Benchmarking Selective Disclosure Mechanisms for Verifiable Credentials: A Systematic Comparison for Security and Privacy

Benchmarking Selective Disclosure Mechanisms for Verifiable Credentials: A Systematic Comparison for Security & Privacy

Alessandro Buldini, Carlo Mazzocca*, *Member, IEEE,* Rebecca Montanari, Selcuk Uluagac, *Senior Member, IEEE*

***Abstract*—In a world where digitalization is *reshaping* every aspect of our society, digital identity has become more crucial than ever to establish trust and accountability across all entities, whether human, organizational, or machine-based. Numerous initiatives are emerging worldwide, such as the United States' Mobile Driver's Licenses (mDLs) and Singapore's National Digital Identity (NDI). In May 2024, the European Union introduced Regulation 2024/1183, establishing the European Digital Identity Framework. By 2026, this initiative will provide all European citizens with a European Digital Identity Wallet (EUDIW), allowing them to access both online and offline public and private services. The EUDIW empowers individuals to retain full control over their data, allowing them to selectively disclose only the specific information necessary for each interaction. However, the current wallet design relies on Selective Disclosure for JSON Web Token (SD-JWT), which does not fully meet the privacy requirements outlined in the regulation. This paper presents a comprehensive comparison of the main selective disclosure schemes. Specifically, we identify relevant threat models, formalize associated security and privacy properties, and assess the extent to which existing techniques satisfy these properties in mitigating the identified threats. Furthermore, we introduce an open-source benchmark that evaluates selective disclosure mechanisms across key performance indicators, including computational latency, bandwidth consumption, and storage requirements.**

***Index Terms*—Self-Sovereign Identity, Digital Identity, Verifiable Credentials, Decentralized Identifiers, Privacy.**

I. Introduction

With the increasing digitalization of services, we need an electronic counterpart to paper-based documents and certifications [1]. In this context, the notion of digital identity has become increasingly important for enabling trust, accountability, and access control across a wide range of entities, including individuals, organizations, and systems [2]–[4]. This growing relevance is also reflected by the numerous initiatives emerging worldwide. United States federal agencies and industry are promoting Mobile Driver's Licenses (mDLs) [5] and Digital Travel Credentials (DTCs) [6], laying the groundwork for replacing physical identification documents as passports. In May 2024, the EU Regulation 2024/1183 [7] established a European Digital Identity Framework, aiming to enhance secure and user-controlled digital identity. A core component of this framework is the European Digital Identity Wallet (EUDIW), projected to drive a significant shift toward digital identity, with 500 million smartphone users expected to regularly use the EUDIW by 2026 [8]. Furthermore, according to Article 5a of the regulation, this wallet will ensure "... *that all natural and legal persons in the Union have secure, trusted, and seamless cross-border access to public and private services, while having full control over their data ...*". Thus, the EUDIW aims to simplify the lives of citizens and businesses by supporting a wide range of use cases, such as opening a bank account or obtaining an educational certification [9].

One of the crucial points of these initiatives is to empower identity owners with full control over their data, embracing the principles of Self-Sovereign Identity (SSI) [10], [11]. SSI allows users to control *how*, *when*, and *with whom* they share their data. In this ecosystem, each individual or entity receives digital credentials, issued by or on behalf of a public sector body responsible for an authoritative source (e.g., a municipality), that attest to specific attributes or properties of the identity owner. These credentials, also known as Verifiable Credentials (VCs) [2], are digitally signed by the issuing entity, making them cryptographically verifiable.

As opposed to paper-based documents, VCs provide individuals with stronger tamper resistance, unforgeability, and privacy guarantees, while allowing them to *selectively disclose* the information contained within a credential. For example, when a person presents a physical ID card to prove being overage, it often reveals additional personal information that is irrelevant to the specific service and unnecessarily exposed to the verifier. In contrast, VCs enable individuals to selectively disclose only the information strictly necessary to prove a given claim by generating a Verifiable Presentation (VP).

Currently, there are a limited number of standardized formats for the issuance of electronic attestations [12]. In the EUDIW, credentials are implemented through Selective Disclosure for JSON Web Token (SD-JWT) [13]. Specifically, they are issued by replacing each of the plain text claims with a corresponding digest, obtained by hashing the claim itself and a unique salt. In order to disclose information, the holder also receives the set of claim-salt pairs used to compute each of the elements included in the SD-JWT. While this mechanism can be immediately deployed and is compatible with every signature scheme [14], it does not inherently meet the privacy requirements expressed through the European regulation, suggesting a *wallet redesign* [15].

In this direction, the European Commission issued in Octo-

Manuscript received May 19, 2025; revised **X**, **X**.

* indicates the corresponding author.

Alessandro Buldini and Rebecca Montanari are with the Department of Computer Science and Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: {alessandro.buldini, rebecca.montanari}@unibo.it

Carlo Mazzocca is with the Department of Information and Electrical Engineering and Applied Mathematics, University of Salerno, 84084 Fisciano, Italy (e-mail: cmazzocca@unisa.it).

Selcuk Uluagac is with the Cyber-Physical Systems Security Lab, School of Computing and Information Science, Florida International University, 33174 Miami, Florida, United States (e-mail: suluagac@fiu.edu).

ber 2024 a €4 million call for tenders to design and develop a solution that allows users to verify their age using a dedicated application while safeguarding privacy [16]. This motivates a systematic analysis of the security and privacy aspects of existing selective disclosure mechanisms. In the literature, only a few works [17], [18] specifically focus on selective disclosure. However, these studies do not fully provide a comprehensive treatment of security and privacy dimensions for selectively disclosing claims with VCs.

This paper aims to fill this gap by analyzing the main schemes for selectively disclosing claims. Specifically, we identify the key threats affecting selective disclosure mechanisms and formalize the corresponding security and privacy properties that should be satisfied. We systematize existing solutions according to the disclosure technique employed and analyze to what extent they are resilient against the identified threats. Furthermore, due to the lack of publicly available and reproducible implementations, we developed a novel benchmarking framework that currently supports seven selective disclosure mechanisms and enables comprehensive performance evaluation. Unlike prior efforts, our framework is not only open-source but also modular and extensible by design, allowing researchers and developers to seamlessly integrate and benchmark custom selective disclosure mechanisms against established solutions.

Contributions. The main contributions of this work are summarized as follows:

- We systematize knowledge on selective disclosure schemes according to methods, security, and privacy properties, also analyzing their resilience against threats.
- Given the lack of open-source implementations, we developed a novel extensible framework that supports several selective disclosure mechanisms. We made our code available[1], to the research community.
- We extensively compared several selective disclosure schemes by evaluating their setup latency, storage requirements, and size and timing metrics for both generated VCs and VPs. This provides a valuable comparison to categorize the implemented techniques.

Organization. The remainder of this work is structured as follows: Section II introduces the SSI paradigm. Section III reviews related works in the field. Section IV identifies the main threats related to selective disclosure. Section V characterizes existing mechanisms according to the methods employed to selectively disclose claims. Section VI discusses how the reviewed solutions meet security and privacy requirements. Section VII introduces our benchmarking framework and compares different selective disclosure techniques. Section VIII concludes the paper.

II. Self-Sovereign Identity

SSI empowers identity owners with fine-grained control over their information. This represents a remarkable shift in the digital identity landscape, as users are no longer forced to relinquish data to centralized service providers, thereby reducing the potential for data breaches and misuse [19], [20].

[1]https://github.com/alebldn/sd_benchmark

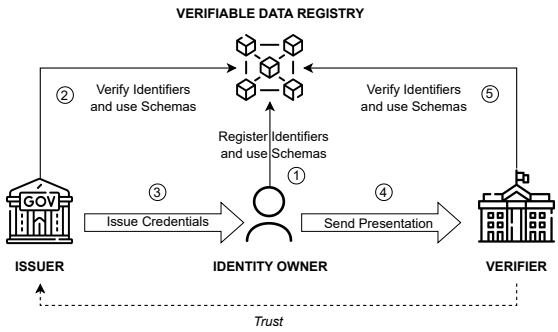


Fig. 1: Self-Sovereign Identity (SSI) [21] overview.

Trusted third parties provide VCs, digital credentials containing claims that attest to some properties of an entity, to identity owners, who usually store them in secure digital wallets. Contrary to centralized and federated models, the user is not dependent on the issuer every time they need to prove something about themselves. Individuals have direct control over *how*, *when*, and *with whom* they share specific attributes contained in their credentials. A verifier can check the authenticity of a credential without real-time communication with its issuer. SSI relies on a verifiable data registry for decentralized verification, promoting seamless interoperability and standardization. In the following, we describe the main entities of SSI, along with their role within the systems. Figure 1 overviews the roles and information flows.

Issuers. VCs are released by trusted entities, such as a public sector agency responsible for authoritative sources (e.g., a university). Each issuer has an asymmetric cryptographic key pair (sk, pk) comprised of the private key sk used to digitally sign credentials, and a public key pk, leveraged by the verifier to ensure the authenticity of the document. Issuers must meet a level of reliability and trustworthiness equivalent to qualified trust service providers.

Identity Owners. The identity owner, also referred to as the holder, is any entity in possession of one or more digital credentials. These credentials comprise a collection of attributes or claims, uniquely representing the subject, which are authenticated by an issuer. Each holder is associated with an identifier, derived from a key pair (sk, pk), that allows proving ownership over the credentials they possess. Credentials and the private key are stored in a secure repository, such as a storage vault or a hardware wallet.

Verifiers. A verifier, or relying party, determines whether to grant or not access to a service depending on the validity of credentials presented by identity owners. Verifiers are required to assess the authenticity, the integrity, and the ownership of credentials.

Verifiable Data Registry. The SSI paradigm also requires a verifiable data registry, acting as a *trust anchor* for other layers. This is usually employed to certify attributes such as the public key pk associated with the credential issuer or holder. Although there is no standard solution on what technology should be employed for this registry, it is usually implemented through Distributed Ledger Technologies (DLTs) due to their immutability and availability [22].

Information Flow. With reference to Figure 1, a typical SSI scenario comprises the following operations. We assume that the issuer is already registered as a trusted accreditation organization:

1) Identity owners register their identifiers in the SSI ecosystems.
2) Once a user requires a credential, the issuer authenticates them.
3) Upon verifying their identity and the veracity of the statements to be attested, the issuer provides the identity owner with a VC containing claims referring to the holder.
4) The identity owner generates a VP encompassing the claims necessary to access a service or resource.
5) The verifier assesses the authenticity and ownership of VP by retrieving the issuer's and holder's public key. If the claims presented are authentic, the user can access the required services.

