A First Appraisal of Cryptographic Mechanisms for the Selective Disclosure of Verifiable Credentials

Andrea Flamini[2][a], Silvio Ranise[1,2][b], Giada Sciarretta[1][c], Mario Scuro[2][d], Amir Sharif[1][e] and Alessandro Tomasi[1][f]

[1]*Center for Cybersecurity, FBK, Trento, Italy*

[2]*Department of Mathematics, University of Trento, Trento, Italy*

fl

Keywords: Selective Disclosure, Verifiable Credentials, Zero-Knowledge Proof, eIDAS 2, GDPR.

Abstract: Verifiable credentials are a digital analogue of physical credentials. Their authenticity and integrity are protected by means of cryptographic techniques, and they can be presented to verifiers to prove claims about the holder of the credential itself. One way to preserve privacy during presentation consists in selectively disclosing the attributes in a credential. In this paper we present the most widespread cryptographic mechanisms used to enable selective disclosure of attributes, describing their structure and comparing them in terms of performance, size of the associated verifiable presentations, and the ability to produce predicate proofs and unlinkable presentations.

1 INTRODUCTION

As more services move online, increasing importance is given to an individual's digital identity as the foundation for secure and trusted online interactions, such as e-government and e-commerce.

A new paradigm for identity management based on digital identity wallets is emerging to empower data subjects to selectively disclose credentials in a privacy-preserving and secure way. The most prominent example is the revised regulation eIDAS 2 (EU, 2021), proposing a European Digital Identity (EUDI) Framework and Wallet to improve cross-border interoperability. The privacy enhancing aims of the EUDI wallet include offering data subjects the means to control who has access to which of their personally identifiable information, and making it possible to selectively disclose only some of the attributes in their credentials to trusted parties. These aims are technically non-trivial to achieve; a service provider has to satisfy the principles of *data minimisation* and *privacy by design* under the GDPR (EU, 2016), while considering trade-offs between simplicity vs sophistication of protocol, implementation, and deployment issues including resource constraints.

Scenario. To exemplify disclosure, we consider the following simplified scenario: a young adult wishes to purchase alcohol and to prove that s/he is over the legal age limit in the jurisdiction, e.g., 18, without fully disclosing his/her entire mobile driving license (mDL).

In this example, the agency in charge of issuing mDL (Issuer) verifies the mDL Subject's age during the issuance process and includes it as an attribute in the mDL. The data Subject holding the mDL can select to disclose the single mDL attribute "age" to the liquor store employee (Verifier). The Verifier can verify that the Subject is of age to buy alcohol without learning any other personal information.

This enhances privacy for the Subject while enabling the Verifier to verify their age in a trusted and efficient manner.

Contributions. eIDAS 2 states that EUDI wallets "should technically enable the selective disclosure of attributes", and amendments to the proposal add "where attestation of attributes does not require the identification of the user, zero knowledge attestation shall be performed" (EU, 2023). The EUDI Wallet

[a] https://orcid.org/0000-0002-3872-7251
[b] https://orcid.org/0000-0001-7269-9285
[c] https://orcid.org/0000-0001-7567-4526
[d] https://orcid.org/0000-0003-2410-3760
[e] https://orcid.org/0000-0001-6290-3588
[f] https://orcid.org/0000-0002-3518-9400

Flamini, A., Ranise, S., Sciarretta, G., Scuro, M., Sharif, A. and Tomasi, A.
A First Appraisal of Cryptographic Mechanisms for the Selective Disclosure of Verifiable Credentials.
DOI: 10.5220/0012084000003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 123-134
ISBN: 978-989-758-666-8; ISSN: 2184-7711

Architecture and Reference Framework (ARF) (DG CONNECT, 2023), intended to provide more concrete technical guidelines and tools, states that "attestation MUST enable Selective Disclosure of attributes by using Selective Disclosure for JSON Web Tokens (SD-JWT) and Mobile Security Object (ISO/IEC 18013-5) scheme".

Both schemes cited in the ARF are based on hiding commitment mechanisms - generating a commitment to a value while keeping it hidden, with the ability to reveal the committed value later. The ARF does not currently cover zero knowledge proofs (ZKP) - e.g., repeatedly proving knowledge of a value without ever having to reveal it. Given the complexity and range of available options, it is non-trivial to assess their pros and cons. In order to facilitate an informed choice, we start to fill this gap by providing cryptographic building blocks for credentials with selective disclosure capability based on hiding commitments and ZKP. This paper provides the following main contributions:

- We summarize four cryptographic mechanisms (`cm`) for selective disclosure based on hiding commitment and ZKP.
- We provide the structure of Verifiable Credentials and Presentations for the `cm`, together with the operation of entities that must be performed for their creation (issuing) and consumption (presentation).
- We compare the `cm` w.r.t. several features to assist in selecting the most appropriate for the use case of interest.

