and share verifiable credentials in a secure wallet they fully control. With decentralization, Gravity achieves tamper-evident for revocation, but it suffers from correlation since Revocation List 2020 is used as the revocation mechanism. The storage cost is linear with the number of credentials with Revocation List 2020.

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

IV. System design

This section introduces the design of our revocation mechanism. Here "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 holder to state the verifiable credential is not revoked.

A. Trust model

Our revocation mechanism follows the trust model of the verifiable credential data model, which means: that the verifier trusts the issuer to issue the credential that it received; all entities trust the verifiable data registry to be a tamper-evident record of published data; the holder and verifier trust the issuer to issue valid credentials; the holder trust the repository to store credentials securely. Based on the trust model, we give our definition of tamper-evident and privacy-preserving:

Definition IV.1 (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 a certain degree of hacking and disaster.*

Definition IV.2 (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.*

B. Requirements

For the definitions, 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 is the unforgeability property.
(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 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) guarantees the proof of unrevocation is unlinkable. The holder repeatedly presents the verifiable credential, so he needs to generate multiple proofs of unrevocation for the same verifiable credential. Multi-show unlinkable means the verifier can not use the proof of unrevocation to link verifiable credentials. Given two different proofs of unrevocation, the verifier can not determine if they are generated from the same or different verifiable credentials. Since 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. The proof of unrevocation should be untraceable for the issuer. Additionally, revocation information may suffer from join-revoke linkability, so we set the requirement (4) against it.

C. Revocation mechanism

Our revocation mechanism has three building blocks: an efficient type-3 pairing-based accumulator, the BBS+ signature scheme, and a role-based blockchain. The accumulator is a type-3 version of the accumulator described in [21]. It achieves the same privacy guarantees as the CL-RSA-B accumulator but has faster speed and a smaller key size. We incorporate the accumulator with the BBS+ signature to make it interoperable for different verifiable credentials formats. The role-based blockchain is utilized as a tamper-evident ledger to record and publish the revocation information. For the requirements of tamper-evident and privacy-preserving, we achieve (1) and (3) by incorporating the BBS+ signature and the efficient pairing-based accumulator, (2) by the employment of a role-based blockchain, and (4) by the pairing-based accumulator.

We assume four actor types in our revocation mechanism: issuer, holder, verifier, and verifiable data registry. Only the issuer can write to the verifiable data registry. The holder and the verifier can only read the verifiable data registry. In this section, the term "user" used in credential issuance is the same as the "holder". "User" refers to the state that the holder has not gotten the credential from the issuer. Since the verifiable data registry is a blockchain, we use blockchain to refer to it. The issuer generates blocks on the blockchain, whereas the holder and verifier validate the blocks.

Due to the lack of space, we cannot present the details of our protocols and algorithms. Instead, we will put the pseudo code and reference to a detailed description of relevant algorithms.

TABLE I
REVOCATION COMPARISON WITH RELATED WORKS

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

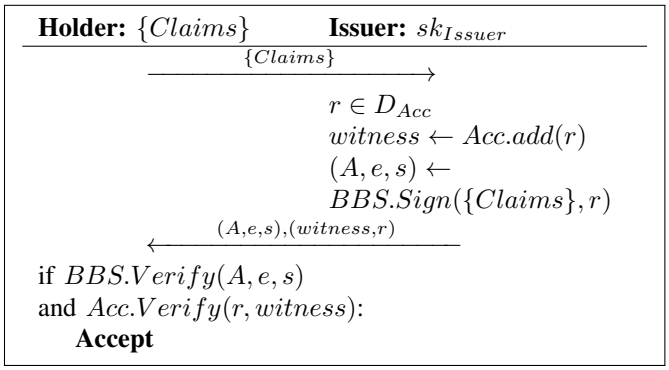
1) System initialization: Before the issuance, the issuer sets up: the bilinear pairing-based accumulator, the BBS+ signature and the Pedersen commitment scheme [31]. The accumulator is used to accumulate the revocation handler and generate a witness for the revocation handler. The BBS+ signature is to sign the claims provided by the user, the revocation handler generated by the issuer, and the revocation handler's witness. The Pedersen commitment is used to generate proof of unrevocation. Algorithm 1 shows the pseudo-code of initializing the system. The algorithm takes a pairing instance $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and the number of claims N_{claims} as input. The pairing instance provides the elliptic curves required by the cryptographic tools. The detailed description of the $BBS.KeyGen$, $Acc.Gen$, $Commitment.SetUp$ can be found in [13], [21] and [31]. Note that $BBS.KeyGen$ takes $N_{claims} + 1$ as the input because the issuer needs to sign the revocation handler aside from the provided claims. Moreover, $Acc.Gen$ takes a type-3 pairing instance, which means the initial accumulator value acc_0 is generated using g_2.