III. Related Work

Although several works survey SSI systems [11], [23]–[25], none provide a comprehensive analysis of selective disclosure mechanisms, as they often overlook security and privacy aspects. Ernstberger [26], Krul et al. [21], and Mazzocca et al. [2] address general dimensions of SSI and introduce selective disclosure as a privacy-preserving technique. However, they do not examine specific schemes in detail, nor do they assess their resilience against security and privacy threats. Mukta et al. [27] explore data minimization approaches in blockchain-based healthcare systems, with a few limited references to selective disclosure for privacy protection. The most closely related works are [17], [18], which focus primarily on selective disclosure mechanisms.

We identify threat models that are specific to selective disclosure scenarios, going beyond the scope of existing works. Ramić et al. [17] do not address security and privacy considerations. While, Flamini et al. [18] introduce threats for the underlying cryptographic primitives rather than selective disclosure in VCs. Ernstberger [26] and Mazzocca et al. [2] discuss general SSI threats that partially overlap with selective disclosure concerns. Only Krul et al. [21] provide an in-depth analysis that explicitly covers threats affecting selective disclosure mechanisms. Additionally, the security and privacy properties of selective disclosure remain largely underformalized in existing literature, such as the nuanced distinctions between unlinkability variants (i.e., unobservability and untraceability). In this work, we explicitly formalize these properties to provide a clearer foundation for evaluating and comparing selective disclosure techniques.

We systematically categorize selective disclosure schemes into four main families; our classification also considers the extent to which each technique satisfies the identified security and privacy properties. Ramic et al. [17] classify selective disclosure mechanisms by methods, while Flamini et al. [18] categorize cryptographic approaches according to various properties, including standardization, which are not exclusively focused on security and privacy. However, neither work provides an in-depth analysis of how these approaches satisfy security and privacy properties.

Finally, a core contribution of this paper is the comprehensive evaluation of multiple selective disclosure mechanisms for VCs. All code developed is open-source and fully reproducible, providing a practical benchmark for assessing new selective disclosure schemes across a wide range of performance indicators. To date, only Flamini et al. [18] have conducted an experimental evaluation of the cryptographic mechanisms underlying VCs. However, their analysis covers a narrower set of performance metrics, and their implementations are neither publicly available nor reproducible.

IV. Threat Model

This section identifies the main threats to selective disclosure schemes and the corresponding security and privacy properties to preserve. We consider a scenario where the adversary aims to break the individual's privacy in multiple ways. Specifically, a malicious actor might try to disclose sensitive data, to track the holder by linking credentials usage over time, and to present previously collected credentials or fraudulent ones. We assume that the adversary knows how VCs are released and the mechanism employed by the identity owner to reveal a subset of claims in a credential. Following the Dolev-Yao adversary model [28], the adversary can compromise the identity owner's privacy by eavesdropping on, intercepting, and injecting an unlimited number of messages.

A. Threats

In the following, we report the main threats that affect selective disclosure schemes:

- *Data Inference*: Adversaries or verifiers may acquire additional knowledge about the holder by leveraging side information such as the nature and number of disclosed claims.
- *Cross-Presentation Correlation*: Multiple and repetitive interactions with the same verifier or across colluding verifiers may allow the correlation of the user's information over time, re-identifying the credential holder.
- *Issuer Monitoring*: An issuer may learn how the holder uses the provided credentials.
- *Issuer and Verifier Collusion*: Issuers and verifiers may collude, allowing them to leverage exchanged data to track and re-identify the user performing a transaction.
- *Presentation Forgery*: Identity owners may disclose fraudulent information.
- *Replay Attacks*: Selectively disclosed claims can be captured and reused by adversaries to spoof the legitimate credential holder, gaining access to services on their behalf.

B. Security & Privacy Properties

We formalize the definitions of the main security and privacy properties that must be satisfied by selective disclosure mechanisms. Let vc be a credential containing a set of claims $C = \{c_1, c_2, \ldots, c_n\}$, and let v be a verifier requesting a

subset of claims $C_v \subseteq C$ to fulfill a verification policy P_v in order to grant a service. We define a selective disclosure mechanism as a function $(C, \sigma) \rightarrow (C_v, \sigma_v)$ that generates a verifiable presentation π. We denote with σ an authentication and integrity verification method (e.g., a digital signature), and with $M_\pi \subseteq M$ the metadata exposed during presentation. Finally, we denote by λ the security parameter.

Definition 1 (**Minimization**). *A selective disclosure scheme satisfies the minimization property if and only if:*

$$C_v = \{C_v \subseteq C \mid P_v(C_v) = 1 \wedge \arg\min_{C_v \subseteq C} |C_v|\}.$$

Beyond claim minimization, a selective disclosure scheme should minimize metadata leakage; hence, it requires that:

$$M_\pi = \arg\min_{M' \subseteq M} |M'|$$

Definition 2 (**Predicate Disclosure**). *A selective disclosure mechanism that supports predicate disclosure allows the identity owner to prove that a claim $c \in C$ satisfies a specific condition $P(c)$ without revealing the actual value of c. Let $P : C \rightarrow \{0, 1\}$ be a predicate function that evaluates whether a claim satisfies a given condition. A predicate disclosure mechanism provides a cryptographic proof generation function PG:*

$$\vdash = \mathsf{PG}(sk, c, P),$$

such that a verifier can check without learning c.:

$$P_v(\vdash, P) = 1$$

Definition 3 (**Unlinkability**). *A selective disclosure mechanism satisfies unlinkability if, for a vc possessed by a holder h, the presentation π_1 generated for a verifier v_1 and the presentation π_2 generated for a second verifier v_2, cannot be used by the colluding verifiers to determine that h was the author of both. Formally, unlinkability holds if:*

$$\Pr[\mathsf{L}_{v_1}(\pi_1, h) = 1)] \approx \Pr[\mathsf{L}_{v_2}(\pi_2, h) = 1] \leq \mathit{negl}(\lambda),$$

where L_{v_*} is a function executed by the verifier to attest if the holder of the credential is h. It outputs 1 if h is the holder that produced the verifiable presentation, 0 otherwise.

Definition 4 (**Unobservability**). *Let π denote a verifiable presentation derived from a verifiable credential vc. A credential presentation mechanism satisfies the* unobservability *property if the issuer i cannot determine when, where, or to which verifier v a credential is presented. Formally, unobservability holds if:*

$$\Pr[\mathsf{O}_i(\pi, v, t) = 1] \leq \mathit{negl}(\lambda),$$

where O_i is an algorithm executed by the issuer attempting to infer whether a credential it issued was presented at time t to verifier v. The function outputs 1 if the issuer successfully identifies the presentation event, and 0 otherwise.

Definition 5 (**Untraceability**). *A selective disclosure mechanism satisfies untraceability if colluding issuers and verifiers cannot trace a user's credential usage across multiple interactions. Let $\pi_1, \pi_2, \ldots, \pi_n$ be presentations generated from the same credential. Untraceability holds if:*

$$\Pr[\mathsf{T}_{i,v}(\pi_1, \ldots, \pi_n) = 1] \leq \mathit{negl}(\lambda),$$

where $\mathsf{T}_{i,v}$ is a function run by colluding entities attempting to infer that the presentations originate from the same credential.

Definition 6 (**Non-Transferability**). *A selective disclosure mechanism satisfies non-transferability if presentations cannot be reused or delegated to another party. Let π be a presentation generated by the legitimate credential holder. Non-transferability holds if, for any adversary $\mathcal{A}$ with access to π, it cannot generate a valid presentation $\pi^{\mathcal{A}}$ that is accepted by a verifier without interaction with the original holder:*

$$\Pr[(P_v(\pi^{\mathcal{A}}) = 1] \leq \mathit{negl}(\lambda).$$

Definition 7 (**Unforgeability**). *A selective disclosure satisfies unforgeability if a holder h cannot create or present fraudulent credentials that have not been legitimately issued. Let $\mathsf{I}(sk_i, h)$ be the credential issuance process, where sk_i denotes the issuer's private key. Unforgeability requires that, for any adversary $\mathcal{A}$, the probability of producing, without access to sk_i, a valid presentation $\pi^{\mathcal{A}}$ containing at least one claim $c^{\mathcal{A}}$ that does not belong to any validly issued:*

$$\Pr\left[c^{\mathcal{A}} \in \pi^{\mathcal{A}} \wedge c^{\mathcal{A}} \notin \mathsf{I}(sk_i, \cdot) \mid P_v(\pi^{\mathcal{A}}) = 1\right] \leq \mathit{negl}(\lambda)$$

V. Existing Methods

This section reviews existing literature and cryptographic solutions to selectively disclose claims within VCs, categorizing them according to the method employed to reveal information. Table I summarizes the time and storage complexities of the reviewed solutions, using Big-O notation with respect to the number of claims. While many methods share similar complexity, practical performance can differ significantly due to constant factors, implementation details, and cryptographic costs, as shown in Section VII. We do not provide an in-depth analysis of Zero-Knowledge Proof (ZKP)-based mechanisms due to the broad range of available mechanisms; a comprehensive evaluation of the advantages and disadvantages of each solution would require a dedicated study.

A. Atomic Credentials

Atomic credentials represent the naive approach to achieve selective disclosure [29] by breaking down a multi-claim credential into a series of individual credentials, each containing a single claim. This approach inherently supports selective disclosure and can be easily integrated into various systems. For example, instead of presenting a full credential with multiple claims, such as a digital ID containing name, address, and age, atomic credentials allow users to disclose only one

TABLE I: Comparison of selective disclosure mechanisms, with the complexity of key operations expressed using Big-O notation.

Method	Technique	Ref.	VC Generation Efficiency	VP Generation Efficiency	VC Verification Efficiency	VP Verification Efficiency	VC Storage Efficiency	VP Storage Efficiency
Atomic Credentials	Mono-claim	[29]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Hashed Values	SD-JWT	[13]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	HMAC	[30]	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	Merkle Tree	[31]	$O(n)$	$O(n\log(n))$	$O(n)$	$O(n\log(n))$	$O(n)$	$O(n\log(n))$
Signatures	CL Signatures	[32]	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	PS Signatures	[33]	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
	BBS+ Signatures	[34]	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(n)$

of these claims (e.g., age) by sharing the specific credential that verifies it.

