

# Compact and Selective Disclosure for Verifiable Credentials

Alessandro Buldini\*, Carlo Mazzocca†, Rebecca Montanari\*, Selcuk Uluagac‡

\*University of Bologna, Bologna, Italy

†University of Salerno, Fisciano, Italy

‡Florida International University, Miami, Florida, USA

Email: {alessandro.buldini, rebecca.montanari}@unibo.it, cmazzocca@unisa.it, suluagac@fiu.edu

**Abstract**—Self-Sovereign Identity (SSI) is a novel identity model that empowers individuals with full control over their data, enabling them to choose what information to disclose, *with whom*, and *when*. This paradigm is rapidly gaining traction worldwide, supported by numerous initiatives such as the European Digital Identity (EUDI) Regulation or Singapore’s National Digital Identity (NDI). For instance, by 2026, the EUDI Regulation will enable all European citizens to seamlessly access services across Europe using Verifiable Credentials (VCs). A key feature of SSI is the ability to selectively disclose only specific claims within a credential, enhancing privacy protection of the identity owner. This paper proposes a novel mechanism designed to achieve *Compact and Selective Disclosure* for VCs (CSD-JWT). Our method leverages a cryptographic accumulator to encode claims within a credential to a unique, compact representation. We implemented CSD-JWT as an open-source solution and extensively evaluated its performance under various conditions. CSD-JWT provides significant memory savings, reducing usage by up to 46% compared to the state-of-the-art. It also minimizes network overhead by producing remarkably smaller Verifiable Presentations (VPs), reduced in size by 27% to 93%. Such features make CSD-JWT especially well-suited for resource-constrained devices, including hardware wallets designed for managing credentials.

**Index Terms**—Self-Sovereign Identity (SSI), Verifiable Credentials (VCs), Selective Disclosure, Digital Identity, Privacy.

## I. INTRODUCTION

The growing emphasis on preserving user privacy has driven a significant transformation in the digital identity ecosystem, shifting from centralized to decentralized models [1]. Emerging privacy-preserving regulations further reinforce this evolution [2]–[4] that place individuals at the center of the identity management process. This paradigm shift has given rise to *Self-Sovereign Identity* (SSI) [5], a novel identity model that is gaining worldwide traction. In February 2024, the European Commission approved the “electronic IDentification, Authentication, and trust Services” (eIDAS 2.0) [6], mandating the creation of the European Digital Identity Wallet (EUDI Wallet) [7]. Under this framework, by 2026, European member states must provide citizens with a digital identity wallet for managing their credentials. Similarly, Singapore’s National Digital Identity (NDI) system [8], implemented through Singpass, enables citizens to seamlessly access public and private services while maintaining full ownership of personal data.

The core principle of SSI consists of empowering individuals with full control over their data by allowing them to decide what information to disclose, with whom, and when, eliminating reliance on centralized authorities. This is achieved through Verifiable Credentials (VCs) [9], digital credentials that attest specific properties or attributes of the holder, hereinafter called *claims*. Unlike traditional Public Key Infrastructure (PKI) certificates [10], which provide secure public key distribution and rely on centralized authorities, VCs are designed for broader use cases and operate within a decentralized framework. These credentials are cryptographically signed by trusted issuers such as governments, enabling decentralized verification of the stored information. A key feature of SSI is the ability for identity owners to *selectively disclose* only the specific subset of claims contained in VCs. For instance, individuals can prove a legal drinking age without revealing unnecessary information like their address.

Various VC formats have been proposed [11], with JSON Web Token (JWT) [12] emerging as one of the most widely adopted as it offers a concise approach to exchanging verifiable claims. To enable privacy-preserving selective disclosure, the Internet Engineering Task Force (IETF) proposed the Selective Disclosure for JWTs (SD-JWT) [13] specification. SD-JWT replaces plain text claims with digests of their salted values. The issuer signs the resulting JWT and shares it with the holder, along with the original claims and salts. To disclose data, the holder provides the verifier with the signed JWT and salt-claim pairs corresponding to disclosed claims.

**Problem Definition.** A major downside of this method is that credential size grows linearly with the number of claims, increasing end-user storage and network requirements. Moreover, although digests hide undisclosed information, SD-JWT reveals the exact number of included claims, raising privacy concerns. This knowledge can potentially expose the identity owner to *inference attacks* [14], [15] as the number of claims may be exploited to infer details about hidden information.

These drawbacks of SD-JWT increase when VCs are applied to entities beyond individuals, such as Internet of Things (IoT) devices [16]–[18]. Given the heterogeneous capabilities of these devices, considerations around storage, computation, and networking are even more crucial. In constrained environments, resources are shared across different functions. For example, memory must store both VCs and firmware, making

even KB-level reductions impactful. Moreover, minimizing resource usage is essential for enhancing energy efficiency, a critical factor in resource-constrained settings. These concerns also apply to hardware wallets, known for their enhanced security in managing VCs [19], [20], which often have limited storage capacities ranging from a few KB to a couple of MB [21]. This highlights the need for further selective disclosure mechanisms that minimize on-device storage requirements and optimize data transmission.

**CSD-JWT.** This paper proposes a mechanism that achieves Compact and Selective Disclosure for VCs (CSD-JWT), diminishing storage and network overhead of SD-JWT. Our method gives identity owners full control over their information while minimizing the amount of data to store and share. CSD-JWT replaces the full list of salted hashes with a fixed-length value, generated by accumulating all claims into a cryptographic accumulator [22]. This ensures that the credential size remains constant regardless of the number of claims it contains. Furthermore, it hides the exact number of claims in the VC, offering stronger resilience against inference attacks.

The issuer must provide proof of inclusion for each claim to be disclosed, known as *witness*. The holder combines these witnesses with corresponding claims to generate Verifiable Presentations (VPs). The storage requirements for disclosing claims, i.e., CSD-JWT and witnesses, are lower than those for SD-JWT and salts due to the constant size of our credentials. This property also reduces the amount of data shared with the verifier, making CSD-JWT an efficient solution for constrained environments.

**Evaluation.** We implemented a prototype of CSD-JWT and conducted a series of experiments, including tests on an RFC7228 class-2 constrained device [23], to compare our method with the state-of-the-art mechanism, i.e., SD-JWT. In particular, we measured storage and computational requirement to evaluate CSD-JWT suitability for constrained devices. It is worth noting that our evaluation considers a holder maintains a single credential. However, as underscored by the EUDI Regulation [4], users are likely to use multiple VCs by 2026 in real-world scenarios [24], making the overhead reduction offered by CSD-JWT even more impactful. Experimental findings proved that our approach can achieve up to 46% memory savings for each stored credential. Experiments on the network overhead shows that VP in CSD-JWT is 93% smaller when the holder aims to disclose only 1 claim out of 100 (maximum privacy), while it still provides a 27% reduction when no privacy preservation is required and all claims are revealed.

**Contributions.** Our work has the following contributions:

- We propose CSD-JWT, a mechanism for compact and selective disclosure of claims within VCs, providing the holder with full control over their data;
- In CSD-JWT, all claims are mapped into a fixed-length value, concealing the number of claims within the VC.

This contributes to enhancing the privacy level as it minimizes the amount of information leaked;

- CSD-JWT remarkably reduces the amount of information to maintain and share, thereby lowering both storage and network overhead. This makes it suitable for constrained environments, such as hardware wallets and IoT devices;
- We implemented CSD-JWT, and conducted experiments to evaluate and compare the proposed mechanism with SD-JWT. We also made our code fully available to the research community.

**Organization.** The remainder of this paper is structured as follows: Section II provides the background on VCs and cryptographic accumulators. Section III presents the reference system and threat model. Section IV details how CSD-JWT achieves compact and selective disclosure for VCs. Section V analyzes the security of the proposed mechanism in addressing the identified threats. Section VI comprehensively evaluates CSD-JWT. Section VII reviews literature in the field. Finally, Section VIII draws our conclusions.

## II. BACKGROUND

### A. Verifiable Credential (VC)

A digital identity can be envisioned as a set of attributes or properties, which provides a snapshot of any entity, including individuals, companies, and objects. Any third party must be able to directly verify these attributes through attestations, embedded in digital credentials issued by trusted authorities. A data structure that satisfies this property is referred to as a VC. Several VC data models have been proposed in recent years, among which the World Wide Web Consortium (W3C) VCDM [25] and SD-JWT VC [26] being considered the most prominent. We propose CSD-JWT as a selective disclosure mechanism that can be seamlessly integrated into any model, offering a practical alternative to SD-JWT.