Algorithm 1 *SetUp*: Set up the cryptographic tools

Input: $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), N_{claims}$
Output: $(PK_{issuer}, SK_{issuer}), acc_0$
1: $(PK_{BBS}, SK_{BBS}) \leftarrow BBS.KeyGen(\mathcal{P}, N_{claims} + 1)$
2: $(PK_{Acc}, SK_{Acc}, acc||0) \leftarrow Acc.Gen(\mathcal{P})$
3: $PK_{Com} \leftarrow Commitment.SetUp(\mathcal{P})$
4: $PK_{issuer} \leftarrow (PK_{BBS}, PK_{Acc}, PK_{Com})$
5: $SK_{issuer} \leftarrow (SK_{BBS}, SK_{Acc})$
6: **return** $(PK_{issuer}, SK_{issuer}, acc_0)$

2) Credential Issuance: Credential Issuance refers to how the issuer signs the values obtained from the user. Before signing the values, the issuer generates a random value r from the domain of the accumulator D_{Acc} as the revocation handler and accumulates r in the accumulator. The accumulator generates a $witness$ for the accumulated r. The issuer signs the given claims and the revocation handler r together. Finally, the user puts the signature and (r, $witness$) pair in the credential's proof part and sends it back to the user. When receiving the credential, the user verifies the revocation handler r is signed and accumulated in the accumulator. Protocol 1 shows the credential issuance. Detailed descriptions of BBS+ signature and the accumulator can be found in [13] and [21].

3) Credential presentation and presentation verification: Credential presentation notices the operation of using one or more verifiable credentials to generate a presentation that meets the requirements. During the credential presentation, the holder does not reveal the $(r, witness)$ for each verifiable credential included in the presentation. Instead, the holder generates proof of unrevocation to assert the presented credential is not revoked. The proof of unrevocation is a zero-knowledge statement about the $(r, witness)$. We can describe the ZK statement as follows (using Camenisch-Stadler notation [32]):



Protocol 1. Protocol of credential issuance.

$$\begin{aligned} ZKP = [(r, witness) : & Commit.Verify(r) \\ & \wedge Acc.VerMem(r, witness) \\ & \wedge BBS.verify(r)](acc, PK_{issuer}, C_r) \end{aligned} \quad (1)$$

The zero-knowledge statement indicates the prover can prove that the committed revocation handler C_r is signed by the issuer and r is accumulated in the accumulator without revealing $(r, witness)$ to the verifier. To verify the given zero-knowledge statement, the verifier only needs to know the current accumulator value acc, the issuer's public key PK_{issuer}, and the Pedersen commitment of r C_r to finish the verification. For more details about individual protocols in this proof, please refer to [33], [21] and [13]. Running the above zero-knowledge proof protocols, the holder will generate two SPKs: SPK_{BBS}, the SPK for the BBS+ signature and SPK_{ACC}, the SPK for the accumulator. SPK_{BBS} is the zero-knowledge proof for the claims, whereas SPK_{ACC} function as the proof of unrevocation. By verifying the two SPKs, the verifier can check the revocation status of a presented credential.

4) Revocation: The revocation of a verifiable credential is a two-step process. Firstly, the issuer removes the verifiable credential's revocation handler and updates the accumulator value. Secondly, the issuer publishes 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 his verifiable credentials' witnesses. Otherwise, he 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. The issuer writes the revocation information to the blockchain in the format of (acc, rh, v) where acc is the accumulator value. rh is the revocation handler of the recently revoked verifiable credential. v is the version number of the revocation handler. The holder stores the last used revocation information. Each time the issuer revokes a verifiable credential, it will make a transaction on the blockchain, and the transaction will generate a new block that documents the newest revocation information. When the holder needs to generate a revocation handler, he compares the version number of stored revocation information and the latest revocation information on the data registry. If two version numbers are the same, the holder does not need to update the witness for the verifiable credential. Algorithm 2 shows the process of using revocation information to update the witness. $RI_{local}, RI_{latest}, \{RI_{blockchain}\}$ refer to the local revocation information, the latest revocation information, and the set of revocation information stored on the blockchain. The detail of $Acc.WitUp$ can be found in [21].