However, they come with substantial drawbacks regarding scalability and storage: the number of credentials a user must manage increases linearly with the number of contained claims. Moreover, every atomic credential is verified individually, which introduces delays when an identity owner presents multiple claims at once. In high-throughput systems, handling credentials individually can create performance bottlenecks, degrading response times and user experience. Atomic credentials also introduce network overhead, as each credential must be transmitted independently, making it less practical for bandwidth-constrained environments or applications that require frequent credential sharing.

B. Hashed Values

Hash-based methods leverage the one-way nature of cryptographically secure hash functions to conceal claims within credentials. To verify these claims, holders are typically required to supply supplementary information alongside the credential.

SD-JWT. The state-of-the-art solution is represented by SD-JWT [13], which allows selective disclosure on a set of claims $\mathcal{C}$. In particular, when providing a VC, an issuing authority generates a salt s_i for each claim $c_i \in \mathcal{C}$ before producing a hash $h_i = H(c_i||s_i)$ corresponding to the concatenation of a single claim and a single salt using a cryptographically secure hash function. These hashes are then included in a list, which is digitally signed by the issuing authority. Finally, the holder is given a VC including the signed list of digests and the Salt Value Container (SVC), a document that includes every c_i, s_i pair that was previously employed to compute the salted hashes.

Whenever the holder needs to prove the authenticity of a subset of the claims $C \subseteq \mathcal{C}$, they will only disclose the signed list of hashes and the subset of tuples in the SVC corresponding to the claims in C. By providing these data to the verifier, they can perform the same hashing operation to assert whether the provided values match the hashes included in the list. If so, as this list is signed with the issuer's private key, the verifier can trust the authenticity of the claims. Salting the claims before hashing them provides guarantees on producing different digests even for claims whose entropy is low (i.e., the nationality of an individual). This prevents malicious verifiers from guessing undisclosed elements of the credential by pre-computing rainbow tables.

HMAC. Similarly, De Salve et al. [30] leverage HMAC [35] as the underlying mechanism. With this method, the issuing authority produces a hash-based message authentication code h_i using a freshly generated key k_i for each of the claims c_i to be included in the credential. The list of HMACs is then signed and included in the VC along with another document, which includes tuples c_i, k_i for each claim. Similarly to the SD-JWT approach, whenever the holder of the VC must disclose a subset of claims, they will only provide the tuples c_i, k_i relative to the revealed claims, allowing the verifier to reproduce a subset of the HMACs included in the signed list.

Merkle Trees. Merkle Trees [36] can also be leveraged for selective disclosure as proposed in [31]. In this context, to authenticate statements within a credential, it is required to build a binary Merkle Tree in which each leaf node corresponds to a salted hash of a claim c_i such that $h_i = H(c_i||s_i)$, where s_i is a salt generated for each claim in the credential. Each parent node $p_{i,j}$ is derived by hashing its children $h_{p_{i,j}} = H(h_i||h_j)$. Once the tree is built, the issuer can digitally sign its root. When the identity owner needs to disclose a specific subset of claims, for every statement and its corresponding salt, they must provide a Merkle proof consisting of the path obtained by recursively hashing the node and its sibling up to the root level. To verify the authenticity of a VP, the relying party verifies the digital signature of the root and checks, for each disclosed claim, its respective path from the leaf level up to the root itself.

Tian et al. [37] present a selective disclosure mechanism that leverages Erasure Coding (EC) [38] and a Merkle Tree structure. VC is partitioned into multiple chunks, further encoded using EC to generate parity chunks. These chunks are encrypted and stored off-chain with a distributed storage provider. Meanwhile, the blockchain stores a digest, which includes the credential identifier, its hash, the Merkle root (with leaves representing the chunks and their corresponding parity data), and the maximum chunk size. The holder requests a proof object from the off-chain storage provider to reveal data. This object contains the encrypted data chunks and the Merkle Tree proof (the hashes of all sibling nodes). The recipient can verify the integrity of the disclosed data by reconstructing the Merkle root and comparing it with the one stored on-chain.

Mukta et al. [39] propose CredChain, a mechanism to implement selective disclosure through redactable signatures. CredChain signs the credential by combining the pseudo-

random function Goldreich-Goldwasser-Micali (GGM) [40] construction and the Merkle Trees. On the left side of the tree, a pseudo-random value is generated at each node from a random seed. On the right side, each leaf corresponds to an attribute of the credentials. The root tree is computed, as previously discussed, and signed with the issuer's private key. To reveal information, the holder provides the verifier with the claim to disclose, their position in the tree, and the hashes of redacted attributes. This information is used to verify whether these values are included in the root signed by the issuer.

C. Signatures

A third category of methods employs specialized signature schemes built to sign an array of messages that permit their verification even if only a subset of them is disclosed. Moreover, some of these mechanisms provide means to randomize the issued signature or completely avoid its disclosure by providing a Zero Knowledge Proof of Knowledge (ZKPOK) of the signature itself to prevent the presenter of the signed credentials from being tracked.

CL Signatures. Camenisch-Lysyanskaya (CL) signatures [32] are a foundational tool in privacy-preserving credentials based on the hard RSA assumption. The algorithm setup requires the selection of an RSA modulus $n = p \cdot q$, where both p and q are safe primes. To generate the cryptographic key pair for signing a VC containing L claims, the issuer must pick at random L scalar values $a_1, \ldots, a_L$, and two scalars b and c. The public key is then defined as $pk = (n,\, a_1,\, \ldots\, a_L,\, b,\, c)$ and the secret key is defined as $sk = p$. In order to sign a VC containing the claims $c_1 \ldots c_L \in \mathcal{C}$, the issuer selects at random two scalars e and s, and computes a value v such that:

$$v = (a_1^{c_1}\, a_2^{c_2} \ldots a_L^{c_L}\, b^s\, c)^{1/e} \pmod n \tag{1}$$

The signature is then defined as $\sigma = (e,\, s,\, v)$. Each commitment $a_i^{c_i} \pmod n$ included in the computation of v conceals the corresponding claim c_i, enabling the credential holder to present any combination of commitments and disclosed claims. For example, to reveal c_1 while keeping c_2 hidden, the identity owner presents $((a_1, c_1), a_2^{c_2})$. This presentation proves their possession of the commitment $a_2^{c_2}$ without disclosing the actual value of c_2. By employing the public key pk, the issuer can verify its validity by checking the correctness of the following equation:

$$v^e = (a_1^{c_1}\, a_2^{c_2} \ldots a_L^{c_L}\, b^s\, c) \pmod n \tag{2}$$

The paper then proposes a ZKPOK method to prove the validity of a given signature without disclosing neither the signature nor the claims signed by leveraging commitment schemes. Commitment on claims can then be opened, effectively disclosing the claim used to compute them.

Pointcheval-Sanders Signatures. In [33], Pointcheval and Sanders proposed a protocol for generating a signature over an array of messages (PS-MS). This holds the same properties of the CL signatures scheme with pairing-friendly curves of type III that are secure under the bilinear group model and random oracle model [41]. Sanders proposed a variation of the signature scheme that directly supports selective disclosure [42]. The author first defines $G_i^* \leftarrow G_i - 1_{G_i}$ as the multiplicative group i without the identity element. When producing a credential with L claims to be signed with this algorithm, an issuer needs to produce two generators $(g, \tilde{g}) \leftarrow G_1^* \times G_2^*$ and $L + 1$ random scalar values $x, y_1, \ldots, y_L \leftarrow Z_p$ to compute the asymmetric key pair. Using these values, the issuer will compute:

$$\begin{aligned} X &\leftarrow g^x, \\ Y_i &\leftarrow g^{y_i}, && \forall 1 \le i \le L, \\ \tilde{Y}_i &\leftarrow \tilde{g}^{y_i}, && \forall 1 \le i \le L, \\ Z_{i,j} &\leftarrow g^{y_i \cdot y_j}, && \forall 1 \le i \ne j \le L. \end{aligned} \tag{3}$$

The secret key sk is defined as $(x, y_i, \ldots, y_L)$ and the public key pk is defined as $(X, \{(Y_i, \tilde{Y}_i)\}_{1 \le i \le L}, \{Z_{i,j}\}_{1 \le i \ne j \le L})$. To produce a signature σ on a VC storing a set of claims $(c_1, \ldots, c_L)$, the issuer must pick at random $\tilde{\sigma}_1 \leftarrow G_2$ and compute $\tilde{\sigma}_2 \leftarrow \tilde{\sigma}_1^{x + \sum_{i=1}^{L} y_i \cdot c_i}$. The output signature is $\sigma = (1_{G_1}, 1_{G_1}, \tilde{\sigma}_1, \tilde{\sigma}_2)$. The holder of the signed VC can produce a derived signature on a subset $I \subset [1, L]$ of claims they want to disclose by generating two random scalars $r, s \leftarrow Z_p$ and computing the following elements:

$$\begin{aligned} \tilde{\sigma}_1' &\leftarrow \tilde{\sigma}_1^r, \\ \tilde{\sigma}_2' &\leftarrow \tilde{\sigma}_2^r \cdot (\tilde{\sigma}_1)^t, \\ \sigma_1' &\leftarrow g^t \prod_{j \in \overline{I}} Y_j^{m_j}, \\ \sigma_2' &\leftarrow \left(\prod_{i \in I} Y_i \right)^t \prod_{i \in I, j \in \overline{I}} Z_{i,j}^{m_j}. \end{aligned} \tag{4}$$

The so computed derived signature is defined as $\sigma_I = (\tilde{\sigma}_1', \tilde{\sigma}_2', \sigma_1', \sigma_2')$. Upon receiving a VP, the verifier can assert the validity of the derived signature is valid on a subset $I \subset [1, L]$ of claims by checking that:

$$e\left(X \cdot \sigma_1 \prod_{i \in I} Y_i^{c_i}, \tilde{\sigma_1} \right) = e(g, \tilde{\sigma}_2), \tag{5}$$

$$e\left(\sigma_1, \prod_{i \in I} \tilde{Y_i} \right) = e(\sigma_2, \tilde{g}). \tag{6}$$

If both of these checks are successful, the verifier is fully convinced of the authenticity of the received VP.

BBS+ Signatures. The BBS+ signature scheme [34], an extension of the original BBS construction [43], is based on bilinear pairings and achieves security under the q-Strong Diffie-Hellman (q-SDH) assumption. This signature scheme permits the signing of an array of claims $(c_1, \ldots, c_L) \in Z_p$. Similarly to the CL signature scheme, it leverages ZKPOK of a signature over a set of committed messages, by employing Pedersen commitments [44].