To be effective and address the challenges of centralized identity models, a verifier (e.g., a service provider) should be able to directly verify the authenticity of the claims contained, without direct interaction with the issuer. This necessitates that the issuer identity can be directly retrieved from the credential itself. Consequently, every party must have a globally unique self-generated digital identity [27]. The commonly adopted approaches are public keys and Decentralized Identifiers (DIDs) [28]. A DID is a Uniform Resource Identifier (URI) standardized by the W3C. Each DID is associated with a DID Document, encompassing publicly available information (e.g., public key) about the entity it identifies. The DID Document is typically shared in DID-based scenarios through verifiable data registries such as a blockchain [29]. Digital identifiers are crucial for identifying issuers and holders, preventing the reuse of previously collected VCs, which may allow an adversary to impersonate the identity owner. In CSD-JWT, digital identification for both holders and issuers is achieved through DIDs. However, it could be easily extended to support public key identification.

## B. Cryptographic Accumulators

Cryptographic accumulators are cryptographic primitives that securely map a set of values in a fixed-size output called accumulator value [30]. Each accumulated element can be associated with a proof of inclusion, known as a witness. This enables verifying the membership without disclosing any other elements of the set. Although several implementations have been proposed, all types of accumulators must satisfy two properties:

- 1) *One-wayness*: Given an hash function  $h_l : X_l \times Y_l \rightarrow Z_l$  this is said to be one-way if:
  - For each integer  $l$ ,  $h_l(x, y)$  is computable in polynomial time for all  $(x, y) \in X_l \times Y_l$ .
  - Additionally, no polynomial-time algorithm exists such that, for a sufficiently large  $l$ , given a pair  $(x, y) \in X_l \times Y_l$  and  $y' \in Y_l$ , it is possible to find a suitable  $x' \in X_l$  that satisfies  $h_l(x, y) = h_l(x', y')$ .
- 2) *Quasi-commutativity*: Given  $h_l : X_l \times Y_l \rightarrow Z_l$ , the following equality holds  $\forall x \in X_l$  and  $\forall y_1, y_2 \in Y_l$ :

$$h_l(h_l(x, y_1), y_2) = h_l(h_l(x, y_2), y_1).$$

We apply  $h_l$  to all the  $e_i$  to be included in the accumulator, obtaining the accumulator value  $a$ . For each  $e_i$  there exists a witness  $w_i$  such that  $a = h_l(e_i, w_i)$ , proving its inclusion within  $a$ . The one-way property ensures that given the accumulator value  $a$ , it is computationally infeasible to find a  $w_i$  that proves membership for an  $e_i$  that does not belong to the set. Moreover, using a function that meets the quasi-commutative requirement guarantees that the accumulator remains independent of the order in which values are accumulated.

Accumulators can be classified based on their support for verifying element inclusion. Specifically, those that enable verification using membership witnesses are called positive, as opposed to those supporting non-membership witnesses which are known as negative. The ones that support both types of verification methods are called universal. Various accumulators have been proposed over the years [31]–[35]. Among these, accumulators based on Bilinear Groups [22], [32], [33] are widely acknowledged for their efficiency [36], [37], especially in terms of witness size and verification efficiency for inclusion. Given these properties, CSD-JWT leverages a positive ECC-based accumulator to hide claims within VCs.

## C. Selective Disclosure

In the SSI model, an identity owner or holder  $h$  is any entity provided with a VC by a trusted issuer  $i$ . A credential comprises a set of claims  $\mathcal{C}$ , which are assertions made by the issuer about  $h$ . One of the key focuses of SSI is to empower  $h$  with full control over their personal data. This is achieved through selective disclosure, which is defined as follows:

**Definition 1 (Selective Disclosure).** *Selective disclosure is a mechanism that allows an identity owner  $h$  to disclose only a subset of claims  $C \subseteq \mathcal{C}$  from a verifiable credential to a verifier  $v$ .*

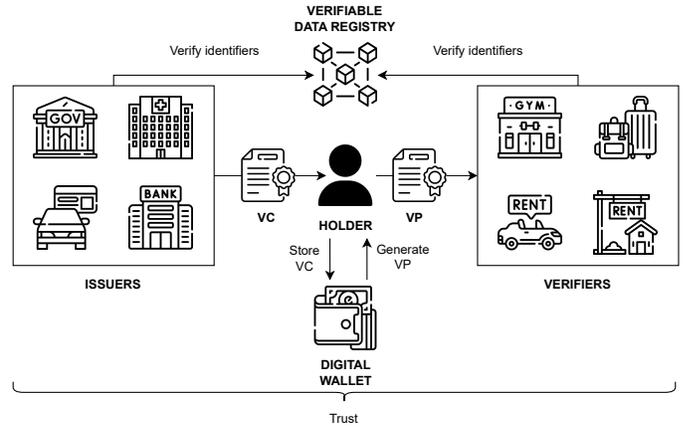


Fig. 1: SSI reference system.

A mechanism that implements selective disclosure must grant the holder fine-grained control, ensuring that they can independently determine which claims to share in response to a verifier’s request. Simultaneously, it must adhere to the principle of data minimization by limiting the disclosure to only the information strictly necessary to satisfy the verifier’s requirements, thereby reducing the risk of excessive or unintended data exposure. Selective disclosure must also preserve the verifiability of the shared information: the verifier must be able to independently assess the authenticity and integrity of the disclosed claims without needing access to the full credential.

## III. SYSTEM AND THREAT MODEL

### A. Reference System

This section describes the main actors involved in the SSI model, where all participants are digitally identified with a DID following the W3C specification. Figure 1 sketches the main entities and their interactions within the SSI paradigm.

**Holder.** In the SSI paradigm, the holder, also known as the identity owner, is any entity that requires digital identification. Each holder is identified with a DID, whose corresponding DID Document is shared through a verifiable data registry. The holder collects VCs from an issuer and stores them in a credential repository such as a file system, storage vault, or hardware wallet, which securely stores and protects access to their credentials. These credentials are presented to access different services and facilities. To preserve their privacy, holders are willing to share only the minimal amount of personal information to access a service.

**Issuer.** An issuer is a trusted entity such as a government, university, or municipality that issues VCs to identity owners. For example, an issuer may provide an individual with a VC serving as a driving license. The issuer signs credentials with its private key, making it directly verifiable to third parties. It is worth noting that a crucial assumption of the SSI model is that holders and verifiers trust the issuer.

**Verifier.** A verifier is an entity, such as a service provider, that requests credentials from identity owners to provide a specific

service. Regarding the driving license example, a car rental company may act as a verifier by requesting a digital driver’s license from a customer before allowing them to rent a vehicle. The verifier checks the authenticity and validity of the VC to ensure the customer meets the driving requirements.

**Verifiable Data Registry.** When using DIDs as digital identifiers within VCs, it is essential to have *trust anchors* that certify specific attributes associated with the identity, such as the DID Document, containing the identity owner’s public key. In DID-based scenarios, this certification is achieved through verifiable data registries. These data sources, such as blockchains, are employed to share the DID Document, ensuring its immutability and auditability.

### B. Threat Model

The security of CSD-JWT depends on the secure generation of the accumulator value and witnesses. As the issuer is a trusted entity for all the parties involved in an SSI ecosystem, we can safely assume that they have been securely computed, ensuring resistance to collisions. In this subsection, we provide an overview of the adversary perspective and outline the potential threats to our system, based on the threat model presented in [9].

**Adversary’s Goal.** The adversary has two primary objectives: to disclose personal information within a VC and to reuse previously collected credentials to impersonate the holder and gain access to services on their behalf.

**Adversary’s Capability.** We adopt the Dolev-Yao adversary model [38], which assumes that the adversary can eavesdrop on, intercept, and inject an unlimited number of messages.

**Adversary’s Knowledge.** We consider a setting where the adversary is aware of the algorithm used to issue VCs and knows the selective disclosure mechanism employed by holders to present their claims.

**Threats.** Our model focuses on the following threats affecting the holder while presenting their credentials.

- *Replay Attacks:* An adversary may present a valid credential that belongs to another identity owner. This VC could be obtained through collusion with another holder or stolen from an unsuspecting identity owner. A service provider may wrongly grant authorization believing claims, which do not represent the adversary.
- *Data Overcollection:* A service provider may collect more claims than necessary to provide a service, either with malicious intent or unintentionally. Moreover, the service provider may be able to infer additional details on the identity owner given side information such as the number of claims in the VC.
- *Compromised Communication Channel:* An adversary may intercept a credential during interactions between two legitimate parties, potentially learning its contents or modifying it.

TABLE I: CSD-JWT Notations.

Symbol	Description
$h$	Holder / Identity owner
$i$	Issuer / Trusted entity
$v$	Verifier
$vc$	Verifiable credential
$vp$	Verifiable presentation
$(pk_h, sk_h)$	Holder key pair
$(pk_i, sk_i)$	Issuer key pair to manage the accumulator
$a$	Accumulator value
$\mathcal{C}$	Set of the holder claims
$C$	Subset of the holder claims
$c_j$	Single claim of the holder
$\mathcal{W}$	Set of witnesses corresponding to $\mathcal{C}$
$W$	Subset of witnesses corresponding to $C$
$w_j$	Single witness corresponding to $c_j$

## IV. COMPACT AND SELECTIVE DISCLOSURE

This section introduces CSD-JWT, a mechanism for compact and selective disclosure of claims for VCs. CSD-JWT is implemented as a JWT that leverages the features of cryptographic accumulators. Credential holders are provided with an accumulator value and a Witness-Value Container (WVC), enabling them to selectively disclose their claims while minimizing both data exposure and storage overhead. Table I reports the adopted notation. In the remainder of this section, we first explain our use of cryptographic accumulators and then describe the main phases of our method.