Outline. Section 2 introduces the cryptographic primitives used to implement the cryptographic mechanisms described in Sections 3 and 4. In Section 5, we analyse and compare the mechanisms and we discuss how they support some privacy-enhancing features. We summarize the main results and discuss future work in Section 6.

2 SELECTIVE DISCLOSURE

Following the VC data model (Sporny et al., 2022), a credential can be defined as "a set of one or more claims [assertions about a Subject] made by an Issuer", and a Verifiable Credential (VC) as "a tamper-evident credential that has authorship that can be cryptographically verified". We consider the following entities and quote the descriptions from (Lodderstedt et al., 2023):

Issuer: "a role an entity can perform by asserting claims about one or more subjects, creating a VC from these claims, and transmitting the VC to a holder".

Holder: "a role an entity might perform by possessing one or more VCs and generating presentations from them".

Subject: "the entity about which claims are made".

Verifier: "a role an entity performs by receiving one or more VCs, optionally inside a verifiable presentation" and verifies it "to make a decision regarding providing a service to the Subject".

There are several methods that allow VCs to support selective disclosure. (Sporny et al., 2019), identifies the following categories:

Atomic credentials contain only a single claim: the Issuer may provide a set of atomic credentials and the Holder presents to a Verifier only those that it wants to show.

Hashed values allow an Issuer to issue a single VC containing multiple claims. Each claim is hidden and committed to using hash functions, then the commitment is signed by the Issuer. Examples include hash lists (Section 3.1) and Merkle trees (Section 3.2).

Selective disclosure signatures are signatures schemes that natively support selective disclosure of VC claims by using non-interactive zero knowledge proofs. Examples are *CL signature* (Section 4.1) and *BBS+ signature* (Section 4.2).

We provide noteworthy examples of cryptographic mechanisms based on hashed values, considered as an instance of hiding commitments, which are adopted in the standardized mobile Driving License (ISO/IEC 18013-5, 2021) or discussed in (Steele and Prorock, 2021) (Section 3). We also present examples of the most relevant selective disclosure signatures adopted in (IBM, 2010; Khovratovich et al., 2022; Lodder et al., 2019)(Section 4). Atomic credentials are unwieldy to manage, particularly to guarantee that a presentation contains a collection of atomic credentials that is valid as a whole, but do not introduce or require substantially different cryptographic techniques than the other two mechanisms; therefore, we do not discuss them further.

2.1 VC Structure and Lifecycle

We describe the general structure of VCs and Verifiable Presentations (VPs) regardless of the cryptographic mechanism used.

A VC is composed of three sections: an *Issuer protected header*, containing general information about the credential, for instance the Issuer, the Subject and the credential type, an *Issuer payload* containing information about the credential attributes, and an *Issuer proof* which contains the cryptographic material which attests the authenticity of the credential (see Table 1).

A VP is composed of three sections (see Table 2): a *presentation protected header* with general information about the credential; a *presentation payload* with information related to the disclosed attributes; and a *presentation proof* with the cryptographic material that allows the Verifier to check the authenticity of the presentation.

The structure of the VC and VP we adopt is consistent, albeit simplified to focus on selective disclosure, with the structure of JSON Web Proof (JWP) (Miller et al., 2023), a proposal to standardize a JSON container which aims to describe the structure of VCs to allow the selective disclosure of attributes.

In a preliminary *set-up phase*, the Issuer must generate its private-public key pair $(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{pk}_{\mathtt{Iss}})$ using the key generation function of the digital signature scheme used to sign the VCs, $\mathtt{keyGen}()$. In the *issuing phase*, the Issuer generates an Issuer proof with the function $\mathtt{genIssuerProof}(-)$. The Holder, upon reception of the VC created by the Issuer, verifies its validity computing the function $\mathtt{verIssuerProof}(-)$.

In the *presentation phase* the Holder can create a VP specifying the attributes it wants to disclose. In particular, the Holder creates the VP containing the *Holder-generated proof* by computing the function $\mathtt{genHolderProof}(-)$. The Verifier, upon reception of the VP computes the function $\mathtt{verPresentProof}(-)$ to verify it and possibly accept the Holder's claims.