Whenever in need of generating a cryptographic key pair, an issuer picks a random $\lambda \in Z_p^*$ as its secret key SK, and computes $w = h_0^{\lambda}$, where h_0 is a public generator for G_2 and setting w as the public key pk. Upon receiving a request to

TABLE II: Classification of selective disclosure mechanisms. **Legend:** ● Supported, ◐ Partially Supported, ○ Not Supported.

Method	Technique	Ref.	Minimization	Predicate Disclosure	Unlinkability	Unobservability	Untraceability	Non-Transferability	Unforgeability
Atomic Credentials	Mono-claim	[29]	●	○	○	●	○	◐	●
Hashed Values	SD-JWT	[13]	◐	○	○	●	○	◐	●
	HMAC	[30]	◐	○	○	●	○	◐	●
	Merkle Tree	[31]	◐	○	○	●	○	◐	●
Signatures	CL Signatures	[32]	◐	◐	●	●	●	●	●
	PS Signatures	[33]	◐	○	●	●	●	●	●
	BBS+ Signatures	[34]	◐	◐	●	●	●	●	●

sign a VC, containing $(c_1, \ldots, c_L) \in Z_p^L$, the issuer chooses a pairing e and a random scalar value s. To compute the signature, it needs to select a generator $g_i \, \forall i \in [0, L+1]$ and to evaluate:

$$A = [g_0 \, g_1^s \, g_2^{c_1} \, g_3^{c_2} \cdots g_{L+1}^{c_L}]^{1/(e+\lambda)}. \tag{7}$$

The final signature computed on the set of messages $(c_1, \ldots, c_L)$ is then defined as $\sigma = (A, e, s)$. The holder of the VC, knowing the generators $g_i \, \forall i \in [0, L+1]$ employed to construct the signature, can conceal the undisclosed claims by providing a commitment $d_i = g_i^{c_i}$ and accumulating them by product. Formally, if $\overline{I}$ is the set of undisclosed claims and I is the set of disclosed ones, such a product would be $C_m = g_1^s \prod_{i \in \overline{I}} g_i^{c_i}$. Moreover, the scheme allows the production of a ZKPOK of the signature to fully prevent its exposure. By leveraging these techniques, the holder of a VC can produce a VP with a subset of the claims included in the original VC. To verify the claims, the verifier must perform a single pairing check:

$$e(A, w \, h_0^e) = e(g_0 \, g_1^s \, C_m \prod_{i \in I} g_i^{c_i}, h_0) \tag{8}$$

If the left side and the right side match, the verifier is convinced of the authenticity of the disclosed claims. BBS+ leverages discrete-logarithm ZKPs, which do not reveal any information about undisclosed claims.

D. Zero-Knowledge Proofs

First introduced in [45], Interactive ZKP systems enable a prover (i.e., the identity owner) to convince a verifier of the truth of a statement about an input x, without revealing any additional information. From the verifier's perspective, the data presented are indistinguishable from a distribution that could be efficiently generated without accessing the holder's secret. Essentially, the verifier gains confidence in the statement's validity without learning anything beyond its truth.

In [46], the authors proposed a method to convert any Interactive ZKP System into a Non-Interactive one, laying the groundwork for a new class of digital signature schemes. This concept led to significant advancements, including Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARK) [47], and Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARK) [48]. As highlighted in [15], ZKP systems in the Interactive Oracle Proof models have short and prover-efficient proofs that only rely on collision-resistant hash functions.

T1 – Takeaways on Existing Methods. Atomic credentials and hash-based approaches offer simplicity, broad compatibility with existing signature schemes, and immediate deployability. However, atomic credentials suffer from scalability limitations, particularly when the holder possesses multiple or large credentials. In contrast, methods based on selective disclosure signatures and ZKPs are more computationally demanding but enable the generation of more compact VPs. These properties make them better suited for scenarios where reducing data exposure is a priority, even at the cost of increased computational overhead during VP creation and verification.

VI. Security & Privacy Analysis of Existing Methods

In this section, we analyze the extent to which existing solutions address the security and privacy properties outlined in Section IV. Table II illustrates the degree to which each mechanism addresses these properties.

Minimization. A verifier should acquire only the lowest amount of information needed to assess whether the individual can access a given service. This prevents malicious providers from exploiting data for financial gain or profiling [49]. Given the nature of the selective disclosure properties, all the mechanisms that empower individuals to reveal only a subset of their claims meet the data minimization.

Verifiers may infer additional information from a VP, particularly the number of claims in the credential. Except for mono-claim credentials, most mechanisms reveal either partial or full claim counts. SD-JWT and its HMAC-based variant disclose a list of cryptographic commitments with a fixed size. Merkle Trees reveal a range, as proof lengths scale with $\log_2(n)$ for n claims. Signature-based methods also leak claim counts: the PS scheme reveals the cardinality of the claim set via the public key; BBS+ exposes a lower bound by requiring indices of disclosed claims; and CL signatures require both the basis elements used in commitments and the indices of disclosed claims.

To mitigate these concerns, the recent SD-JWT specification by the IETF [50] recommends including one or more decoy digests (i.e., mock claims) in the payload to obscure the actual number of disclosed claims. This is especially important for conditionally included claims: when a claim is only present under certain conditions, a decoy digest should be added when the condition is not met to avoid revealing its absence. However, the use of decoy digests can increase the size of credentials, introducing additional storage and network overhead.

Nonetheless, decoy digests alone may not fully eliminate privacy risks, especially in sensitive contexts where indirect inferences remain possible. In fact, shifting the leakage from the exact number of claims to an upper bound still allows a verifier to draw conclusions based on the structure of the credential. The presence of multiple digests, real or decoy, can suggest the existence of undisclosed information, potentially leading to unintended inferences. For instance, a VC issued by a healthcare provider may contain 20 claims detailing various health examinations. The holder wants to share only two specific pieces of information, such as the date of their last routine check-up and proof of vaccination, with a local employer offering a wellness-based insurance discount. If the employer receives the entire credential, they might infer the presence of many undisclosed claims, potentially suggesting chronic or serious health conditions. Even though these inferences remain unconfirmed, they could affect the employer's willingness to grant the discount, showing how selective disclosure can still entail privacy risks through indirect leakage.

Predicate Disclosure. Selective disclosure mechanisms with predicate disclosure capabilities limit the information revealed and reduce the risk of data inference attacks. For example, an individual may want to prove they are over 18 without disclosing their birth date. Hash-based mechanisms cannot directly support this property, as they generate presentations from the original claims without expressing them as predicates. A workaround solution would involve the issuer signing precomputed predicate claims like the assertion of an individual being overage.

Conversely, signature-based selective disclosure techniques such as CL signatures can support range proofs [51] but are inadequate for complex computations [52]. For instance, if a user must prove their financial asset level, components such as goods and debts may need to be evaluated through a complex appraisal function. In cases where the appraisal function cannot be directly represented within the selective disclosure or range-checking frameworks, solely relying on CL signatures and range-proof schemes may not suffice for the verifiable credential data model.

The natural solution to achieve predicate proof is ZKP mechanisms, which allow users to prove statements about some of the claims in the credential without disclosing underlying data. ZKPs can handle complex computations by enabling proofs over predicates, such as demonstrating that age surpasses a certain threshold. By embedding predicate checks within the proof circuit, ZKPs provide greater flexibility and privacy than other techniques.

Unlinkability. A selective disclosure mechanism that supports unlinkability must prevent verifiers from correlating data exchanged during different transactions, thus protecting against cross-presentation correlation. This means different verifiers cannot establish whether two presentations refer to the same identity owner or not. Furthermore, unless the individual is explicitly required to disclose an information-sensitive claim that can be used to identify the holder itself, this property should also hold when presenting the same VC twice to the same verifier. Mechanisms based on hashed values do not meet the unlinkability property due to the intrinsic limitation of hashes and signatures being static values. This limitation could be partially addressed by leveraging single-show [53] or k-show [54] credentials, which offer unlinkability only for a fixed number of presentations. However, this requires periodical re-issuance of credentials from the user.

Methods based on signatures for selective disclosure only disclose a set of attributes and their corresponding proof of authenticity. At the same time, by leveraging ZKPs, they do not reveal anything else about the identity owner other than the claims disclosed. From the verifier's perspective, this makes credential presentation indistinguishable in the interaction with two distinct users with identical selectively disclosed attributes. Finally, as opposed to its original version, though the variant of the PS signature scheme for selective disclosure does not support ZKPOK of the signature, it supports its randomization. This randomization generates a unique output for each VP, providing privacy benefits similar to ZKPs by preventing verifiers from linking multiple transactions to the same holder or colluding to correlate different presentations.

Unobservability. One of the fundamental principles of SSI is to empower identity owners with full control over their data. This means they can disclose claims independently, without needing to interact with any intermediary, including the trusted issuer of the credential. The only information required about the issuer is the public key, which is used to verify the authenticity of the revealed claims.

This property is satisfied by all the reviewed selective disclosure schemes as users can directly decide *when and to whom* to reveal information, with no requirement to notify any other party, including the issuer. Additionally, under the assumption of them acting honestly, verification is performed locally by relying parties through cryptographic mechanisms without querying the entity that certified the claims in the presentation.

Untraceability. Unlinkability and unobservability capture security respectively against malicious verifiers or malicious issuers, while they do not cover when both verifiers and issuers may collude. Granting untraceability means that the entity releasing credentials cannot obtain information from the entity that verifies it, such that it is possible to identify the user who is presenting the credential. Similarly to the unlinkability property, the only way for the issuer and verifier to track a user is by leveraging a fixed value that uniquely identifies it. Therefore, this property cannot be achieved neither by atomic credentials, nor by hash-based mechanisms, as it is trivial to link together disclosures and attestation through signatures and the disclosed hashes. On the other hand, signature-based protocols that use randomized signatures or ZKPs do prevent holder traceability by either avoiding the disclosure of the signature, or by dynamically changing it at every interaction.

Non-Transferability. An individual may present credentials that were either legitimately issued to another user, obtained through collusion with that user, or stolen from them. For example, a citizen may present a driver's license released to a friend or a relative. In many cases, this property is met by including an identifier of the holder within the credential

and proving ownership of a private key corresponding to that identifier. The presentation should also hold a nonce or a challenge to be resistant against replay attacks. However, including an identifier is in contrast with the unlinkability property.

Selective disclosure signature techniques address this limitation as non-transferability is achieved by proving *what you know* [21]. The holder demonstrates knowledge of a secret committed during issuance (a link secret [55]), without revealing it. The verifier checks the holder's knowledge of the committed secret and verifies that the issuer signed a commitment to that hidden secret.