### A. Preliminaries

**Accumulator.** We aggregate all claims  $\mathcal{C}$ , within a verifiable credential  $vc$  belonging to an identity owner  $h$ , into a single value  $a$ . The elements included in  $a$  are derived by applying a hash function  $h()$  to each  $c_j \in \mathcal{C}$ . Specifically, we use the cryptographically secure SHA-256 hash function, which is resistant to collisions [39]. CSD-JWT employs the positive ECC-based accumulator presented in [22], which supports batch operations for accumulating elements and generating witnesses. The accumulator value is mapped as a 256-bit point over the BN254 curve.

**Proof of Inclusion.** For each  $c_j$  included in  $a$ , there exists a witness  $w_j$ . Witnesses are pieces of cryptographic evidence that holders provide to verifiers for proving that  $c_j$  is included in  $a$ , ensuring that it has been certified by a trusted issuer.

**Functions.** In the following, we report the functions used to realize CSD-JWT:

- $par \leftarrow \text{Setup}(1^\lambda)$ : Given the security parameter  $1^\lambda$ , which determines the level of security, this function initializes and outputs the accumulator parameters  $par$ . Accumulator parameters comprise the random generator of the ECC curve  $\mathcal{G}$ , and a key pair  $(pk_i, sk_i)$ .
- $a \leftarrow \text{AccumulatorSetup}(par)$ : This function receives accumulator parameters  $par$  and setups the accumulator, generating the initial accumulator value  $a$ .
- $a \leftarrow \text{AccumulateClaims}(a, \mathcal{C}, sk_i)$ : This function takes the set of claims  $\mathcal{C}$ , the private key  $sk_i$ , and maps  $c_j \in \mathcal{C}$

---

**Algorithm 1: Verifiable Credential Issuance**

---

**Input:**  $par, \mathcal{C}, sk_t$ **Output:**  $vc$ 

```
1  $a \leftarrow \text{AccumulatorSetup}(par)$ ;  
2  $pk_i \leftarrow par.pk$ ;  
3  $sk_i \leftarrow par.sk$ ;  
4  $a \leftarrow \text{AccumulateClaims}(a, \mathcal{C}, sk_i)$ ;  
5  $\mathcal{W} \leftarrow \text{ComputeWitnesses}(a, \mathcal{C}, sk_i)$ ;  
6  $wvc \leftarrow \emptyset$ ;  
7 foreach  $(c_j, w_j) \in (\mathcal{C}, \mathcal{W})$  do  
8    $wvc \leftarrow wvc \cup \{(c_j, w_j)\}$ ;  
9  $vc \leftarrow \{a, wvc\}$ ;
```

---

onto the accumulator elliptic curve, aggregating them into a single accumulator value  $a$ .

- $\mathcal{W} \leftarrow \text{ComputeWitnesses}(a, \mathcal{C}, sk_i)$ : Given the accumulator value  $a$ , the set of claims  $\mathcal{C}$ , the private key  $sk_i$ , this function generates the set  $\mathcal{W}$ , where each  $w_j \in \mathcal{W}$  corresponds to a  $c_j \in \mathcal{C}$ .
- $0, 1 \leftarrow \text{VerifyWitness}(a, c_j, w_j, pk_i)$ : This function verifies whether the claim  $c_j$  was accumulated in  $a$ . This verification requires both the witness  $w_j$  corresponding to  $c_j$  and the accumulator public key  $pk_i$ .
- $vp \leftarrow \text{GenerateVerifiablePresentation}(a, \mathcal{C}, \mathcal{W}, n, sk_h)$ : The identity owner  $h$  uses their private key  $sk_h$  to sign the accumulator value  $a$ , a subset of claims  $\mathcal{C}$ , along with the corresponding witnesses  $\mathcal{W}$ , and a nonce  $n$  to prevent replay attacks, producing a verifiable presentation  $vp$ .
- $0, 1 \leftarrow \text{VerifyPresentation}(vp)$ : The verifier retrieves the public keys of  $pk_h$  and  $pk_i$  through their respective DID using a `Resolve()` function, which allows obtaining corresponding DID Documents. The verifier checks the presentation ownership through `Verify(vp, pk_h)`. Finally by with the `VerifyWitness()` function, the verifier confirms whether the provided claims are included in the accumulator value  $a$  contained in  $vp$ .

### B. Protocol

This section describes how CSD-JWT enables compact and selective disclosure of claims within VCs using a positive ECC-based accumulator [22]. An identity owner  $h$ , also called holder, can selectively disclose a subset of claims  $\mathcal{C}$  among those included in a  $vc$  issued by a trusted entity  $i$ , also known as the issuer. Beyond credentials,  $i$  provides holders with the plain text of their claims  $c_j \in \mathcal{C}$ , along with the corresponding witnesses  $w_j \in \mathcal{W}$ . This information is used by  $h$  to selectively disclose claims, empowering the holder with maximum control over their data meeting the SSI requirements.

**Setup.** The issuer securely generates the security parameter  $1^\lambda$  and provides it to the `Setup()` function, producing the accumulator parameters  $par$ . This  $par$  comprises the random generator of the ECC curve  $\mathcal{G}$ , and a key pair  $(pk_i, sk_i)$ . The private key  $sk_i$  is necessary to accumulate claims to produce the accumulator value and to generate the witnesses,

while  $pk_i$  is used to verify their membership. This public key must be shared through a verifiable data registry such as blockchain, ensuring its authenticity and integrity. The issuer  $i$  uses the `AccumulatorSetup()` function to initialize the accumulator obtaining the initial accumulator value  $a$ . This setup phase occurs only once, as the generated keys can support the management of multiple credentials.

**Issuance of Verifiable Credential.** The protocol begins with an identity owner  $h$  requesting a  $vc$  from a trusted entity  $i$  (i.e., the issuer), which certifies a set of claims  $\mathcal{C} = \{c_1, \dots, c_n\}$ . Each claim is a key-value pair, composed of the name identifying a property and its value.

In CSD-JWT, instead of listing hashes for every  $\mathcal{C}$ , as for SD-JWT. CSD-JWT contains an accumulator value  $a$ , computed by accumulating all  $c_j \in \mathcal{C}$ , claims also include the holder and issuer identifier, respectively  $DID_h$  and  $DID_i$ . The use of a unique identifier ensures that two distinct  $vc$  containing the same  $\mathcal{C}$  do not result in the same accumulator value  $a$  and set of witnesses  $\mathcal{W}$ . The verifier uses  $pk_i$ , along with witness  $W$ , to verify whether  $\mathcal{C}$  is included in  $a$ . Algorithm 1 shows how VCs are issued in CSD-JWT.

The issuer executes the `AccumulateClaims()` function, providing the initial accumulator value  $a$ ,  $\mathcal{C}$ , and  $sk_i$ , updating  $a$ . It is worth noting that, instead of accumulating  $c_j$ , we add  $h(c_j)$  to the accumulator to map it as a scalar value in the curve. Thus, it is highly unlikely that two distinct claims  $c_j$  and  $c_k$  produce the same digest, also ensuring that each witness corresponds to a unique claim. While not explicitly mentioned, in practice the issued  $vc$  also contains all the necessary information that usually accompanies  $vc$  such as the identifiers for  $i$  and  $h$  (i.e., credential structure or expiry date).

Furthermore,  $i$  utilizes the `AccumulateWitnesses()` function operating in batch mode, giving as input  $a$ ,  $\mathcal{C}$ , and  $sk_i$ , and receiving as output  $\mathcal{W}$ , the set of witnesses corresponding to the claims included in the issued credential. In addition to  $vc$ ,  $h$  is also provided with a WVC, containing a list of claim-witness pairs necessary for selectively disclosing claims and generating VPs.

$$WVC = [(w_1, c_1), (w_2, c_2), \dots, (w_n, c_n)]. \quad (1)$$

**Generation of Verifiable Presentations.** By holding  $a$  and the  $WVC$ ,  $h$  may reveal all claims or selectively disclose a subset  $\mathcal{C} \subseteq \mathcal{C}$  to any verifier  $v$ , such as service providers. To do this,  $h$  executes the `GenerateVerifiablePresentation()` function with the accumulator value  $a$ , the claims to disclose  $\mathcal{C}$ , the corresponding witnesses  $\mathcal{W} \subseteq \mathcal{W}$ , and a nonce  $n$ , generating a verifiable presentation  $vp$  signed with their private key  $sk_h$ . This presentation is shared as a JWT. Specifically, for each  $c_j$  to be disclosed,  $vp$  contains a claim-witness pair  $(c_j, w_j)$ . The nonce  $n$  is used to prevent replay attacks, where an adversary might intercept  $vp$  and reuse it to spoof  $h$ .