2.2 Cryptographic Building Blocks

We provide the main cryptographic notions that are useful to understand the approaches for the creation of VCs supporting selective disclosure of attributes: hashing and salting for the creation of hiding commitments, and ZKP to prove statements about undisclosed attributes in selective disclosure signatures.

Digital signatures define the algorithms $\mathtt{keyGen}()$ to generate the public and private keys $(\mathtt{pk}, \mathtt{sk})$, $\mathtt{genSig}(\mathtt{sk}, m)$ to sign a message m, and $\mathtt{verSig}(\mathtt{pk}, m, \sigma)$ to verify the signature σ.

While the algorithms to implement the above functions mentioned in Section 3 may be any standardized digital signature algorithm, those described in Section 4 are a special class of signatures designed to support ZKP, and may require more structured input, e.g., ordered lists of messages. We use the same notation for brevity, but we stress that they enable different features.

2.2.1 Hash and Salt Technique

A *commitment scheme* allows a party to commit to a value v by sending a commitment, and then to reveal v by *opening* the commitment at a later point in time. A commitment scheme is *hiding* if the commitment reveals nothing about v.

In hiding commitments based on a hash function $\mathcal{H}$, the commitment creation algorithm takes as input a value v to be committed to, and outputs $\mathcal{H}(v||s)$, where s is chosen uniformly at random and is referred to as the salt of the commitment, and $v||s$ is the concatenation of the bytes strings v and s. This scheme is said to be:

- *binding*: it is computationally infeasible to find another pair $(v', s') \neq (v, s)$ such that $\mathcal{H}(v||s) = \mathcal{H}(v'||s')$;
- *hiding*: no information about v can be gained by only looking at the commitment $\mathcal{H}(v||s)$.

Only someone who knows both v and s can open the commitment (by revealing v, s) and prove that the message committed to is v.

2.2.2 Non-Interactive Zero-Knowledge Proofs

Non-interactive zero-knowledge proofs (NIZKP) allow an actor, called prover, to convince another actor, called verifier, of the truthfulness of some claim, without revealing anything else to the verifier. The protocol is non-interactive, meaning the prover generates a proof π and the verifier checks that π is valid without requiring additional interactions between prover and verifier.

In Section 4 we mention two NIZKP based on the NIZKP for linear relations: given a group $\mathbb{G}$ - which in Section 4.1 will be a group of unknown order and in Section 4.2 will be a group of prime order p - and $y, g_1, \ldots, g_m \in \mathbb{G}$, the prover can prove that it knows $a_1, \ldots, a_m$ such that $y = \prod_{i=1}^{m} g_i^{a_i}$. For an introduction to such protocols see Section 19.5.3 of (Boneh and Shoup, 2023) and (IBM, 2010).

3 HIDING-COMMITMENT MECHANISMS

Instances of hiding commitment mechanisms can be obtained, for example, by using lists of hash-based

hiding commitments (cmtList, see Section 3.1), Merkle Trees (merTree, see Section 3.2), or vector commitments (Catalano and Fiore, 2013), as suggested in (Steele and Prorock, 2021).

The Issuer commits to a set of attributes, then digitally signs the commitment. The properties of hiding commitments allow the Issuer of a credential to sign the commitments, then a Holder, who knows the attribute values of a credential, can open only some of the committed values proving to a Verifier the truthfulness of its claims.

Operations in the Issuing Phase. The Issuer can create a VC with the structure of Table 1 and issues it to the Holder. The VC is composed of the three parts already mentioned:

- the *Issuer protected header* containing the cryptographic mechanism identifier cm, - specifying primitives such as the chosen digital signature algorithm and cryptographic hash function - and the Issuer public key $\mathtt{pk}_{\mathtt{Iss}}$;
- the *Issuer payload* containing a list of attributes $\mathtt{A} = (a_1, \ldots, a_m)$ certified by the Issuer who created the credential, together with a list of random salts, one for each attribute $\mathtt{S} = (s_1, \ldots, s_m)$;
- the *Issuer Proof* containing the digital signature of the commitment CMT to the attributes A, constructed according to the chosen cryptographic mechanism and the list of attributes and salts, signed by the Issuer, obtaining $\sigma = \mathtt{genSig}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{CMT})$. These operations are performed executing the function $\mathtt{genIssuerProof}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{A}, \mathtt{S})$.

Note that the choice of the digital signature scheme adopted by the Issuer to sign the CMT is not restricted to a specific primitive.