Algorithm 2 *Update*: Update the witness for a VC

Input: $RI_{local}, RI_{latest}, witness_{old}, \{RI_{blockchain}\}$
Output: $witness_{new}$

$v_{local} \leftarrow RI_{local}$
$V_{latest} \leftarrow RI_{latest}$
if $v_{local} == v_{latest}$ **then**
  $witness_{new} = witness_{old}$
else
  for $v = v_{local} + 1..v_{latest}$ **do**
    Find RI with version v in $\{RI_{blockchain}\}$
    $(acc, rh) \leftarrow RI$
    $witness_{new} = Acc.WitUp(witness_{old}, acc, rh)$
  end for
end if
return $witness_{new}$

V. Analysis

A. Security and privacy analysis

Theorem V.1 (Unforgeability)**.** *Our revocation mechanism provides unforgeability: a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can not 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, $\mathcal{A}$ needs to prove a revoked revocation handler is accumulated in the accumulator. This is infeasible under the strong Diffie-Hellman assumption [18]. To pass the authentication with the proof of unrevocation for other valid credentials, $\mathcal{A}$ 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 refer to [13]. □

Theorem V.2 (Multi-show Unlinkability)**.** *Our revocation mechanism provides multi-show unlinkability: a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to determine if two given proof of unrevocation are generated from the same or different verifiable credentials.*

Proof. The multi-show unlinkability is achieved by the zero-knowledge proof of the BBS+ signature, the efficient pairing-based accumulator, and the Pedersen commitment. Since the zero-knowledge proofs do not leak any knowledge about the identifying signature, revocation handler, and revocation handler's witness, the adversary can not use proofs of unrevocation to link the verifiable credentials. □

Theorem V.3 (Issuer Untraceability)**.** *Our revocation mechanism provides issuer untraceability: given a verifiable credential and a proof of unrevocation, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can not determine if the given verifiable credential generates the proof of unrevocation.*

Proof. Similar to the proof of multi-show unlinkability, the zero-knowledge property guarantees no identifying information leakage under the strong Diffie-Hellman assumption. As a result, the issuer cannot use proof of unrevocation to distinguish verifiable credentials. □

Theorem V.4 (Join-revoke Unlinkability)**.** *Our revocation mechanism provides join-revoke unlinkability: given a verifiable credential and proof of unrevocation, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is unable to determine when the given credential is issued.*

Proof. The efficient pairing-based accumulator achieves the join-revoke unlinkability successfully. For the detailed description, we refer the readers to read the appendix of [18]. □

B. Performance analysis

Here we analyze the impact of adding our revocation mechanism to a verifiable credential system. Figure 3 shows the lifecycle of a verifiable credential. There are repeatable operations such as the presentation of a verifiable credential, the verification of presented verifiable credentials, and the revocation status check. In the verifiable credential system, the revocation status check and the verification of presented verifiable credentials can be merged. The performance of the system (with revocation) is dominated by two operations: verifiable credential presentation and verifiable credential verification.

The verifiable credential presentation includes generating SPK_{BBS} and 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}. [21] shows that generating

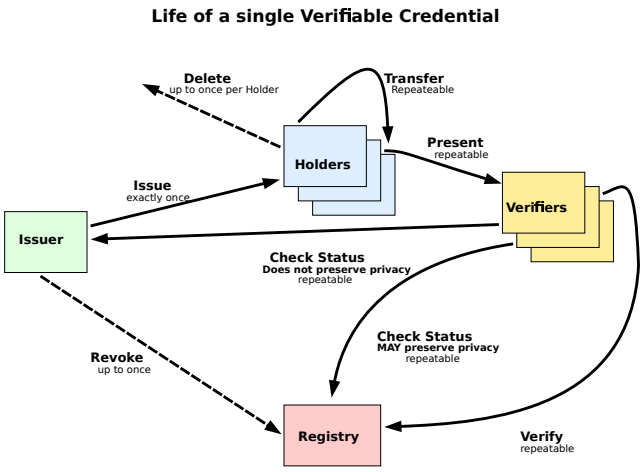


Fig. 3. The lifecycle of a verifiable credential [1].