**Verification of Verifiable Presentations.** The verifier  $v$  collects the  $vp$  as JWT, including the accumulator value  $a$ , and the witness-value pairs  $(w_j, c_j) \in WVC$  corresponding to

---

**Algorithm 2:** Verifiable Presentation Verification

---

**Input:**  $vp$ **Output:** 0 or 1

```
1  $pk_i \leftarrow \text{Resolve}(vp.DID_i);$ 
2  $pk_h \leftarrow \text{Resolve}(vp.DID_h);$ 
3  $status_{vp} \leftarrow \text{Verify}(vp, pk_h);$ 
4 if  $status_{vp} = 1$  then
5   foreach  $(c_j, w_j) \in vp$  do
6      $res \leftarrow \text{VerifyWitness}(a, c_j, w_j, pk_i);$ 
7     if  $res = 0$  then
8       return 0;
9 return 1;
```

---

the disclosed claims. Algorithm 2 shows how presentation are verified in CSD-JWT. To verify the claims presented by  $h$ ,  $v$  must be provided with the issuer’s and holder’s identifiers, previously included in  $a$ , which resolve to their respective DID Documents, containing their public keys, respectively,  $pk_i$  and  $pk_h$ . The issuer’s DID Document is typically retrieved from a verifiable data registry, whereas the holder’s DID Document is usually provided directly by the holder.

Thus,  $v$  executes the `VerifyPresentation()` function to verify the authenticity of claims included in the presentation, and to assess ownership over the credential. After verifying the holder’s signature through  $pk_h$ , the verifier proceeds by asserting that the disclosed claims  $C$  are included in the accumulator value  $a$  presented. Then, the verifier then executes the `VerifyWitness()` function with  $pk_i$  for each disclosed claims  $c_j \in C$  and their corresponding witnesses  $w_j \in \mathcal{W}$ . Both the holder’s and the issuer’s identifiers are claims that must be validated in this process. As done by  $i$  when computing  $a$ , before being combined with  $w_j$  to prove inclusion in  $a$ , each  $c_j$  is hashed with  $h()$  using the same algorithm employed by  $i$ . If the combination of all the  $h(c_j)$  and  $w_j$  matches  $a$ , the set of disclosed  $C$  is authentic, and  $v$  can trust the information shared.

### C. Discussion

CSD-JWT enables the generation of VCs, requiring minimal storage, regardless of the number of hidden claims within the credential. This key feature makes CSD-JWT ideal for use in constrained scenarios where identity holders may have limited storage and computing capabilities, e.g., when using a hardware wallet for managing VCs. In the remainder of this subsection, we provide further considerations on this key property.

**Storage Requirement.** For each claim in the VC, SD-JWT requires storing a 256-bit digest and a salt whose minimum length is 128 bits. In contrast, by leveraging the ECC-based accumulator proposed in [22] with a  $\text{BN}_{254}$  curve, CSD-JWT only requires storing a single 256-bit accumulator value and, for each claim to be included in the VC, a 256-bit witness. As the number of claims increases, our solution

consumes less storage compared to SD-JWT. In Section VI, we comprehensively demonstrate that the reduction in storage space is about  $(N - 2) \times 256$  bits, where  $N$  represents the number of claims included in the VC.

If the holder could perform runtime generation of  $W$ , we could further reduce the storage requirements of CSD-JWT. However, with the current state of cryptographic accumulators, the holder cannot generate  $W$  while keeping the accumulated value at a fixed length [40]. This limitation is fundamentally tied to trapdoor-based accumulators, as enabling the holder to generate witnesses would require sharing the private key  $sk_i$ , allowing them to craft valid  $W$  for claims  $c_j \in C$  that have not been certified by any trusted party  $i$ . Therefore, to achieve runtime generation of constant-size  $\mathcal{W}$ , a trapdoor-less accumulator with fixed-length witnesses is needed, but such a construct does not currently exist.

**Computing Requirement.** CSD-JWT does not require intensive computations, making it suitable for constrained scenarios, such as hardware wallets or IoT devices, where computing capabilities may be limited. Using the accumulator, CSD-JWT does not impose any overhead on the holder side, as all the operations involving direct computation over this cryptographic structure (i.e., accumulating claims and generating corresponding witnesses) are performed on the issuer side.

On the other hand, generating VPs, as seen in SD-JWT and other approaches, must performed on the holder side since they are required to prove ownership of the disclosed claims. However, in CSD-JWT, the holder performs minimal computation, being only responsible for constructing and signing the JWT used for presentation, which contains pairs  $(c_j, w_j)$  to disclose claims. Thus, the most resource-intensive operation for the holder is signing the JWT with their private key  $sk_h$ , which is the minimum requirement also for SD-JWT.

## V. SECURITY ANALYSIS

This section examines the resilience CSD-JWT against the threats identified in the threat model described in Section III

### A. Replay Attacks

Only the legitimate holder in control of a private key should be able to generate a valid  $vp$ . Without such guarantees, adversaries could impersonate credential holders by replaying their  $vp$  or exploiting leaked data. CSD-JWT enforces this requirement by binding the presentation of claims to the holder’s DID. This ensures that credentials cannot be reused by third parties, even in cases of collusion or credential theft. We assume the unforgeability of asymmetric cryptographic mechanisms used by both the issuer and the holder, and the binding of DIDs to key pairs.

**Definition 2 (Replay Attack Resilience).** *A selective disclosure scheme is resilient against replay attacks if, for any probabilistic polynomial-time (PPT) adversary  $\mathcal{A}$  who does not possess the holder’s private key  $sk_h$ , the probability that  $\mathcal{A}$  can produce a  $vp$  that is accepted by a verifier is negligible.*

Let us consider an adversary  $\mathcal{A}$  who obtains a valid credential bound to a legit holder  $h$  by collusion or through theft. The adversary attempts to generate a valid presentation to impersonate  $h$  to a verifier. Each credential includes the holder identifier  $DID_h$ . The adversary  $\mathcal{A}$  cannot produce a valid witness corresponding to their identifier  $DID_{\mathcal{A}}$  as this would require the issuer’s private key  $sk_i$ . Moreover, in CSD-JWT, each presentation  $vp$  includes a cryptographic proof of possession of  $sk_h$ , in the form of a digital signature over a  $vp$  containing a verifier-provided nonce  $n$ . Without  $sk_h$ , the adversary cannot generate a valid signature, due to the assumed unforgeability of the holder’s signature scheme. Even if  $h$  colludes with  $\mathcal{A}$  and provides a valid  $vp$ , the verifier’s use of a fresh nonce  $n$  in each authentication session ensures that previously issued proofs cannot be replayed. Thus,  $\mathcal{A}$  cannot reuse any presentation proof without interaction with  $h$ .

To further mitigate risks from these attacks, additional defenses should be enforced at the service provider level. These include implementing multi-factor authentication [41], monitoring usage patterns to detect suspicious activities [42], and using device attestation techniques to ensure that credentials are presented only from authorized devices [43].

### B. Data Overcollection

Identity owners should provide only minimal information to verifiers such as service providers. Excessive collection of claims from a service provider could increase the risks for the holder, independently of its honesty. Malicious service providers may exploit data for financial gain or profiling purposes [44]. However, as seen for centralized models, honest servers could expose users to data breaches [45]. CSD-JWT overcomes this challenge by enabling  $h$  to selectively disclose only the claims necessary to access a service, along with the corresponding witnesses. Thus, the service provider can verify the authenticity and integrity of the shared information, without seeing the full content of the credential.

**Definition 3 (Data Minimization).** *A selective disclosure scheme satisfies data minimization if, during a verification session, the verifier  $v$  learns only (i) the subset of claims  $C \subseteq \mathcal{C}$  intentionally disclosed by the holder  $h$ ; and (ii) auxiliary information that is independent of the remaining claims  $\mathcal{C} \setminus C$  and their number.*

During a credential presentation, the holder  $h$  transmits: the disclosed claims  $C$ , cryptographic witnesses  $W$ , and an accumulator value  $a$  committing to the full set  $\mathcal{C}$  without revealing any additional information. The proof system used in CSD-JWT ensures that the witness  $W$  is both sound (i.e., the claim truly belongs to the credential) and reveals nothing about  $\mathcal{C} \setminus C$ . The commitment  $a$  is constructed using a cryptographic accumulator, which is collision-resistant and hiding and ensures that it is computationally infeasible to determine the number or content of accumulated values.