Unforgeability. To be accepted, a VC must be provided by a trusted issuer. Therefore, the mechanism leveraged to implement selective disclosure must be unforgeable. Tampering with the cryptographic construct must fail verification. Additionally, the holder cannot produce a valid presentation for a claim not included in the credential. This property is usually granted by the underlying asymmetric mechanism by signing the credential with the issuer's private key using a secure signature algorithm. The holder may also leverage a ZKP of knowledge of a signature, which holds only if the issuer signed the credential.

T2 – Takeaways on Security & Privacy Properties. All reviewed selective disclosure schemes address unobservability and unforgeability. While atomic credentials and hash-based mechanisms are the simplest to implement and the most efficient to compute, they lack the necessary means to grant unlinkability and untraceability. Only mechanisms based on ZKPs or signature randomization fully satisfy these security and privacy properties. However, these advanced signature schemes incur significantly higher computational overhead compared to atomic credentials and hash-based methods. Further research is needed to optimize these schemes for practical deployment, balancing strong privacy guarantees with computational efficiency.

T3 – Takeaways on Standard Compatibility. SD-JWT is currently the standardized solution for performing selective disclosure in mDL [5], EUDI-ARF [56], and the W3C VCDM [57], owing to its simplicity of implementation and compatibility with existing identity frameworks. However, following a consensus opinion of relevant cryptographers to the European Commission [15], both EUDI-ARF and VCDM have been updated to incorporate the BBS+ signature scheme. This revision aims to enhance privacy guarantees. Specifically, unlinkability and untraceability cannot be fully achieved through SD-JWT or other hash-based mechanisms. Despite its technical merits, the broader adoption of BBS+ within the eIDAS framework [58] faces regulatory challenges, as pairing-based cryptography is not currently approved by SOG-IS [59]. Emerging proposals [60] that adapt BBS+ to elliptic curve cryptography offer a promising alternative, though they remain under review.

VII. Experimental Analysis

This section outlines our benchmarking framework for selective disclosure mechanisms. We describe the implementation setup and detail the evaluation metrics employed. Finally, we present an in-depth analysis of the experiments conducted, offering additional insights into the performance and characteristics of various selective disclosure schemes.

A. Implementation Setup

Our framework offers a valuable benchmark for evaluating selective disclosure mechanisms. The library encompasses different methods for selectively revealing information within a VC. Specifically, our benchmark evaluates the selective disclosure methods described in Section V, except for ZKPs as their implementation for selective disclosure is still in its early stage [16]. Each selective disclosure mechanism was evaluated assuming a security level of $\lambda = 128$ bits.

Libraries. Rust is a natural choice for implementing this framework, offering high efficiency, built-in security, and a rich ecosystem of cryptographic libraries. Our implementation specifically leverages the following libraries: `josekit`[2] for both the implementation of HMAC functions and for manipulating, signing, and verifying JWTs, `serde_json`[3] for serialization and de-serialization of data, `zkryptium`[4] for the management of the BBS+ and CL signature schemes, and `rs-merkle`[5] for handling Merkle Trees. Given that Hyperledger's `ursa`[6] library is now marked as deprecated, aside from SD-JWT, the only ready-to-use library for achieving selective disclosure in Rust is, to the best of our knowledge, the BBS+ implementation offered by `zkryptium`. To enable selective disclosure with the CL signature scheme, we extended the `zkryptium` library with support for multi-message disclosure and enhanced security interfaces; our code was later merged into the main library. Despite the main key functionalities for selective disclosure being widely accessible (i.e., digital signatures, message authentication codes, hashing algorithms), the mechanisms are only hand-crafted at a high level by the authors. Additionally, we are also the first to provide a Rust implementation for the Pointcheval-Sanders signature scheme modification provided by Sanders in [42] and previously mentioned in subsection V-C. All the developed code has been made available to the research community. Further details on its usage and extension are reported in Appendix A.

Core Operations. Each selective disclosure mechanism exposes functions to: setup the selective disclosure mechanism; generate a JWT-like VC from a raw credential containing the claims to be authenticated; verify the correctness of the issued VC; produce a JWT-like VP from both a previously issued VC and an array of claims to be disclosed; verify the authenticity of the so generated VP. These core

[2]https://crates.io/crates/josekit
[3]https://crates.io/crates/serde_json
[4]https://crates.io/crates/zkryptium
[5]https://crates.io/crates/rs-merkle
[6]https://crates.io/crates/ursa

functions simulate issuer–holder and holder–verifier interactions. To standardize testing, raw credentials utilized for VC generation held a variable number of claims in the format `claim_key:claim_value`.

Experiments were executed on credentials whose number of claims would vary between 1 and 100; on the other hand, tests on VPs were executed on VCs for which the number of claims is a multiple of 10: for each VC that satisfies this condition, we kept the number of claims fixed and tested our metrics by varying the amount of disclosed claims from 10% to 100% with step 10%.

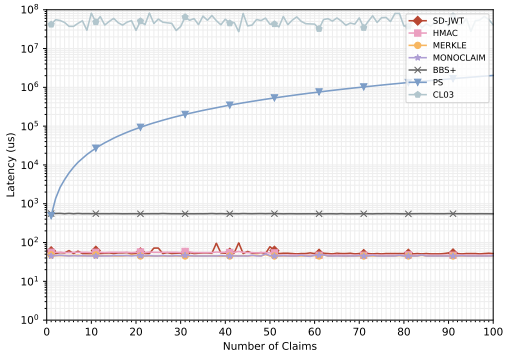
Testbed. Benchmarking was performed on a system equipped with an Intel i9-14900K processor (5.6 GHz base frequency) and 48 GB of DDR5 RAM (7000 MHz). The selective disclosure mechanisms evaluated in this section are `SD-JWT` [13], its variation built upon HMAC [30] (`HMAC`), Merkle Trees for selective disclosure [31] (`MERKLE`), atomic credentials [29] (`MONOCLAIM`), the BBS+ signature scheme [34] (`BBS+`), the variant of the Pointcheval-Sanders signature scheme proposed in [42] (`PS`), and the Camenisch-Lysyanskaya signature scheme based on [32] (`CL03`). All tests were executed sequentially to avoid any bias from random process scheduling that could impact the results. Moreover, for better accuracy on experimental findings, each test was repeated 100 times and the results were averaged over this interval.

B. Evaluation Metrics

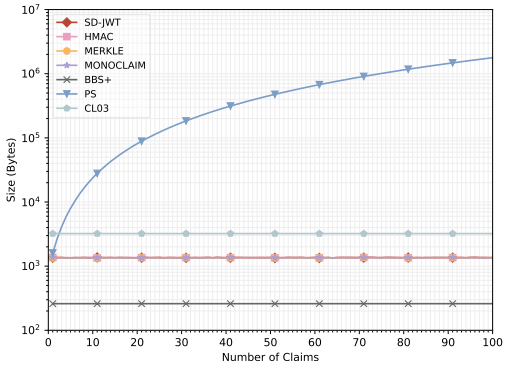
The main goal of our framework is to evaluate the performance of selective disclosure mechanisms. A selective disclosure mechanism is considered *efficient* if the time to generate VCs, verify their authenticity, and storage requirements is not significantly impacted by the number of the included claims. The same considerations also hold for VPs; the only difference being that they should be almost independent of the number of disclosed claims. To this purpose, we compare selective disclosure schemes on the following metrics:

- **Generation Efficiency**: Time required for the credential issuer to generate a VC, a VP, and the cryptographic material necessary for the mechanism to function properly.
- **Verification Efficiency**: Time required for the verification of the correctness of the issued VC (holder side) and the presented VP (verifier side).
- **Storage Requirement**: Size of the JWT that encodes the VC stored in the holder's hardware and size of the cryptographic keys.
- **Network Overhead**: Size of the JWT that encodes the VP shared with the verifier.

While our theoretical analysis (summarized in Table I) employs asymptotic notation to characterize each mechanism with respect to the number of claims, the experimental results presented in this section report concrete performance in terms of execution time (μs) and size (bytes), providing a more comprehensive picture. All figures adopt a logarithmic scale to emphasize differences in magnitude across the evaluated methods.



(a) Time required for generating cryptographic keys (μs).



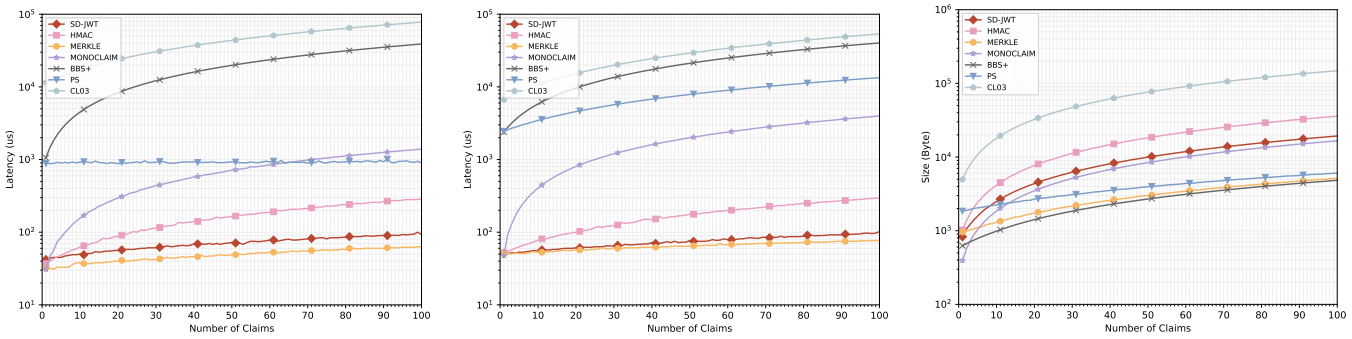
(b) Size of cryptographic keys (Bytes).

Fig. 2: Cryptographic key metrics.

C. Key Generation

Performance metrics such as key generation time and storage overhead are critical when evaluating selective disclosure mechanisms in SSI systems. Although the generation of cryptographic keys used to sign VCs is typically performed by issuers with powerful hardware, the size of these keys remains a crucial factor for other participants in the system. In the SSI ecosystem, verifiable data registries facilitate the distribution of public keys for all involved entities. However, identity holders and verifiers must still retrieve and store these keys to authenticate VCs and VPs. Since verification may occur on resource-constrained devices (e.g., hardware wallets [61]), selective disclosure mechanisms must be designed with these limitations in mind. This section evaluates both the key generation time and the associated storage requirements of the cryptographic keys used in these operations.