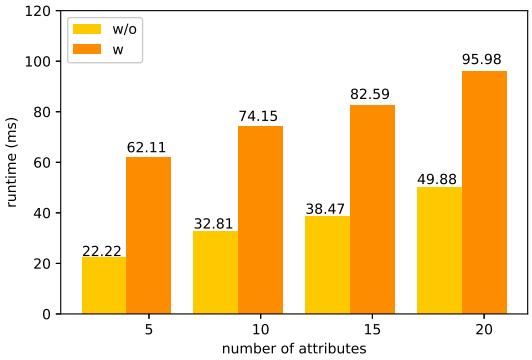


Fig. 4. Runtime measurements of verifiable credential presentation

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 $O(1)$.

C. Storage

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

VI. Experimental Results

A. Runtime of off-chain zero-knowledge proof

This section measures the experimental runtime of our revocation mechanism by comparing the credential system

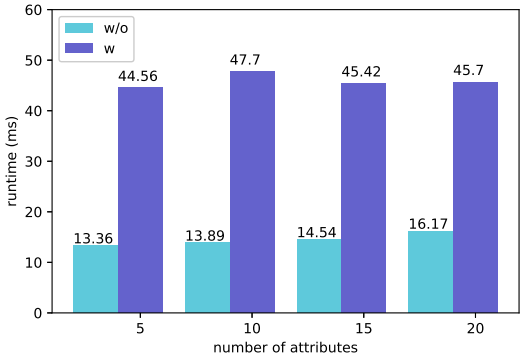


Fig. 5. Runtime measurements of verifiable credential verification.

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). Similar to the performance analysis in Section V, we compare the runtime for verifiable credential presentation and verifiable credential verification. We use credentials with 5, 10, 15, and 20 attributes for the evaluation. The proof-of-concept implementation is written in *Rust*. The pairing-friendly curve is Bls12-381. The random generator is thread_rng(). The measurements were performed with macOS 12.2.1, Apple M1 chip, and 8GB memory.

Figure 4 shows the runtime of verifiable credential presentation. For the system without a revocation mechanism, the runtime only involves the generation of SPK_{BBS}. When our revocation mechanism is embedded, both SPK_{BBS} and SPK_{ACC} are considered. The average runtime overhead by our revocation mechanism on verifiable credential presentation is 42.86 ms. Our analysis in Section V shows the runtime overhead has a constant computation complexity. Thus, the runtime overhead will not fluctuate significantly when the number of claims changes in a credential. The generation of SPK_{BBS} is linear with the number of claims in the credentials [13]. When the number of claims in the credential increases, the generation of SPK_{BBS} gradually dominates the runtime of verifiable credential presentation. In other words, the impact of our revocation mechanism on verifiable credential presentation decreases with an increasing number of claims contained in a verifiable credential, which is probably the case in the future with more complex scenarios for verifiable credentials.

Figure 5 shows the runtime of verifiable credential verification. Without a revocation mechanism, the runtime is the verification of SPK_{BBS}. With our revocation mechanism, the runtime is the overall measurement of verifying SPK_{BBS} and SPK_{ACC}. Our revocation mechanism brings 31.36 ms average runtime overhead, and the average overall verification runtime is 45.84 ms. Section V indicates the computation complexity of verifiable credential verification is $\mathcal{O}(1)$. Therefore, the runtime of verifiable credentials verification does not change significantly with an increasing number of claims.

Overall, the overhead brought by the revocation mechanism is around tens of milliseconds, and the overhead is (near) constant with the number of claims in a verifiable credential. Considering the density and number of verifiable credentials

being verified per day, the addition of the revocation mechanism is practical in most scenarios.

B. On-chain scalability

We combine our proof of concept implementation with Hyperledger Iroha [34] to test the scalability of our system. We evaluate the transaction per second (TPS) of our mechanism using the tools provided by Iroha. The test is run on the same machine we employ in the runtime measurements of off-chain computation. The TPS of our revocation mechanism with four nodes is 70 transactions/second, so our system can support 6,048,000 revocations per day. We believe the TPS is sufficient for practical use since revocation is an infrequent operation. From 2015 to 2016, 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 41,263 [35]. In other words, there are only, on average, 113 revoked credentials per day. The throughput of our revocation mechanism is far more extensive than 113 transactions/day. Therefore, our revocation mechanism is scalable in the real world.