This contrasts with schemes such as SD-JWT, where structural metadata (e.g., position or number of undisclosed claims)

may be revealed during verification, potentially enabling side-channel inference attacks. For instance, consider a scenario where an individual holds a credential issued by a healthcare provider containing 20 claims related to their health condition. The holder wants to share only the health center and the last check-update with a fitness center to qualify for a health-based discount. By providing the entire credential for verification,  $v$  can observe that many claims are not being disclosed. This observation can lead the verifier to speculate about the sensitive nature of the omitted information, even if those claims were not relevant to the service.

CSD-JWT is resilient by design against this class of attacks because beyond the disclosed claims  $C \subseteq \mathcal{C}$ , the only additional information revealed are  $a$  and  $W$ . This information does not leak anything about the full set of claims, specifically how many of them are contained in it.

### C. Compromised Communication Channel

To prevent  $\mathcal{A}$  from passively accessing credential content, secure communication among parties must be established. This can be effectively achieved by employing asymmetric encryption, utilizing the public keys stored in the DID Document of participants. Additionally, tampering with the accumulator value  $a$  or  $WVC$  would be ineffective, as any modification would require a respective witness, impossible to generate without  $sk_i$ . Any altered or newly generated accumulator value by the adversary  $\mathcal{A}$  would not authenticate against the public key  $pk_i$  since the adversary must know  $sk_i$  to compute a valid  $a$ .

## VI. PERFORMANCE EVALUATION

In this section, we present a comprehensive evaluation of CSD-JWT. We performed a series of experiments to assess the impact of the proposed mechanism on the primary actors within the SSI ecosystem. Specifically, the following experiments were conducted:

- **Credential Issuance:** Evaluated the overhead for issuers to issue VCs using CSD-JWT;
- **Credential Storage:** Measured the storage requirements for a credential, highlighting CSD-JWT feature of facilitating selective disclosure with minimal storage needs.
- **Generating and Verifying Verifiable Presentations:** Assessed the latency and network overhead for presenting and verifying VPs.

Experiments were conducted by comparing CSD-JWT to SD-JWT and run on a 14th Gen Intel(R) Core(TM) i9-14900k that features 8 p-cores and 16 e-cores, and 48GB of RAM. For holder-side operations, we utilized the Nordic nRF52840 [46], an RFC7228 class-2 constrained device [23]. The nRF52840 is equipped with a 32-bit ARM Cortex-M4 processor, 1 MB of Flash memory, and 256 KB of RAM. We compared CSD-JWT to SD-JWT as it represents the current state-of-the-art solution and remains the only fully implemented and widely recognized solution available in the field at the time of this work. The experimental findings demonstrate that our mechanism is a valuable solution to enable selective disclosure

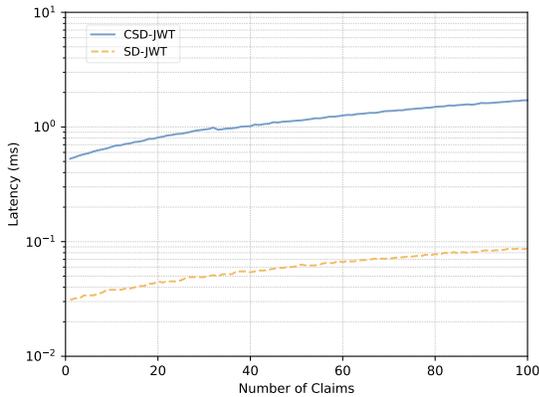


Fig. 2: Issuer overhead for generating VCs.

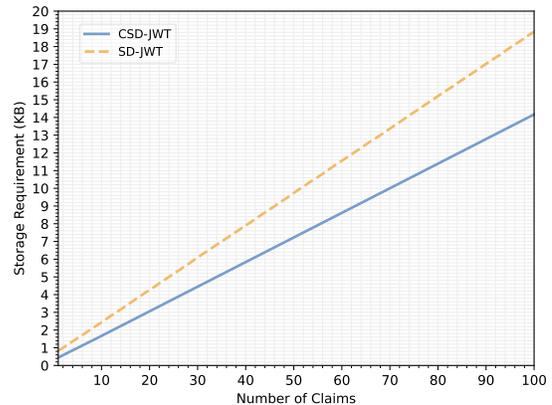


Fig. 3: Storage requirements (KB) for VCs.

while maximizing holder privacy. For fairness, all comparisons between CSD-JWT and SD-JWT were conducted using the same claims within the credentials. All experiments were repeated 1,000 times, and the results were averaged.

### A. Implementation Setup

We implemented a prototype of CSD-JWT<sup>1</sup> and evaluated its performance while delivering the core functionalities. To effectively build the SD-JWT mechanism, we relied mainly on two libraries: `josekit`<sup>2</sup>, an openssl-based library that provides ease of access to cryptographic signatures, and `serde_json`<sup>3</sup>, a library that provides API to serialize and deserialize Rust objects to a JSON format. For generating accumulator values and the witnesses associated with claims, we relied on `vb-accumulator`<sup>4</sup>, developed by DockNetwork, which meets the requirements of [22]. For the underlying ECC, we selected the BN254 curve, which offers the best trade-off between security and parameter length, as accumulator values and witnesses require 256 bits. To evaluate the feasibility of CSD-JWT and SD-JWT on constrained holders, we developed a framework<sup>5</sup> in C using the nRF Connect SDK and `Parson`<sup>6</sup>, a lightweight library for managing JSON data structures. Our credentials can include a variable number of claims, each formatted as `claim_key:claim_value`. For our evaluation, we consider VCs with up to 100 claims, reflecting variations across application domains [47]. For instance, a VC representing an identification document typically contains around 15 claims. In contrast, in healthcare scenarios, the number of claims can vary significantly, ranging from approximately 10 claims, as in the case of the healthcare passport established by many countries for COVID-19 [48], to over 100 in detailed medical reports [49]. Finally, for transparency and reproducibility, we open-sourced the proposed implementations.

<sup>1</sup>Code will be made available upon acceptance

<sup>2</sup><https://crates.io/crates/josekit>

<sup>3</sup>[https://crates.io/crates/serde\\_json](https://crates.io/crates/serde_json)

<sup>4</sup><https://crates.io/crates/vb-accumulator>

<sup>5</sup>Code will be made available upon acceptance

<sup>6</sup><https://github.com/kgabis/parson>

### B. Credential Issuance

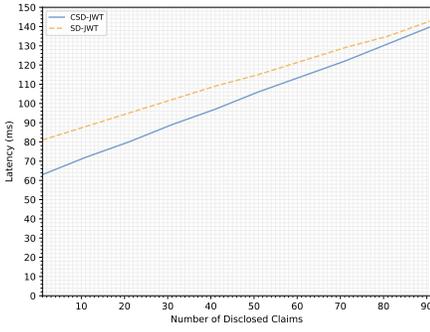
To simulate credential issuance, we evaluated the issuer overhead associated with issuing a valid VC. Specifically, we consider the latency to accumulate claims in the accumulator, embed the accumulator value within the VC, and generate witnesses. For SD-JWT, we measured its latency to generate salts with a pseudo-random number generator, compute hashes for all the claims to certify their validity, and produce an ECDSA signature.

Figure 2 compares the issuance latencies, expressed on a logarithmic scale, of the considered methods. SD-JWT achieves slightly better performance, mainly due to the CPU instructions used by the two mechanisms. Indeed, despite the great improvements in the instruction set of processors regarding finite field operations, hashing algorithms remarkably outperform them. However, it is worth noting that CSD-JWT only requires a few milliseconds even for credentials including 100 claims. Moreover, this does not represent a serious concern since, in real-world scenarios, issuers typically have powerful computing resources to manage the issuance process, significantly reducing these latencies.

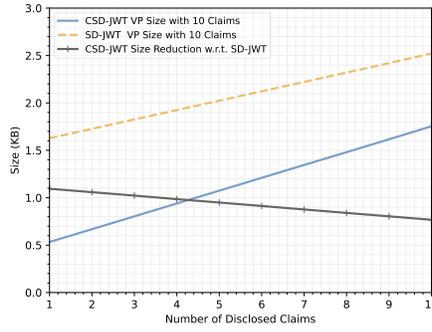
### C. Credential Storage

One of the key properties of CSD-JWT is its ability to minimize the amount of disclosed data, making it resilient against inference attacks and reducing the size of the VC. This compactness is especially crucial in scenarios where holders have restricted storage capabilities, such as hardware wallets or IoT devices. We conducted a series of experiments varying the number of claims within the VC, from 1 to 100. For each generated credential, we evaluated the overall storage requirements for the two compared methods. For our mechanism, this encompasses the storage of the accumulator value and the *WVC*, which contains the pairs of claims and witnesses. The storage requirements for SD-JWT comprise the signed list of hashes and the Salt-Value Container (SVC), which includes the pairs of claims and salts.