The Holder can verify the VC's validity by computing the function $\mathtt{verIssuerProof}(\mathtt{VC})$, which consists in verifying that the commitment CMT is actually a commitment to the elements in A and S, and verifying the Issuer's digital signature.

Operations in the Presentation Phase. The Holder creates a VP to convince the Verifier that the attributes revealed are included in a credential issued by a trusted Issuer.

A VP in this context has the structure described in Table 2. It is composed by:

- a *presentation protected header* containing the name of the cryptographic mechanism cm adopted in the creation of the underlying credential and the Issuer public key;
- the *presentation payloads*, containing a subset $\mathtt{DA} \subset \mathtt{A}$ of attributes $(a_{i_1}, \ldots, a_{i_d})$ that the Holder wants to disclose together with $\mathtt{DS} \subset \mathtt{S}$, the list of associated salts $(s_{i_1}, \ldots, s_{i_d})$;
- a *presentation proof* generated by the Holder including the commitment CMT and its signature σ created by the Issuer associated to $\mathtt{pk}_{\mathtt{Iss}}$ and the Holder-generated proof obtained computing the function $\mathtt{genHolderProof}(\mathtt{DA}, \mathtt{DS}, \mathtt{A}, \mathtt{S})$.

Table 1: A simplified representation of VC which allows selective disclosure of attributes.

VC	**Hiding-commitment**	**Selective disclosure signature**
Issuer Protected Header	Cryptographic mechanism: cm Issuer public key: $\mathtt{pk}_{\mathtt{Iss}}$	Cryptographic mechanism: cm Issuer public key: $\mathtt{pk}_{\mathtt{Iss}}$
Issuer Payloads	Attributes and salts: $\mathtt{A} = (a_1, \ldots, a_m)$ $\mathtt{S} = (s_1, \ldots, s_m)$	Attributes: $\mathtt{A} = (a_1, \ldots, a_m)$
Issuer Proof	Signed commitment: $\mathtt{genIssuerProof}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{A}, \mathtt{S}) =$ $= (\mathtt{CMT}, \sigma = \mathtt{genSig}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{CMT}))$	Selective disclosure signature: $\mathtt{genIssuerProof}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{A}) =$ $= \sigma = \mathtt{genSig}(\mathtt{sk}_{\mathtt{Iss}}, \mathtt{A})$

Table 2: The general structure of a VP.

VP	**Hiding-commitment**	**Selective disclosure signature**
Presentation Protected Header	Cryptographic mechanism: cm Issuer public key: $\mathtt{pk}_{\mathtt{Iss}}$	Cryptographic mechanism: cm Issuer public key: $\mathtt{pk}_{\mathtt{Iss}}$
Presentation Payloads	Disclosed attributes and salts: $\mathtt{DA} = (a_{i_1}, \ldots, a_{i_d}) \subset \mathtt{A}$ $\mathtt{DS} = (s_{i_1}, \ldots, s_{i_d}) \subset \mathtt{S}$	Disclosed attributes: $\mathtt{DA} = (a_{i_1}, \ldots, a_{i_d}) \subset \mathtt{A}$
Presentation Proof	Signed commitment: $(\mathtt{CMT}, \sigma)$ Holder-generated Proof: $P = \mathtt{genHolderProof}(\mathtt{DA}, \mathtt{DS}, \mathtt{A}, \mathtt{S})$	Holder-generated proof: $P = \mathtt{genHolderProof}(\mathtt{pk}_{\mathtt{Iss}}, \mathtt{DA}, \mathtt{A}, \sigma)$

The Verifier verifies a VP received from the Holder by computing the function `verPresentProof(VP)`, which consists in (i) verifying the signature of the `CMT` created by the Issuer, and (ii) verifying the proof that the disclosed attributes in `DA` are a subset of the attributes committed to in `CMT`.

Once the commitment opening algorithm for the pairs (a_i, s_i) in $\texttt{DA} \times \texttt{DS} \subset \texttt{A} \times \texttt{S}$ is defined, the functions $\texttt{genHolderProof}(\texttt{DA}, \texttt{DS}, \texttt{A}, \texttt{S})$ and $\texttt{verPresentProof}(\texttt{VP})$ are well defined.

3.1 Commitment List Mechanism

In the `cmtList` mechanism, credentials contain ordered lists of attribute-salt pairs; for each pair, the issuer creates a hiding commitment, then signs the list of commitments.