Generation. Cryptographic material is typically generated by the issuer, who is assumed to have access to powerful hardware. As a result, key generation is generally not a primary performance concern, particularly given that asymmetric keys are created or rotated with a relatively low frequency, depending on factors such as regulatory requirements and certificate expiration policies. Consequently, unless key generation time is exceptionally high, none of the evaluated selective disclosure mechanisms should be dismissed based solely on this metric.



(a) VC generation latency (μs). (b) VC verification latency (μs). (c) VC storage requirement (Bytes).

Fig. 3: Latency and storage requirements varying the numbers of claims within a VC.

Figure 2a illustrates the time required to generate a key pair for each evaluated selective disclosure mechanism. All the algorithms are far below the threshold of acceptance, with the only exception being the Camenisch-Lysyanskaya parameter generation algorithm that requires $\approx 45s$ to be generated for a key length of 3072 bits due to the requirement of safe primes. While relatively high, this is still a fair timing. For most schemes, key generation time remains constant, independent of the number of claims. The only exception to this behavior is the variant of the PS scheme [42], which exhibits a quadratic increase in key generation time with respect to the number of claims in VC since its public key includes a matrix whose horizontal and vertical dimensions grows with the number of claims to be signed for a given credential.

Size. In contrast to key generation time, the size of cryptographic keys can have a more significant impact on the choice of a selective disclosure mechanism. While the verifier is typically expected to operate on a relatively powerful machine, this assumption does not always hold for the holder. In many cases, the device storing the credential may have severely limited resources, making key size a critical consideration.

Figure 2b shows that all the evaluated selective disclosure mechanisms require key pairs of fixed size, with the sole exception of the PS signature scheme. In fact, as previously noted, the key size in the PS scheme for selective disclosure grows quadratically with the number of claims in the VC. Although this behavior is uncommon, it does not necessarily represent a drawback in the SSI context. VCs issued by legal entities often contain a fixed number of claims, allowing the same PS key pair to be reused across multiple instances of the same document type.

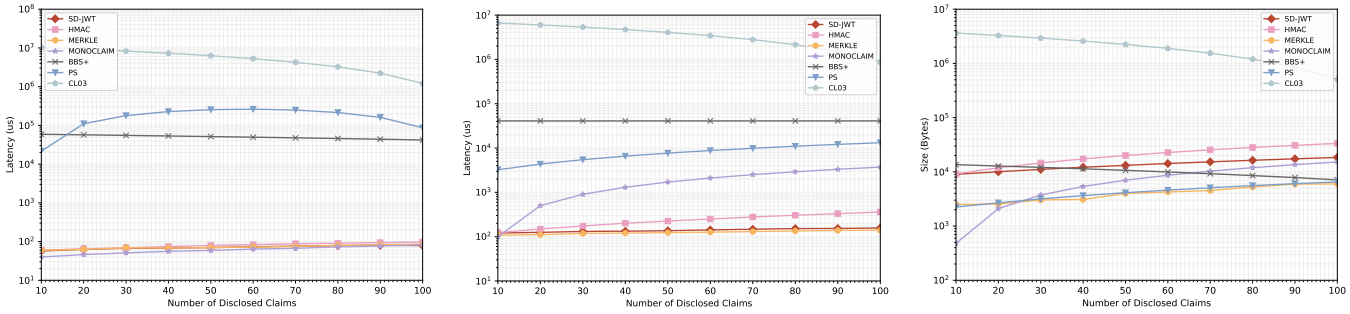
D. Verifiable Credentials

This set of experiments aims to evaluate the issuer overhead in generating VCs across different selective disclosure mechanisms. To simulate different application scenarios, we simulated the issuance of 100 VCs, each including a unique number of claims ranging from 1 to 100. These documents are encoded as JWTs and include all the parameters required for their verification, aside from the cryptographic key material. Figures 3a, 3b, and 3c report the results.

Issuance. For issuance overhead, we consider the amount of time to perform all the operations needed to compute a VC. For example, regarding SD-JWT, this includes computing hashes for each claim and generating a document that maintains the association between the plain claim and the hash, for Merkle Trees requires the generation of a salt for each claim and the construction of the binary tree, while for BBS+ it encompasses the generation of a valid BBS+ signature over a block of messages.

As foreseeable, Figure 3a outlines that hash-based mechanisms are the most efficient since modern CPUs are optimized to perform hashing algorithms. The HMAC-based method comes right after: message authentication codes require multiple iterations of hashing algorithms. As expected, mono-claim is less effective since the issuer is required to generate multiple digital signatures. Finally, the BBS+ and CL signature schemes demand higher computation time as they leverage several modular exponentiations to produce the signature. Interestingly, while the PS variant presents a relatively high offset for producing a signature for a single claim, this computation remains constant on a logarithmic scale. This is justified as it requires a single modular exponentiation for producing a signature while necessitating just scalar multiplication and a scalar sum for each claim included in the VC, in contrast to the standard behavior of requiring a modular exponentiation for every claim, which is performed by the other evaluated signature schemes.

Verification. As the identity owner may be willing to verify the authenticity of the provided credential, a key metric to be evaluated is the time required for its verification (see Figure 3b). The plots do not highlight outliers in the expected behavior. Each algorithm requires a verification time that is linear with respect to the number of claims included in the VC. The algorithms differ in the starting point and slope of the line. As expected, the most efficient ones are hash-based methods as they require running highly optimized hashing operations. Signature-based methods, on the other hand, require higher computation costs due to the underlying operations on finite fields. Specifically, the variant of the PS signature scheme for selective disclosure exhibits a similar behavior to what was evaluated in the generation of a VC, starting from a high offset. The number of operations is not fixed but greatly reduced,



(a) VP generation latency (μs). (b) VP verification latency (μs). (c) VP network overhead (Bytes).

Fig. 4: Latency and overhead for VP varying the percentage of disclosed claims, given a VC containing 100 claims.

exhibiting a smaller scaling factor as the number of claims grows with respect to the other methods based on signature schemes.

Storage Requirement. The size of the stored VC grows linearly with the number of claims within the credential. However, as outlined in Figure 3c, this increase is less sharpened for Merkle Trees, PS, and BBS+ approaches. In the Merkle Tree, the amount of information needed to prove membership within the credential, i.e., the path from leaves to the root, grows with the number of claims. BBS+ is the most effective in terms of storage savings as it only requires a single signature to be included in the VC. Similarly, the PS signature scheme has higher requirements due to the larger size of the signature included. The growth factor for these two signature schemes solely depends on the number of claims included in the VC. On the other hand, the signature scheme CL does necessitate the inclusion of the bases necessary to produce the proof of knowledge at a later stage, therefore having a higher initial storage requirement, but displaying a growth factor that is similar to other methods.

E. Verifiable Presentation

The generation, verification, and size efficiency of VPs depend on the information the individuals aim to disclose from the original issued VC. Therefore, we have evaluated our metrics over 10 different VCs, each holding claims that are multiples of 10 up to 100 claims. Each graph presents a variation in disclosed claims by percentage points, varying from 10% up to 100% of the claims included in the VC. Figure 4 reports the results for VPs generated from VCs containing 100 claims, which is the upper bound considered in our evaluation.

Generation. Figure 4a shows a clear division between VP generation timings between signature-based selective disclosure solutions and the rest, explicitly mono-claim credentials and hash-based algorithms. This result is explainable as the latter two groups almost (except Merkle Trees) only require filtering out an array of elements in the VC to produce a VP. SD-JWT filters out the salt-claim values matching disclosed claims. The HMAC-based algorithm filters out key-claim pairs to permit the computation of the cryptographic construct. Mono-claim credentials are only inserted in the VP with the signatures relative to the disclosed claims. Merkle Tree has a slightly higher computation cost than SD-JWT due to the computation of proofs for each disclosed claim. However, as the required operations are simply hashing operations, the overhead remains particularly low.

On the other hand, signature-based algorithms do require the holder to produce either a ZKP or a derived signature from the original one, therefore forcing the execution of resource-intensive computations. The most interesting evidence produced in this graph is the non-monotonic trend of the PS curve. A key factor for understanding this behavior lies in the nature of the structure of the variant of the PS signature scheme's public key and its function to derive the randomized signature. Indeed, a signature is derived as follows: $\sigma_I = (\sigma'_1, \sigma'_2, \tilde{\sigma}'_1, \tilde{\sigma}'_2)$. As mentioned above, the computation of $\tilde{\sigma}'_2$ involves a product of two sequences: one iterating over the set of undisclosed claims, and the other over the disclosed ones. Consequently, the computational load peaks when approximately 50% of the claims are disclosed. This behavior can be analytically explained by examining the derivative of the function $x(1-x)$, which models the interaction between disclosed and undisclosed claims. This is consistent with the shape of the underlying cost function, whose derivative indicates a maximum when the two sets are balanced, i.e., when approximately half of the claims are disclosed.

Verification. To verify the authenticity of VPs, verifiers are required to check that the credential was released by a trusted issuer and that the presented claims are embedded within that VC. Figure 4b shows that, similarly to VP issuance timings, there is a clear separation between methods belonging to different categories. Hash-based methods are proven to be once again the most effective due to the acceleration in the computation of hash functions thanks to modern CPU architectures. Interestingly, the two methods that produce ZKPs are the only mechanisms whose verification time decreases as the number of disclosed claims grows. This makes sense because as more information is revealed, the less concealed information needs to be proven in zero-knowledge. It is worth noting that the same argument applies to both VP issuance and verification latencies, and VP size.

Network Overhead. Similarly to VCs, the size of VPs grows with the number of claims presented to the verifier. However,

while the size of VC impacts devices' storage, that of VPs introduces overhead on the communication channel. Indeed, once a VP is generated, the holder is not required to maintain it as reusing the same VPs over time leads to security concerns such as replay attacks. According to the results shown in Figure 4c, the Merkle Tree and PS solutions consistently introduce the lowest overhead across the experiment. The only exception is atomic credentials, offering a particularly low overhead for VPs with up to 25 disclosed claims. Similarly to what was previously observed for verification time, the only two methods that decrease over time as the number of disclosed claims increases are those that employ ZKPs for disclosing claims.