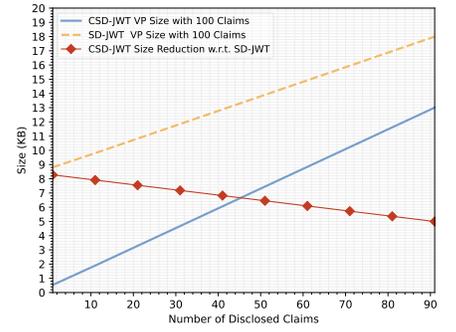
As shown in Figure 3, CSD-JWT remarkably outperforms SD-JWT even with a low number of claims in the VC. Using identical claims for both mechanisms, the primary difference



(a) VP generation (ms) for a credential with 100 claims on a constrained device.



(b) VP sizes (KB) based on a credential with 10 claims.



(c) VP sizes (KB), based on a credential with 100 claims.

Fig. 4: Comparison of CSD-JWT and SD-JWT for VP generation (a) and size (b),(c) across different scenarios.

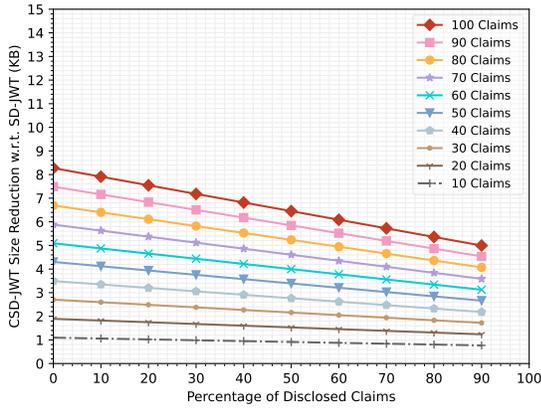


Fig. 5: VP size reduction of CSD-JWT.

lies in the size of the credentials and the storage of witnesses versus salts. The storage requirement for our method is much lower because CSD-JWT only contains the accumulator value, which is 256 bits. In contrast, SD-JWT comprises the hashes of the  $N$  claims, i.e.,  $N \times 256$  bits, making it dependent on the number of attributes. On the other hand, salts used in SD-JWT only require  $N \times 128$  bits, against the  $N \times 256$  bits for witnesses. Therefore, SD-JWT brings an overhead of  $(N - 2) \times 256$  bits for storing a VC with the same claims. As highlighted in the figure, CSD-JWT saves a significantly larger portion of memory while increasing the number of claims, achieving up to 46% storage reduction. These results demonstrate that CSD-JWT is a valuable solution for selectively disclosing credentials in constrained environments. Moreover, this saving is even more impactful as each holder is expected to maintain multiple credentials.

#### D. Generating and Verifying Verifiable Presentations

The compactness of CSD-JWT reduces the storage requirements for holding credentials and decreases the size of the VPs shared with the verifier. These experiments aim to evaluate the generation time and size of VPs, produced from VCs having a different number of claims, while varying the percentage of disclosed ones. Specifically, we generated 10 VCs each having

a number of claims equal to a multiple of 10, starting from 10 up to 100 and, for each of them, we varied the percentage of disclosed claims from 0% + 1 of the total to 90% + 1.

**Generation of Verifiable Presentations.** We conducted experiments to evaluate the latency of an RFC7228 class-2 constrained device in generating VPs using both selective disclosure techniques. CSD-JWT and SD-JWT perform similar operations to generate VPs, as both involve selecting claims to disclose and signing them. This similarity is reflected in the experimental results shown in Figure 4a, where CSD-JWT achieves slightly better performance due to the smaller size of the processed data. As anticipated, memory usage metrics for the firmware also show no significant differences, with RAM consumption remaining around 18%.

**Size of Verifiable Presentation.** Figures 4b and 4c report the results for VCs containing 10 and 100 claims, respectively. As clearly shown, CSD-JWT consistently generates smaller VPs than SD-JWT regardless of the number of disclosed claims. In Figure 4b, when the holder discloses a single claim included in a VC with 10 claims, CSD-JWT achieves a 67% reduction in the VP size, which decreases to 30% when all the claims included in the VC are disclosed. The savings in the VP size are even more significant when considering VCs including a larger number of claims. Figure 4c demonstrates that with CSD-JWT, a VP disclosing 1 claim out of 100 is 93% smaller in size than the same VP produced with SD-JWT. Even when all claims are being disclosed, CSD-JWT still provides a 27% size reduction. Figure 5 illustrates the reduction in VP size by varying the rounded percentage of disclosed claims. These results highlight that CSD-JWT is an effective method to generate VPs with reduced size and network overhead, a feature that is particularly important in scenarios where holders' devices have limited memory and network capabilities.

**Verification of Verifiable Presentations.** Finally, we evaluated the time required for a verifier to assess the authenticity of the presentations. Figure 6 shows the verification latencies for claims of different sizes using both SD-JWT and CSD-JWT. The latency for verifying a VP generated with SD-JWT is negligible (up to a few  $\mu\text{s}$ ) since the procedure

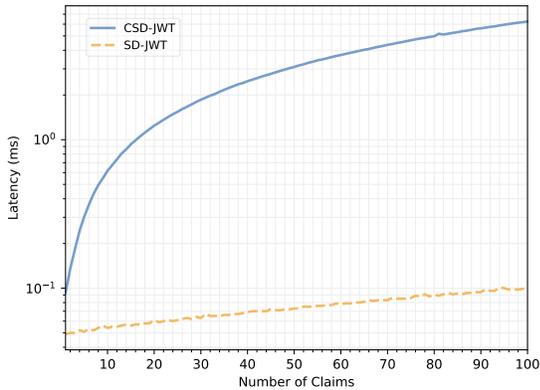


Fig. 6: VP verification latency of CSD-JWT.

involves the verification of a digital signature, a handful of hashing operations, and a comparison with a list of hashes. In CSD-JWT, verification time is affected by the underlying cryptographic accumulator. However, it remains on the order of a few milliseconds, meeting the response requirement of the Resource Animation Idle Load (RAIL) model proposed by Google<sup>7</sup>. As a result, the overhead introduced by CSD-JWT is minimal and comparable to that of SD-JWT. Furthermore, verification is performed by verifiers, which are entities typically equipped with more powerful computational resources, making the impact of such overhead even less relevant.

#### E. Discussion of Results

This section provides further consideration of the conducted experiments. It is worth noting that employing different implementations could lead to improved performance.

**Credential Issuance.** The time required to issue a CSD-JWT is higher than SD-JWT due to the complexity of the operations performed. In particular, the most demanding operation is the witness generation that, as demonstrated in [50], for  $w$  witnesses demands at least  $\Omega(w)$ . However, the issuance time is a few milliseconds and does not represent a concern as trusted authorities acting as issuers may have more powerful computing resources than those used for our experiments. Furthermore, it is worth noting that as these operations are always done on the issuer side, they do not affect at all the computing capabilities of the holder.

**Credential Storage.** In CSD-JWT, the storage requirement for the VC remains constant, regardless of the number of claims included, as it only contains the accumulator value and minimal data for management. This contrasts with SD-JWT, where credentials contain a salted hash for each claim. Although salts are shorter than the witnesses, CSD-JWT offers a size reduction ratio of 27 – 93% for VCs with 100 claims, with even greater savings for credentials containing more fields. In constrained environments, resources are shared across different functions. For instance, it is essential that the memory securely stores VCs and firmware, making even KB-level reductions valuable. Additionally, minimizing resource

usage improves energy efficiency, a critical consideration in resource-limited settings. Finally, while our evaluation focuses on the overhead reduction for a single VC, identity owners are projected to use multiple VCs by 2026, making the reductions offered by CSD-JWT significantly more remarkable.

**Verifiable Presentation.** The generation of VPs must be performed directly by holders. This is a logical consequence of the fact that they have to prove ownership over the disclosed information through a secret key. However, creating a VP does not introduce a particular overhead. Similarly to SD-JWT, the holder is only required to create and sign a JWT containing the claims to be disclosed. This makes both CSD-JWT and SD-JWT practical on constrained devices.

On the other hand, we observed remarkable differences in the sizes of the VPs. In SD-JWT, the credential shared during the presentation comprises the hashes of all the claims, thus the size depends on the number of data included. In contrast, CSD-JWT size is constant and only contains the accumulator value. This minimizes the information shared, significantly reducing the network overhead, and offering higher privacy.

## VII. RELATED WORK

This section reviews the primary selective disclosure mechanisms for VCs, categorizing them based on the underlying cryptographic techniques used to disclose claims. The categories include atomic credentials, hashed values, signature schemes, and Zero-Knowledge Proofs (ZKPs) [51].