In $\texttt{genIssuerProof}(\texttt{sk}_{\texttt{Iss}}, \texttt{A}, \texttt{S})$, the Issuer generates a random salt s_i for each attribute a_i and computes the commitment list entries $L_i = \mathcal{H}(a_i||s_i)$. Finally, $\texttt{CMT} = [L_i]_{i=1}^{\#A}$ is signed by the Issuer to create the Issuer proof.

Since the payload of a Holder-generated VP (Table 2, column 2) contains all the information needed to open the commitments to the disclosed attributes, the Presentation Proof only contains the signed commitment - `genHolderProof` is the null function.

In $\texttt{verPresentProof}(\texttt{VP})$ the Verifier verifies the Issuer signature of `CMT` and compares $\mathcal{H}(a_{i_j}||s_{i_j})$ with L_{i_j}, for each $(a_{i_j}, s_{i_j}) \in \texttt{DA} \times \texttt{DS}$. If the signature is verified and the digests $\mathcal{H}(a_{i_j}||s_{i_j})$ match with L_{i_j}, the VP is accepted.

3.2 Merkle Tree Mechanism

The `merTree` mechanism uses Merkle trees to create commitments `CMT`.

$\texttt{genIssuerProof}(\texttt{sk}_{\texttt{Iss}}, \texttt{A}, \texttt{S})$: the Issuer generates one random salt s_i for each attribute a_i, then uses their ordered concatenated pairs as leaves of a Merkle tree. The Issuer sets the `CMT` equal to the Merkle tree root

$$R = \texttt{getRoot}(a_1||s_1, a_2||s_2, \ldots, a_m||s_m). \quad (1)$$

An example of Merkle tree is given in Figure 1.

To create a VP, the Holder includes the presentation payload as in column 2 of Table 2. The presentation proof, together with the signed commitment, also requires the Holder-generated proof, which the Holder obtains by computing the inclusion paths of the attributes that the Holder wants to disclose.

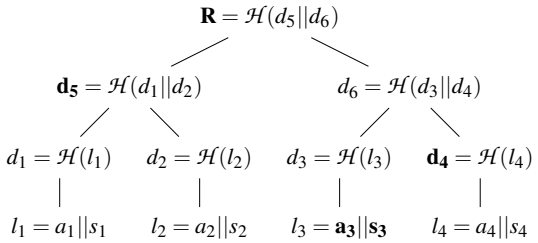

Figure 1: Merkle tree constructed over 4 leaves. Disclosing $a_3||s_3$, their inclusion proof in R is $[3, d_4, d_5]$.

The Verifier verifies the presentation computing $\texttt{verPresentProof}(\texttt{VP})$ verifying the signature of `CMT` and verifying that the inclusion paths in P let the Verifier reconstruct the signed root R, for each $(\texttt{a}_{\texttt{i}_\texttt{j}}, \texttt{s}_{\texttt{i}_\texttt{j}}) \in \texttt{DA} \times \texttt{DS}$.

For example, the inclusion path of the leaf l_3 in position 3 of the Merkle tree in Figure 1, given the public root R, is $[3, d_4, d_5]$. In order to verify the inclusion of l_3, the Verifier computes $d_3 = \mathcal{H}(l_3)$, $d_6 = \mathcal{H}(d_3||d_4)$, and verifies that $\mathcal{H}(d_5||d_6) = R$.

4 SELECTIVE DISCLOSURE SIGNATURE MECHANISM

Selective disclosure signatures, following the naming in (Sporny et al., 2019), are a class of digital signature algorithms that enable (a) an Issuer to sign multiple attributes with a single signature, (b) a Holder to prove possession of a signature and some undisclosed attributes, generating fresh ZKP without involving the Issuer - recall Section 2.2.2, and (c) a Verifier to verify the validity of a disclosed subset of attributes, given only the ZKP of undisclosed attributes and signature.

Examples of selective disclosure signatures are `CL` (Section 4.1) and `BBS+` (Section 4.2), which are *multimessage signature algorithms* for which an ordered list is input to the signature generation $\texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, (\texttt{a}_1, \ldots, \texttt{a}_\texttt{m})) = \sigma$ and signature verification $\texttt{verSig}(\texttt{pk}_{\texttt{Iss}}, (\texttt{a}_1, \ldots, \texttt{a}_\texttt{m}), \sigma) = \texttt{true/false}$.