T4 – Takeaways on Experimental Results. The choice of a selective disclosure mechanism is closely tied to the specific deployment context. In privacy-sensitive applications, such as cross-border credential verification, schemes like BBS+ may be a preferred option due to its strong guarantees on unlinkability and untraceability, along with minimal storage overhead. These properties are particularly valuable in scenarios where user anonymity and data minimization are critical. However, these benefits come at the cost of higher computational complexity, particularly during VP generation, which may impact responsiveness and scalability in time-sensitive applications. In contrast, constrained environments such as IoT deployments or mobile wallets, where devices may have limited processing power, memory, or battery life, require lightweight solutions. In such cases, hash-based mechanisms like Merkle Trees offer a more practical alternative: they enable fast verification and low memory usage, making them well-suited for devices with limited capabilities. While these approaches may sacrifice some privacy guarantees compared to more advanced schemes, they strike a balance between performance and acceptable privacy levels, especially in use cases where the device itself acts as the identity holder and privacy requirements may be relaxed [62], [63].

VIII. Conclusion

This paper presents a systematic study of selective disclosure mechanisms that enable users to reveal only a subset of claims within VCs following SSI principles. We review existing techniques for selective information disclosure and assess how well these schemes satisfy the security and privacy requirements set by emerging regulations, including the EU-DIW framework. Our analysis also delves into the resilience of these mechanisms against threats targeting the security and privacy of credentials. To facilitate real-world application and research, we introduced an open-source framework that supports the effective use of various selective disclosure mechanisms, providing a practical resource for integrating VCs across diverse application domains and for benchmarking and evaluating new selective disclosure schemes. We believe this framework offers significant utility to developers, researchers, and regulators working to enhance the privacy and security of digital identities.

Acknowledgment

This work was funded by the SERICS project (PE00000014) under the MUR National Recovery and Resilience Plan program by the European Union – NextGenerationEU. This research was carried out in collaboration with the U.S. National Science Foundation through the Intergovernmental Personnel Act Independent Research & Development Program. The views expressed are those of the authors and do not necessarily reflect those of the funding agencies.

References

[1] J. Vrana and R. Singh, "Digitization, digitalization, and digital transformation," *Handbook of nondestructive evaluation 4.0*, pp. 1–17, 2021.

[2] 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*, 2025.

[3] Y. Wang, S. Guo, Y. Pan, Z. Su, F. Chen, T. H. Luan, P. Li, J. Kang, and D. Niyato, "Internet of Agents: Fundamentals, Applications, and Challenges," *arXiv preprint arXiv:2505.07176*, 2025.

[4] T. South, S. Marro, T. Hardjono, R. Mahari, C. D. Whitney, D. Greenwood, A. Chan, and A. Pentland, "Authenticated Delegation and Authorized AI Agents," *arXiv preprint arXiv:2501.09674*, 2025.

[5] Transportation Security Administration, "Real id and mobile driver's licenses (mdls)," 2024, accessed: 2025-05-19. [Online]. Available: https://www.tsa.gov/real-id/real-id-mobile-drivers-license-mdls

[6] A.-T. Radutoiu, A. Bassit, R. Veldhuis, and C. Busch, "A study on the next generation of digital travel credentials," in *2024 International Conference of the Biometrics Special Interest Group (BIOSIG)*, 2024, pp. 1–6.

[7] "Regulation (eu) 2024/1183 of the european parliament and of the council of 5 june 2024 on european digital identity wallets," 2024, accessed: 2024-10-13. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/1183/oj

[8] Gartner, "Gartner predicts at least 500 million smartphone users will be using a digital identity wallet by 2026," Sep. 2024, accessed: 2024-10-13. [Online]. Available: https://tinyurl.com/mrz5v362

[9] European Commission, "The Many Use Cases of the EU Digital Identity Wallet," 2024, accessed: 2024-10-29. [Online]. Available: https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/The+many+use+cases+of+the+EU+Digital+Identity+Wallet/?pk_source=linkedin&pk_medium=social_media_organic&pk_campaign=EUDIW_WeeklyUseCase_MDL_07MAY2024

[10] K. L. Tan, C.-H. Chi, and K.-Y. Lam, "Survey on Digital Sovereignty and Identity: From Digitization to Digitalization," *ACM Comput. Surv.*, vol. 56, no. 3, Oct. 2023.

[11] 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.

[12] European Commission, "Architecture and Reference Framework for the European Digital Identity Wallet," 2024, accessed: 2024-10-29. [Online]. Available: https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/blob/main/docs/arf.md

[13] F. D, Y. K, and C. B, "Selective Disclosure for JWTs (SD-JWT)," in *Selective Disclosure for JWTs (SD-JWT)*. IETF, 2024. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-oauth-selective-disclosure-jwt-07.html

[14] Gataca, "Ssi essentials: Which selective disclosure protocol will succeed?" 2022, accessed: 2024-11-04. [Online]. Available: https://gataca.io/blog/ssi-essentials-which-selective-disclosure-protocol-will-succeed/

[15] C. Baum, O. Blazy, J. Camenisch, J.-H. Hoepman, E. Lee, A. Lehmann, A. Lysyanskaya, R. Mayrhofer, H. Montgomery, N. K. Nguyen *et al.*, "Cryptographers' Feedback on the EU Digital Identity's ARF," *Tech. Rep.*, 2024.

[16] European Commission, "Development, Consultancy and Support for an Age Verification Solution ," 2024, accessed: 2024-10-21. [Online]. Available: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/tender-details/ae950883-112f-4139-989e-1c8d794bb77a-CN

[17] Šeila Bećirović Ramić, E. Cogo, I. Prazina, E. Cogo, M. Turkanović, R. T. Mulahasanović, and S. Mrdović, "Selective disclosure in digital credentials: A review," *ICT Express*, 2024.

[18] A. Flamini, G. Sciarretta, M. Scuro, A. Sharif, A. Tomasi, and S. Ranise, "On cryptographic mechanisms for the selective disclosure of verifiable credentials," *Journal of Information Security and Applications*, vol. 83, p. 103789, 2024.

[19] H. Saleem and M. Naveed, "Sok: Anatomy of data breaches," *Proceedings on Privacy Enhancing Technologies*, 2020.

[20] P. Mayer, Y. Zou, F. Schaub, and A. J. Aviv, ""Now I'm a bit angry:" Individuals' Awareness, Perception, and Responses to Data Breaches that Affected Them," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 393–410. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/mayer

[21] E. Krul, H.-y. Paik, S. Ruj, and S. S. Kanhere, "SoK: Trusting Self-Sovereign Identity," *Proceedings on Privacy Enhancing Technologies*, 2024.

[22] 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.

[23] R. Soltani, U. T. Nguyen, and A. An, "A Survey of Self-Sovereign Identity Ecosystem," *Security and Communication Networks*, vol. 2021, p. 8873429, Jul 2021.

[24] A. Satybaldy, M. S. Ferdous, and M. Nowostawski, "A Taxonomy of Challenges for Self-Sovereign Identity Systems," *IEEE Access*, vol. 12, pp. 16 151–16 177, 2024.

[25] F. Schardong and R. Custódio, "Self-sovereign identity: A systematic review, mapping and taxonomy," *Sensors*, vol. 22, no. 15, 2022.

[26] 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.

[27] R. Mukta, H. young Paik, Q. Lu, and S. S. Kanhere, "A survey of data minimisation techniques in blockchain-based healthcare," *Computer Networks*, vol. 205, p. 108766, 2022.

[28] 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.

[29] "Verifiable Credentials Implementation Guidelines 1.0," 9 2019. [Online]. Available: https://www.w3.org/TR/vc-imp-guide/#selective-disclosure

[30] 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.

[31] O. Steele and M. Prorock, "JSon web proofs for binary Merkle trees," 3 2021. [Online]. Available: https://w3c-ccg.github.io/Merkle-Disclosure-2021/jwp/

[32] 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.

[33] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings*. Springer, 2016, pp. 111–126.

[34] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5*. Springer, 2006, pp. 111–125.

[35] "RFC 2104: HMAC: Keyed-Hashing for Message Authentication." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2104

[36] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," *Ruhr-University Bochum, Tech. Rep*, vol. 12, p. 19, 2008.

[37] R. Tian, L. Kong, B. Zhang, X. Li, and Q. Li, "Authenticated Selective Disclosure of Credentials in Hybrid-Storage Blockchain," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 330–337.

[38] T. Zhou and C. Tian, "Fast erasure coding for data storage: A comprehensive study of the acceleration techniques," *ACM Trans. Storage*, vol. 16, no. 1, Mar. 2020.

[39] R. Mukta, J. Martens, H.-y. Paik, Q. Lu, and S. S. Kanhere, "Blockchain-Based Verifiable Credential Sharing with Selective Disclosure," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 959–966.

[40] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, p. 792–807, Aug. 1986.

[41] J. Camenisch, M. Drijvers, A. Lehmann, G. Neven, and P. Towa, "Short threshold dynamic group signatures," in *International conference on security and cryptography for networks*. Springer, 2020, pp. 401–423.

[42] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *Public-Key Cryptography – PKC 2020*, A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds. Cham: Springer International Publishing, 2020, pp. 628–656.

[43] 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.

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

[45] 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

[46] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[47] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 326–349. [Online]. Available: https://doi.org/10.1145/2090236.2090263

[48] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Paper 2018/046, 2018. [Online]. Available: https://eprint.iacr.org/2018/046

[49] Wikipedia, "Facebook–Cambridge Analytica data scandal," URL https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal, accessed: 2024-10-29.

[50] D. Fett, K. Yasuda, and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)," Internet Engineering Task Force, Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-22, May 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/

[51] Q. Stokkink, G. Ishmaev, D. Epema, and J. Pouwelse, "A truly self-sovereign identity system," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 2021, pp. 1–8.

[52] J. Lee, J. Choi, H. Oh, and J. Kim, "Privacy-preserving identity management system," *Cryptology ePrint Archive*, 2021.

[53] S. Brands, "A technical overview of digital credentials," 2002.

[54] M. Layouni and H. Vangheluwe, "Anonymous k-show credentials," in *Public Key Infrastructure: 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June 28-30, 2007. Proceedings 4*. Springer, 2007, pp. 181–192.

[55] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3*. Springer, 2003, pp. 268–289.

[56] Eu-Digital-Identity-Wallet, "eudi-doc-standards-and-technical-specifications/docs/technical-specifications/ts4-zkp.md at main · eu-digital-identity-wallet/eudi-doc-standards-and-technical-specifications," 5 2025. [Online]. Available: https://github.com/eu-digital-identity-wallet/eudi-doc-standards-and-technical-specifications/blob/main/docs/technical-specifications/ts4-zkp.md

[57] W. W. W. C. [W3C], "Verifiable Credentials Data Model v2.0," 5 2025. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/

[58] European Commission, "eIDAS 2.0," 10 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:02014R0910-20241018

[59] SOG-IS, "SOG-IS Agreed Mechanisms," 2 2023. [Online]. Available: https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.3.pdf

[60] N. Desmoulins, A. Dumanois, S. Kane, and J. Traoré, "Making BBS anonymous credentials eIDAS 2.0 compliant," Cryptology ePrint Archive, Paper 2025/619, 2025. [Online]. Available: https://eprint.iacr.org/2025/619

[61] L. H. Wallets, "Choose your Ledger device ," URL https://support.ledger.com/article/360015259693-zd, accessed on November 2024.