### A. Atomic Credentials

Atomic credentials, or monocredential credentials, represents the simplest selective disclosure technique. A multicredential credential is partitioned into a subset of monocredential credentials. Instead of having a single VC with  $N$  claims, the holder will have  $N$  VCs, each containing a single claim. Although this mechanism is compatible with existing solutions that do not yet support selective disclosure, it does not scale with credentials holding many data fields. Managing  $N$  distinct credentials increases storage requirements, as each credential must include its metadata, cryptographic signatures, and auxiliary data. This also affects verification, as verifiers must individually validate each credential, raising computational and time costs. Furthermore, this approach incurs remarkable network overhead, as transmitting multiple credentials to disclose claims negatively impacts communication efficiency in bandwidth-constrained environments. These challenges are addressed by more efficient mechanisms, such as CSD-JWT, that do not need to generate multiple credentials to selectively disclose claims.

### B. Hashed Values

In hash-based methods, a VC includes the hashed values of each claim instead of the claim themselves. SD-JWT [13] is the state-of-the-art solution of JWT for selective disclosure. Each claim in the JWT payload is combined with a unique salt, hashed, and included in the credentials, ensuring claims are not directly exposed. The holder is given a SVC

<sup>7</sup><https://web.dev/articles/rail>

containing the plaintext claims and their corresponding salts. To disclose claims, the identity owner provides the verifier with the SD-JWT, the plain text claims they wish to disclose, and the corresponding salts. The verifier hashes each claim with its salt and checks if they match the pre-images of the hashed values in the SD-JWT. The main downside of this method is the credential size, which grows with the number of claims, impacting storage and transmission efficiency. Moreover, including hashes for all claims in the SD-JWT can make the system susceptible to inference attacks based on the credential’s structure. Similar to SD-JWT, De Salve et al. [52] propose using Hash-based Message Authentication Codes (HMACs) [53] to replace plaintext claims in VCs. The holder receives plain text claims and keys used to generate the HMACs. During the presentation, the holder discloses selected claims and keys, allowing the verifier to regenerate HMAC and validate HMAC. However, this approach shares the same limitations of SD-JWT. These drawbacks are addressed by CSD-JWT, which maintains a fixed credential size regardless of the number of included claims and prevents leakage of additional information (e.g., claim number).

Merkle trees [54] can also be used to create valid proofs for disclosed claims [55]. Each leaf node represents a claim’s hash, and parent nodes are obtained by hashing their children. Thus, knowing the root hash, a verifier can confirm a claim’s inclusion using a proof consisting of sibling hashes along the path to the root. This method ensures fast verification while keeping claims hidden but, contrary to CSD-JWT, results in variable proof lengths as the tree size grows with the number of claims in the credential. For binary trees, the path length is  $\log_2(N)$ , where  $N$  is the number of claims, potentially revealing a range for the VC’s field count.

### C. Signature Schemes

Signature schemes like Camenisch-Lysyanskaya (CL) signature [56] are widely used in Anonymous Credentials (ACs) [57], which can be seen as a special type of VCs, to ensure privacy guarantees such as anonymity or unlinkability. These schemes enable the holder to reveal specific claims while preserving anonymity. However, the applicability of ACs is limited being the authenticity of an identity or claim a primary requirement in most real-world scenarios [58].

Historically, the foundational work was done by the CL signature [56], which relies on the hard RSA assumption. Each message  $m_i$  is mapped onto a finite field using modular exponentiation on random values  $a_i, b, s, c$  to generate the signature. Given the public key  $pk = (n, a_1, \dots, a_n, b, c)$  and the signature  $(v, e, s)$  the verification is performed as follows:

$$v^e \equiv a_1^{m_1} \dots a_n^{m_n} b^s c \pmod{n} \quad (2)$$

For selective disclosure, the public key reveals  $a_i^{m_i}$  rather than  $a_i$ . Several other schemes have been proposed [59]–[62] with most of them using bilinear maps, allowing holders to derive valid signatures for subsets of signed messages. This prevents issuer-verifier collusion but introduces high

computational and storage overhead especially on the holder side, making them less suitable for constrained environments. Such concerns do not affect CSD-JWT, as the holder is only required to select and sign the claim to disclose, without performing complex cryptographic operations.

### D. Zero-Knowledge Proofs

ZPKs [63] enable a prover to convince a verifier of the truth of a statement without revealing any information beyond its validity. Thus, they are often leveraged alongside signature schemes to prove a possession claim without revealing the actual information. Specifically, research has focused on Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) [64], non-interactive methods to produce ZPKs, where the prover produces proofs by encoding problems as circuits. Schanzenbach et al. [65] proposed ZKclaim, which uses zk-SNARKs with Groth’s scheme [66] to produce succinct proofs from an issuer’s constraint system. While verifying proofs is efficient, their generation is time-consuming and requires significant storage. Lee et al. [67] proposed a more efficient commit-and-prove solution, though time and space efficiency remain limited compared to hash-based methods and CSD-JWT, where VP generation consists of selecting and signing claims to disclose.

## VIII. CONCLUSION

In this paper, we proposed CSD-JWT, a novel selective disclosure mechanism that leverages a cryptographic ECC-based accumulator to minimize the size of VCs, enabling the identity owner to reveal only strictly necessary information. We implemented CSD-JWT as an open-source solution and conducted a comprehensive evaluation against the current state-of-the-art, SD-JWT. Experimental results demonstrate that CSD-JWT generates compact VCs and VPs that ensure significant storage savings and minimize network overhead. These enhanced features make it particularly suitable for hardware wallets and IoT devices that usually have limited storage, computing, and network capabilities.

## REFERENCES

- [1] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, and M. Conti, “A Survey on Decentralized Identifiers and Verifiable Credentials,” *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2025.
- [2] European Union, “Complete guide to general data protection regulation compliance,” URL <https://gdpr.eu/>, accessed on July 2024.
- [3] State of California Department of Justice, “California consumer privacy act (ccpa),” URL <https://oag.ca.gov/privacy/ccpa>, accessed on July 2024.
- [4] European Union, “Regulation (EU) 2024/1183 of the European Parliament and of the Council of 5 June 2024 on European digital identity wallets,” 2024, accessed on October 2024. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2024/1183/oj>
- [5] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, “A survey on essential components of a self-sovereign identity,” *Computer Science Review*, vol. 30, pp. 80–86, 2018.
- [6] R. Jerković, “Revision of the eIDAS Regulation – European Digital Identity (EUid),” in *A Europe Fit for the Digital Age*. European Parliament, Jun. 2024. [Online]. Available: <https://www.europarl.europa.eu/legislative-train/carriage/eid/report?sid=8201>
- [7] B. Podgorelec, L. Alber, and T. Zefferer, “What is a (Digital) Identity Wallet? A Systematic Literature Review,” in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022, pp. 809–818.