Operations in the Issuing Phase. The VC based on the use of selective disclosure signature algorithms as cryptographic mechanism is composed of three parts (see column 3 of Table 1):

- Issuer protected header, containing the name of the cryptographic mechanism `cm` i.e., the chosen selective disclosure signature scheme, and the Issuer public key $\texttt{pk}_{\texttt{Iss}}$;
- Issuer payloads, containing the list of attributes $\texttt{A} = (a_1, \ldots, a_m)$;

- Issuer proof, containing the *selective disclosure signature* (SDSig) of the attributes in A, $\sigma = \texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, \texttt{A})$.

Therefore the function that allows the Issuer to create the Issuer proof is just $\texttt{genIssuerProof}(\texttt{sk}_{\texttt{Iss}}, \texttt{A}) = \texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, \texttt{A})$, and the function that allows the Holder to verify it is $\texttt{verIssuerProof}(\texttt{VC}) = \texttt{verSig}(\texttt{pk}_{\texttt{Iss}}, \texttt{A}, \sigma)$.

Operations in the Presentation Phase. To selectively disclose some attributes of a VC to a Verifier, the Holder creates a VP (see column 3 of Table 2) composed of:

- presentation protected header, containing the name of the cryptographic mechanism cm and the Issuer public key;
- presentation payload, containing the list $\texttt{DA} = (a_{i_1}, \ldots, a_{i_d})$ of disclosed attributes;
- presentation proof P, generated by the Holder executing $\texttt{genHolderProof}(\texttt{pk}_{\texttt{Iss}}, \texttt{DA}, \texttt{A}, \sigma)$, a NIZKP of the signature σ, certifying the revealed attributes in DA and proving in zero-knowledge the knowledge of the hidden attributes in $\texttt{A} \setminus \texttt{DA}$.

The Verifier verifies the NIZKP P by computing the function $\texttt{verPresentProof}(\texttt{VP})$.

For both BBS+ and CL we provide a high level description of $\texttt{genHolderProof}(\texttt{pk}_{\texttt{Iss}}, \texttt{DA}, \texttt{A}, \sigma)$ and $\texttt{verPresentProof}(\texttt{VP})$, including references to computation details omitted for brevity.

CMT vs SDSig. The purpose of CMT is to bind the attributes into an item that is subsequently signed by the Issuer. The Holder can perform selective disclosure by revealing CMT, the attributes to be disclosed, and a presentation proof. On the other hand, SDSig simultaneously binds the attributes into an item that is itself a digital signature, certifying the authorship of the VC. To create a presentation, the Holder must not reveal SDSig, but rather derive from SDSig a randomized proof that assures the Verifier about the claims. For a comparison between CMT vs SDSig w.r.t. important features, see Section 5.

4.1 CL Signature

The CL signature scheme was presented in (Camenisch and Lysyanskaya, 2002).

VC Creation and Verification. To create SDSig for a VC, an Issuer signs the attributes $a_1, \ldots, a_m \in \texttt{A}$ using the following CL digital signature algorithm as defined in (IBM, 2010).

Key generation algorithm $\texttt{keyGen}()$. Let k be the security parameter and $n \leftarrow pq$ be an ℓ_n-bit *special RSA modulus* [1], and choose uniformly at random quadratic residues $R_1, \ldots, R_m, S, Z$ [2].

Output the public key

$$\texttt{pk}_{\texttt{Iss}} = (n, R_1, \ldots, R_m, S, Z) \tag{2}$$

and the secret key $\texttt{sk}_{\texttt{Iss}} = (\texttt{p})$.

Signing algorithm $\texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, \texttt{A})$. On input $\texttt{A} = \{a_1, \ldots, a_m\}, a_i \in \{0,1\}^{\ell_a}$, choose a random prime number $e \in \{0,1\}^{\ell_e}$, $\ell_e > \ell_a + 2$, $e > 2^{\ell_e - 1}$, and a random number $v \in \{0,1\}^{\ell_v}$, where $\ell_v = \ell_n + \ell_a + \ell_\emptyset$ with $\ell_\emptyset$ a security parameter (e.g. $\ell_\emptyset = 80$). Compute

$$A \leftarrow \left(\frac{Z}{R_1^{a_1} \ldots R_m^{a_m} S^v} \right)^{\frac{1}{e}} \mod n. \tag{3}$$

The resulting output signature is

$$\begin{aligned} \sigma &= \texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, (a_1, \ldots, a_m)) \\ &= (A, e, v). \end{aligned} \tag{4}$$