[62] 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

[63] T. Dayaratne, X. Fan, Y. Liu, and C. Rudolph, "SSI4IoT: Unlocking the potential of iot tailored self-sovereign identity," *arXiv preprint arXiv:2405.02476*, 2024.

APPENDIX A
BENCHMARK USAGE AND EXTENSION

A. Adding a Custom Selective Disclosure Scheme

To include a novel selective disclosure algorithm in the benchmark, any additional libraries it requires and their respective versions must be specified in the `Cargo.toml` file located in the source directory. These dependencies will be automatically fetched and compiled by Cargo, making them available at runtime for benchmarking. For a fair evaluation, it is assumed that any custom selective disclosure algorithm integrated into the benchmark leverages JWTs, consistently with all other algorithms already included in the library. To facilitate token generation, the benchmark library provides a superclass `SdAlgorithm`, which exposes the necessary functions to manage core operations. In particular, the methods `encode_jwt` and `encode_and_sign_jwt` automatically convert a `serde_json::Map` object (hereafter referred to simply as `Map`) into a JWT. Conversely, `decode_jwt` and `decode_and_verify_jwt` decode a JWT back into a `Map`. These functions are essential for issuing and verifying both VCs and VPs.

B. Implementing a Custom Selective Scheme

Due to the heterogeneity of selective disclosure mechanisms, it is not feasible to provide a universal template that accommodates all variants. Therefore, once the required libraries for a custom method are made available, users must implement the mechanism themselves, place the resulting files in the `src/sd_algorithms` directory, and register them in the `mod.rs` file. Four core functions are necessary for the benchmark to properly evaluate each selective disclosure mechanism. None of them is required to respect a specific function signature yet. The core functions are:

1) The **`issue_vc`** is the first function invoked by the benchmark. Takes as input a `Map` containing mock claims and all the necessary data to issue a VC leveraging the user's mechanism (e.g., private keys). This function extracts the claims from the mock VC via `SdAlgorithm`'s `extract_claims` function, possibly removes them from the `Map` with `SdAlgorithm`'s `remove_claims` function, and produces a VC built upon the user's provided mechanism. For the fairness of evaluation, each claim and each necessary value to be included in the VC should be introduced through the `SdAlgorithm`'s `serialize_and_insert` function. Considering that VC requires no proof of possession, its encoding as JWT should be done with the `encode_JWT` function. `issue_vc` return both the `Map` containing the VC, and its JWT encoded version.
2) The second step consists of verifying a VC through the **`verify_vc`** function. This method takes as input the previously issued VC and all the data required to verify it according to the user's mechanism (e.g., public keys). With the help of `SdAlgorithm`'s `get_and_decode` function, it is possible to extract all the necessary data from the VC and proceed with its verification. A `Result<(), String>` should be returned, containing a message specifying the error in case the verification fails.
3) To simulate the VP generation, the user should provide the implementation **`issue_vp`**. Input parameters supplied to the benchmark are the previously issued VC, a vector containing a set of disclosures, and the owner's Elliptic Curve (EC) private key to simulate the proof of possession. Considering the wide plethora of selective disclosure mechanisms, more parameters can be inputted. For example, in the case of BBS+, to compute a derived signature, the issuer's public key is required. Typically, the code in this function clones the VC and modifies it by inserting and removing attributes in its `Map` using the previously mentioned methods. Finally, it returns the VP in both as a `Map`, and as its JWT encoding, this time signed with the owner's EC private key via `SdAlgorithm`'s `encode_and_sign_jwt` function.
4) Finally, the benchmark requires the verification of a VP by means of the **`verify_vp`** function. As opposed to `verify_vc`, since they are typically transferred as a string encoding, this function no longer takes as input a `Map` containing the VP but rather its JWT. Additional data (e.g., the owner's EC public key) must also be supplied. Internally, this method decodes and verifies the JWT by means of `SdAlgorithm`'s `decode_and_verify_jwt` function, extracts from the VP all the data necessary for the verification, and proceeds to verify it. A `Result<(), String>` is returned, containing an explanatory error in case of failure of the verification.

In addition to `SdAlgorithm`, the benchmark provides other superclasses tailored to the type of selective disclosure algorithm (e.g., signature-based, hash-based), offering common utility functions. Users are encouraged to adopt and extend these classes as needed. Existing implementations in the `src/sd_algorithms` directory and its subfolders serve as practical examples for integrating custom mechanisms, supported by extensive documentation. As with any Rust-based development, writing tests for custom code is strongly recommended before proceeding to the next step.

C. Pattern Adapter

To enable seamless integration with benchmarking functions, each selective disclosure algorithm must adhere to a standardized interface. Given the diversity of approaches and the impracticality of a universal function signature, a pattern adapter is employed to harmonize custom implementations. This involves creating, between benchmarking methods and

selective disclosure algorithms, a layer that translates user-defined functions into a predefined signature by leveraging a custom structure. Such a structure must encapsulate all necessary data for the selective disclosure mechanism to operate correctly (e.g., cryptographic keys and setup parameters), ensuring that all dependencies are self-contained. As a result, four core functions, corresponding to the ones defined earlier, must be implemented. Additionally, a setup method is required to initialize essential components such as cryptographic keys or random number generators.

Each algorithm `Alg` must implement an `AlgAdapter` conforming to the `Adapter` trait in `src/adapters`. Since the benchmarking functions do not accept external input parameters, all required data must reside within the adapter structure and be accessed via its `Self` instance. The `Adapter` trait exposes seven methods:

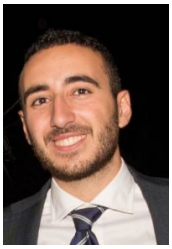
1) **`sd_algorithm`**: This method returns the name of the underlying Selective Disclosure algorithm as a string.
2) **`new`**: This method, standard for every Rust class, receives the number of claims to be included in the VC and, according to it, must initialize and set up every single self-contained data to be used in the adapter. This method returns a `Result` including either the `AlgAdapter` structure itself or, in case of failure, an error specifying the reason.
3) **`issue_vc`**: This method wraps the underlying `Alg`'s `issue_vc` function to produce a VC. The only argument taken as input is a `Map` containing the claims to be inserted in the VC. Similar to the wrapped function, this method must return a `Result` containing the VC in both its representations: as a `Map` and as a JWT string, or an error containing a string that specifies the reason for failure.
4) **`verify_vc`**: This method wraps the underlying `Alg` function that verifies a VC. Takes as input the VC as a `Map`, and returns a Result containing a String in case of error.
5) **`issue_vp`**: This method receives as input the VC as a `Map` and a list of disclosures to produce a VP by using `Alg`'s `issue_vp`.
6) **`verify_vp`**: This method wraps the `Alg`'s `verify_vp`. It takes as input the VP as a JWT string, and returns a `Result` with an error string if the verification fails.
7) **`issuer_keypair`**: This method returns the issuer's public and private keys employed in the selective disclosure mechanism for the asymmetric verification method as strings.

Implementing an adapter for each selective disclosure algorithm is relatively straightforward. Existing classes in the `src/adapters` directory can serve as practical references for writing custom implementations. Once the adapter `AlgAdapter` is defined, it can be registered in the `initialize_sd_algorithms` function within the main file by following the existing structure. To exclude specific selective disclosure methods from the benchmark, a developer can simply remove their entries from the same function. Furthermore, the benchmark parameters, such as the maximum number of claims in the mock credential or the number of iterations used to compute average timings, can be adjusted by modifying the input arguments of the `benchmark_multiple_mock_claims` function in the `main` file.

D. Data Collection

Upon completion of the benchmark execution, all serialized data stored in the `csv_dir` directory can be imported into the Jupyter notebook provided in the source folder. This notebook enables users to visualize and analyze the results through a set of previously configured plots.

Alessandro Buldini received his Master's Degree in Computer Engineering in 2023 from the University of Bologna, Italy. He is currently a Research Fellow in Information Security at the University of Bologna, where he focuses on advancing digital identity systems, with a particular emphasis on verifiable credentials and decentralized identity frameworks that enhance privacy, trust, and interoperability across digital platforms.



Carlo Mazzocca received his Ph.D. in Computer Science and Engineering from the University of Bologna, Bologna, Italy, in 2024. Currently, he is an Assistant Professor in Tenure Track at the University of Salerno, Italy. His research primarily focuses on digital identity, security, and privacy of Federated Learning. He serves as Consulting Area Editor for IEEE Transactions on Information Forensics and Security. He has served on the program committees of top-tier cybersecurity conferences, including USENIX Security, NDSS, and CCS.

Rebecca Montanari is a Full Professor at the University of Bologna, Italy, focusing on blockchain, information security, and middleware solutions for mobile and IoT services. She has collaborated with leading research centers in the semantic web and policy management, including Nokia Research Center Cambridge, MIT, Imperial College London, and IHMC Pensacola. She has served as coordinator and participant in numerous ICT research projects across Europe and internationally, with funding from Italian ministries and regional programs.

Selcuk Uluagac is currently an Eminent Scholar Chaired Professor in the School of Computing and Information Science, where he is the Director of the Center for Integrated SEcurity, PRivacy, and Trustworthy AI (CIERTA) and leads the Cyber-Physical Systems Security Lab (CSL). Before, he was a Senior Researcher at Georgia Tech. He holds a PhD from Georgia Tech and MS from Carnegie Mellon University. He received US National Science Foundation CAREER Award, US Air Force Office of Sponsored Research's Summer Faculty Fellowship, and Google's ASPIRE Research award in security and privacy, inter alia. He is an expert in the areas of cybersecurity and privacy. He has hundreds of publications in the most reputable venues as well as numerous patents. His research has been funded by numerous government agencies and industry. He has chaired/served on the of top-tier security conferences, e.g., NDSS, USENIX, ACM CCS, IEEE SP, and serving as the deputy editor in-chief of IEEE TIFS and associate editors of Elsevier COMNET journals. More information can be obtained from https://users.cs.fiu.edu/ suluagac/.