- [8] GovTech Singapore, “National Digital Identity,” 2025, accessed on May 2025. [Online]. Available: <https://www.tech.gov.sg/our-digital-government-efforts/digital-identity/>
- [9] E. Krul, H.-y. Paik, S. Ruj, and S. S. Kanhere, “SoK: Trusting Self-Sovereign Identity,” *Proceedings on Privacy Enhancing Technologies*, 2024.
- [10] S. Khan, F. Luo, Z. Zhang, F. Ullah, F. Amin, S. F. Qadri, M. B. B. Heyat, R. Ruby, L. Wang, S. Ullah, M. Li, V. C. M. Leung, and K. Wu, “A Survey on X.509 Public-Key Infrastructure, Certificate Revocation, and Their Modern Implementation on Blockchain and Ledger Technologies,” *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2529–2568, 2023.
- [11] “Eudi wallet: Architecture and reference framework,” <https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/1.1.0/arf/>, accessed on January 2025.
- [12] M. Jones, J. Bradley, and N. Sakimura, “Rfc 7519: Json web token (jwt),” 2015.
- [13] F. D. Y. K. and C. B., “Selective Disclosure for JWTs (SD-JWT),” in *Selective Disclosure for JWTs (SD-JWT)*. IETF, 2023.
- [14] S. Mehnaz, S. V. Dibbo, R. De Viti, E. Kabir, B. B. Brandenburg, S. Mangard, N. Li, E. Bertino, M. Backes, E. De Cristofaro *et al.*, “Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models,” in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4579–4596.
- [15] A. Praveena and S. Smys, “Prevention of inference attacks for private information in social networking sites,” in *2017 International Conference on Inventive Systems and Control (ICISC)*, 2017, pp. 1–7.
- [16] C. Mazzocca, A. Acar, S. Uluagac, and R. Montanari, “EVOKE: Efficient revocation of verifiable credentials in IoT networks,” in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1279–1295. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity24/presentation/mazzocca>
- [17] N. Fotiou, V. A. Siris, G. C. Polyzos, Y. Kortessniemi, and D. Lagutin, “Capabilities-based access control for IoT devices using Verifiable Credentials,” in *2022 IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 222–228.
- [18] P. N. Mahalle, G. Shinde, and P. M. Shafi, “Rethinking decentralised identifiers and verifiable credentials for the Internet of Things,” *Internet of things, smart computing and technology: A roadmap ahead*, pp. 361–374, 2020.
- [19] P. Das, S. Faust, and J. Loss, “A Formal Treatment of Deterministic Wallets,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 651–668. [Online]. Available: <https://doi.org/10.1145/3319535.3354236>
- [20] E. Zaghoul, T. Li, M. W. Mutka, and J. Ren, “Bitcoin and Blockchain: Security and Privacy,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10288–10313, 2020.
- [21] L. H. Wallets, “Choose your Ledger device,” URL <https://support.ledger.com/article/360015259693-zd>, accessed on September 2024.
- [22] G. Vitto and A. Biryukov, “Dynamic Universal Accumulator with Batch Update over Bilinear Groups,” in *Topics in Cryptology – CT-RSA 2022*, S. D. Galbraith, Ed. Cham: Springer International Publishing, 2022, pp. 395–426.
- [23] C. Bormann, M. Ersue, and A. Keranen, “RFC 7228: Terminology for Constrained-Node Networks,” <https://datatracker.ietf.org/doc/html/rfc7228>, 2014, accessed on January 2025.
- [24] Gartner, “Gartner Predicts at Least 500 Million Smartphone Users Will Be Using a Digital Identity Wallet by 2026,” URL <https://tinyurl.com/mrz5v362>, Sep. 2024, accessed on January 2025.
- [25] “Verifiable Credentials Data Model 2.0,” 2025. [Online]. Available: <https://www.w3.org/TR/vc-data-model-2.0/>
- [26] “Draft SD-JWT-VC,” 2024. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/>
- [27] J. Ernstberger, J. Lauinger, F. Elsheimy, L. Zhou, S. Steinhorst, R. Canetti, A. Miller, A. Gervais, and D. Song, “SoK: Data Sovereignty,” in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 2023, pp. 122–143.
- [28] W3 Recommendation, “Decentralized Identifiers (DIDs) v1.0,” URL <https://www.w3.org/TR/did-core/>, 2022, accessed on January 2024.
- [29] J. García-Rodríguez, R. Torres Moreno, J. Bernal Bernabé, and A. Skarmeta, “Towards a standardized model for privacy-preserving Verifiable Credentials,” in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES ’21. New York, NY, USA: Association for Computing Machinery, 2021.
- [30] J. Benaloh and M. de Mare, “One-way accumulators: A decentralized alternative to digital signatures,” in *Advances in Cryptology – EUROCRYPT ’93*, T. Hellese, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 274–285.
- [31] J. Camenisch and A. Lysyanskaya, “Dynamic accumulators and application to efficient revocation of anonymous credentials,” in *Advances in Cryptology – CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 61–76.
- [32] L. Nguyen, “Accumulators from bilinear pairings and applications,” in *Topics in Cryptology – CT-RSA 2005*, A. Menezes, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 275–292.
- [33] M. H. Au, P. P. Tsang, W. Susilo, and Y. Mu, “Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems,” in *Topics in Cryptology – CT-RSA 2009*, M. Fischlin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 295–308.
- [34] C. Papamanthou, E. Shi, R. Tamassia, and K. Yi, “Streaming authenticated data structures,” in *Advances in Cryptology – EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 353–370.
- [35] D. Boneh and H. Corrigan-Gibbs, “Bivariate polynomials modulo composites and their applications,” in *Advances in Cryptology – ASIACRYPT 2014*, P. Sarkar and T. Iwata, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 42–62.
- [36] L. Wang, Y. Tian, and D. Zhang, “Toward Cross-Domain Dynamic Accumulator Authentication Based on Blockchain in Internet of Things,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2858–2867, 2022.
- [37] A. Kumar, P. Lafourcade, and C. Lauradoux, “Performances of cryptographic accumulators,” in *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 366–369.
- [38] Dolev, D. and Yao, A., “On the security of public key protocols,” *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.
- [39] H. Gilbert and H. Handschuh, “Security Analysis of SHA-256 and Sisters,” in *Selected Areas in Cryptography*, M. Matsui and R. J. Zuccherato, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 175–193.
- [40] E. Ghosh, O. Ohrimenko, D. Papadopoulos, R. Tamassia, and N. Triandopoulos, “Zero-knowledge accumulators and set algebra,” in *Advances in Cryptology – ASIACRYPT 2016*, J. H. Cheon and T. Takagi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 67–100.
- [41] S. Kim, H.-J. Mun, and S. Hong, “Multi-Factor Authentication with Randomly Selected Authentication Methods with DID on a Random Terminal,” *Applied Sciences*, vol. 12, no. 5, 2022.
- [42] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller, “CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability,” in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1348–1366.
- [43] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, “SEDA: Scalable Embedded Device Attestation,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 964–975.
- [44] Wikipedia, “Facebook–Cambridge Analytica data scandal,” URL [https://en.wikipedia.org/wiki/Facebook-Cambridge\\_Analytica\\_data\\_scandal](https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal), accessed on July 2024.
- [45] H. Saleem and M. Naveed, “Sok: Anatomy of data breaches,” *Proceedings on Privacy Enhancing Technologies*, 2020.
- [46] N. Semiconductor, “nrf52840 - multiprotocol bluetooth 5.3 soc,” 2025, accessed on January 2025. [Online]. Available: <https://www.nordicsemi.com/Products/nRF52840>
- [47] European Commission, “The many use cases of the eu digital identity wallet,” 2025, accessed on May 2025. [Online]. Available: <https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/The+many+use+cases+of+the+EU+Digital+Identity+Wallet>
- [48] M. Eisenstadt, M. Ramachandran, N. Chowdhury, A. Third, and J. Domingue, “COVID-19 Antibody Test/Vaccination Certification: There’s an App for That,” *IEEE Open Journal of Engineering in Medicine and Biology*, vol. 1, pp. 148–155, 2020.

- [49] J. Thomason, "Big tech, big data and the new world of digital health," *Global Health Journal*, vol. 5, no. 4, pp. 165–168, 2021, special issue on Intelligent Medicine Leads the New Development of Human Health.
- [50] P. Camacho and A. Hevia, "On the impossibility of batch update for cryptographic accumulators," in *Progress in Cryptology–LATINCRYPT 2010: First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, proceedings 1*. Springer, 2010, pp. 178–188.
- [51] ETSI, "Electronic Signatures and Trust Infrastructures (ESI); Analysis of selective disclosure and zero-knowledge proofs applied to Electronic Attestation of Attributes," URL [https://www.etsi.org/deliver/etsi\\_tr/119400\\_119499/119476/01.02.01\\_60/tr\\_119476v010201p.pdf](https://www.etsi.org/deliver/etsi_tr/119400_119499/119476/01.02.01_60/tr_119476v010201p.pdf), accessed on September 2024.
- [52] A. De Salve, A. Lisi, P. Mori, and L. Ricci, "Selective Disclosure in Self-Sovereign Identity based on Hashed Values," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, 2022, pp. 1–8.
- [53] J. M. Turner, "The keyed-hash message authentication code (hmac)," *Federal Information Processing Standards Publication*, vol. 198, no. 1, pp. 1–13, 2008.
- [54] R. Merkle, "A certified digital signature," vol. 435, 08 1989, pp. 218–238.
- [55] O. Steele and M. Prorock, "JSON web proofs for binary Merkle trees," URL <https://w3c-ccg.github.io/Merkle-Disclosure-2021/jwp/>, 3 2021.
- [56] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–72.
- [57] S. A. Kakvi, K. M. Martin, C. Putman, and E. A. Quaglia, "SoK: anonymous credentials," in *International Conference on Research in Security Standardisation*. Springer, 2023, pp. 129–151.
- [58] "Verifiable Credentials Data Model v2.0 - Spectrum of Privacy," URL <https://www.w3.org/TR/vc-data-model-2.0/#spectrum-of-privacy>, 7 2024.
- [59] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 41–55.
- [60] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in *Security and Cryptography for Networks*, R. De Prisco and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 111–125.
- [61] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks*, S. Cimato, G. Persiano, and C. Galdi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 268–289.
- [62] Ibm, "GitHub - IBM/idemix: implementation of an anonymous identity stack for blockchain systems," URL <https://github.com/IBM/idemix>.
- [63] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, ser. STOC '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 291–304. [Online]. Available: <https://doi.org/10.1145/22145.22178>
- [64] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again." *IACR Cryptology ePrint Archive*, vol. 2011, p. 443, 01 2011.
- [65] M. Schanzenbach, T. Kilian, J. Schütte, and C. Banse, "Zkclaims: Privacy-preserving attribute-based credentials using non-interactive zero-knowledge techniques," in *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2019.
- [66] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326.
- [67] J. Lee, J. Choi, H. Oh, and J. Kim, "Privacy-preserving identity management system," *Cryptology ePrint Archive*, Paper 2021/1459, 2021, <https://eprint.iacr.org/2021/1459>. [Online]. Available: <https://eprint.iacr.org/2021/1459>