Verification algorithm $\texttt{verSig}(\texttt{pk}_{\texttt{Iss}}, \texttt{A}, \sigma)$. To verify that the triple (A, e, v) is a signature on $a_1, \ldots, a_m$ check that the following holds:

$$Z = A^e R_1^{a_1} \ldots R_m^{a_m} S^v \mod n \tag{5}$$

$$a_i \in \{0,1\}^{\ell_a} \tag{6}$$

$$e \in [2^{\ell_e - 1} + 1, 2^{\ell_e} - 1] \tag{7}$$

These functions completely define $\texttt{genIssuerProof}(\texttt{sk}_{\texttt{Iss}}, \texttt{A})$ which corresponds to $\texttt{SDSig} = \texttt{genSig}(\texttt{sk}_{\texttt{Iss}}, \texttt{A}) = \sigma = (A, e, v) \in \mathbb{Z}_n \times \{0,1\}^{\ell_e} \times \{0,1\}^{\ell_v}$ and $\texttt{verIssProof}(\texttt{VC})$.

VP Creation. At every presentation, the Holder generates a new random signature from a signature received from the Issuer; then the Holder creates a NIZKP of knowledge of a representation for the randomized signature using an argument similar to the one about NIZKP for linear relations (Section 2.2.2) and described in detail in (IBM, 2010).

The Holder-generated proof is a NIZKP of knowledge of a signature and the attributes signed in it, generated by the function

$$\begin{aligned} P &= \texttt{genHolderProof}(\texttt{DA}, \texttt{A}, \sigma, \texttt{pk}_{\texttt{Iss}}) \\ &= (c, A', \widehat{e}, \widehat{v}', \widehat{a}_{i_1}, \ldots, \widehat{a}_{i_{(n-d)}}) \end{aligned} \tag{8}$$

[1] $n = pq$ is a special RSA modulus if $p = 2p' + 1$ and $q = 2q' + 1$ with p', q' prime numbers

[2] q is a quadratic residue modulo n if there exists $a \in \mathbb{Z}_n$ such that $q = a^2 \mod n$

with $\mathtt{pk_{Iss}}$ from Eq. (2), and σ from Eq. (4); $c \in \{0,1\}^{256}$ is the challenge of the underlying NIZKP; $\mathtt{A}' \in \mathbb{Z}_{\mathtt{n}}^*$ is a component of the randomized signature; $\widehat{e} \in \{0,1\}^{\ell'_e+\ell_{\mathcal{H}}+\ell_{\emptyset}+1}$, $\widehat{v}' \in \{0,1\}^{\ell_v+\ell_{\mathcal{H}}+\ell_{\emptyset}+1}$ and $\widehat{a}_{i_1},\ldots,\widehat{a}_{i_{(n-d)}} \in \{0,1\}^{\ell_a+\ell_{\mathcal{H}}+\ell_{\emptyset}+1}$ are the response values of the underlying NIZKP for linear relations.

VP Verification. The verification function $\mathtt{verPresentProof(VP)}$ consists in (*i*) verifying the NIZKP for linear relations to prove the Holder knows a valid undisclosed signature, and (*ii*) verifying that the size of the received values $(\widehat{e},\widehat{a}_{i_1},\ldots,\widehat{a}_{i_{(n-d)}})$ lies in the expected integer interval (IBM, 2010) to ensure that the undisclosed attributes $a_{i_1},\ldots,a_{i_{(n-d)}}$ and parameter e used to build the NIZKP have the expected size.

4.2 BBS+ Signature

The BBS+ signature was presented by Au et al. (Au et al., 2006) as a provably secure extension to BBS group signatures (Boneh et al., 2004) and improved by Camenisch et al. (Camenisch et al., 2016). An optimisation provided with a proof of security has also recently been published (Tessaro and Zhu, 2023).

VC Creation and Verification. To issue a VC, an Issuer creates $\mathtt{SDSig}$ by producing a $\mathtt{BBS+}$ signature of the messages $\mathtt{a_1},\ldots,\mathtt{a_m}$ with its private key $\mathtt{sk_{Iss}}$.

Let $\mathbb{G}_1,\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of prime order p, and $\mathbf{e}:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ be a pairing [3].