VII. Discussion and Conclusion

In this paper, we specify the requirements for a tamper-evident and privacy-preserving revocation mechanism. Among existing revocation mechanisms, smart contracts, OCSP, and revocation list 2020 are used 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 correlate the credentials through a join-revoke attack. The appliance of CL-RSA-B makes IRMA outperforms other works on the enhancement of privacy. However, the central server makes it lose in the performance of tamper-evident. There is no work achieving both tamper-evident and privacy-preserving revocation.

Based on the definition of tamper-evident and privacy-preserving, we propose a revocation mechanism satisfying all the requirements. Table I compares our revocation mechanism with the existing works. Our design combines zero-knowledge proofs of accumulators and BBS+ signature to ensure the revocation status check does not leak identifying information about the presented credential. We apply blockchain to guarantee the published revocation information is trustworthy. Experimental results show the performance is feasible for real-world use. We consider our method the first privacy-preserving and tamper-evident revocation mechanism for verifiable credentials.

References

[1] M. Sporny, D. Longley, and D. Chadwick, "Verifiable credentials data model v1.1," Nov 2021. [Online]. Available: w3.org/TR/vc-data-model/

[2] 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/

[3] T. Kean and L. Hamilton, *The 9/11 commission report: Final report of the national commission on terrorist attacks upon the United States*. Government Printing Office, 2004, vol. 3.

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

[5] "veramo." [Online]. Available: veramo.io/docs/basics/introduction

[6] "Sorvin." [Online]. Available: https://sovrin.org/

[7] "Irma." [Online]. Available: https://irma.app/

[8] "Gravity." [Online]. Available: https://docs.gravity.earth/

[9] "verifiable-credentials-java." [Online]. Available: https://github.com/danubetech/verifiable-credentials-java

[10] R. Cramer, "Modular design of secure yet practical cryptographic protocols," *Ph. D. Thesis, CWI and University of Amsterdam*, 1996.

[11] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO '86*, ser. LNCS, vol. 263. Springer, 1986, pp. 186–194.

[12] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008, applications of Algebra to Cryptography.

[13] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong diffie hellman assumption revisited," in *Trust and Trustworthy Computing*. Cham: Springer International Publishing, 2016, pp. 1–20.

[14] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks*. Berlin, Heidelberg: Springer, 2003, pp. 268–289.

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

[16] P. Camacho, A. Hevia, M. Kiwi, and R. Opazo, "Strong accumulators from collision-resistant hashing," *International Journal of Information Security*, vol. 11, pp. 349–363, 2012.

[17] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *Applied Cryptography and Network Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 253–269.

[18] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *EuroS&P 2017*, 2017, pp. 301–315.

[19] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Topics in Cryptology – CT-RSA 2005*. Berlin, Heidelberg: Springer, 2005, pp. 275–292.

[20] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," in *PKC 2009*. Springer, 2009, pp. 481–500.

[21] I. Karantaidou and F. Baldimtsi, "Efficient constructions of pairing based accumulators," in *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. IEEE, 2021, pp. 1–16.

[22] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: Efficient, succinct, modular," in *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer, 2021, pp. 393–414.

[23] "Blwasp." [Online]. Available: https://github.com/BlWasp/Fido2VC_FirefoxExtension

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

[25] D. W. Chadwick, R. Laborde, A. Oglaza, R. Venant, S. Wazan, and M. Nijjar, "Improved identity management with verifiable credentials and fido," *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 14–20, 2019.

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

[27] "uport." [Online]. Available: https://www.uport.me/

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

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

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

[31] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *CRYPTO '91*. Berlin, Heidelberg: Springer, 1992, pp. 129–140.

[32] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *CRYPTO'97*. Springer, 1997, pp. 410–424.

[33] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," in *CRYPTO '97*. Berlin, Heidelberg: Springer, 1997, pp. 16–30.

[34] "Hyperledger iroha." [Online]. Available: https://www.hyperledger.org/use/iroha

[35] 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/