Key generation algorithm $\mathtt{keyGen}()$. Take a random vector $(h_0,\ldots,h_m)\in\mathbb{G}_1^{m+1}$, a random $x\in\mathbb{Z}_p^*$ and sets $\mathtt{sk_{Iss}}=x$ and $\mathtt{pk_{Iss}}=(w=g_2^x,h_0,\ldots,h_m)$

Signing algorithm $\mathtt{genSig}(\mathtt{sk_{Iss}}=x,\mathtt{A})$. On input the secret key x and the messages $\mathtt{A}=(a_1,\ldots,a_m)\in\mathbb{Z}_p^m$, randomly generate $e,s\in\mathbb{Z}_p$ and compute $A=(g_1h_0^s\prod_{i=1}^m h_i^{a_i})^{\frac{1}{e+x}}$. Output the triple (A,e,s).

Verification algorithm $\mathtt{verSig}(\mathtt{pk_{Iss}},\mathtt{A},\sigma)$. On input the public key

$$\mathtt{pk_{Iss}}=(w,h_0,\ldots,h_m)\in\mathbb{G}_2\times\mathbb{G}_1^{m+1}, \quad (9)$$

messages $(a_1,\ldots,a_m)\in\mathbb{Z}_p^m$, and a signature $(A,e,s)\in\mathbb{G}_1\times\in\mathbb{Z}_p^2$, check $\mathbf{e}(A,wg_2^e)=\mathbf{e}(g_1h_0^s\prod_{i=1}^m h_i^{a_i},g_2)$.

[3]A pairing is a map satisfying *bilinearity*, i.e. $\mathbf{e}(g_1^x,g_2^y)=\mathbf{e}(g_1,g_2)^{xy}$, *non-degeneracy*, i.e. for each generator $g_1\in\mathbb{G}_1,g_2\in\mathbb{G}_2$, then $\mathbf{e}(g_1,g_2)$ generates $\mathbb{G}_T$, and *efficiency* which means that the map can be efficiently computed for any input.

These algorithms define the functions $\mathtt{genIssuerProof}(\mathtt{sk_{Iss}},\mathtt{A})$ which corresponds to $\mathtt{SDSig}=\mathtt{genSig}(\mathtt{sk_{Iss}},\mathtt{A})=\sigma=(A,e,s)\in\mathbb{G}_1\times\mathbb{Z}_p^2$, and $\mathtt{verIssuerProof}(\mathtt{VC})$.

VP Creation. The Holder can generate a VP with $\mathtt{genHolderProof}(\mathtt{DA},\mathtt{A},\sigma,\mathtt{pk_{Iss}})$ whose output is obtained from the construction of a NIZKP of knowledge of the signature and the hidden attributes based on the NIZKP for linear relations. The function returns

$$\begin{aligned}P &= \mathtt{genHolderProof}(\mathtt{DA},\mathtt{A},\sigma,\mathtt{pk_{Iss}})\\ &= (A',\overline{A},d,c,\widehat{e},\widehat{r}_2,\widehat{r}_3,\widehat{s}',\widehat{a}_{i_1},\ldots,\widehat{a}_{i_{n-d}}) \quad (10)\end{aligned}$$

where $A',\overline{A},d\in\mathbb{G}_1$, and all other elements lie in $\mathbb{Z}_p$. For a detailed description we refer to (Looker et al., 2023; Camenisch et al., 2016).

VP Verification. Having received a VP from a Holder, the Verifier computes the function $\mathtt{verPresentProof}(\mathtt{VP})$, which consists in executing the verification steps of the underlying NIZKP for linear relations and verifying that the terms $\overline{A}=A'^x$ using the pairing $\mathbf{e}$.

5 SOLUTION ANALYSIS

To assess the maturity of options, we consider their standardization (Section 5.1) and their cryptographic agility (Section 5.2). Standardization is important for cryptographic protocols to ensure expert vetting of correctness, security, and other properties claimed, as well as to promote interoperability as encouraged e.g., by the proposed Interoperable Europe Act (EU, 2022). Cryptographic agility (Sullivan, 2010) "is achieved when a protocol can easily migrate from one algorithm suite to another more desirable one, over time" (Housley, 2015). The need to transition between cryptographic algorithms and key lengths has been steadily gaining importance, e.g., replacing older versions of the Secure Hash Algorithm, and preparing for quantum computing (Barker and Roginsky, 2019).

We observe how each mechanism supports privacy and offline features with regards to presentation unlinkability (Section 5.3), and briefly discuss the advantages of predicate proofs (Section 5.4). We compare the computation speed of each function described in Section 2 (Section 5.5.2), and the size of presentation elements of each mechanism (Section 5.5.3). Our assessment is summarized in Section 5.6.

5.1 Standardization

cmtList is the only mechanism featured in official standards: it is enabled by design in ISO 18013-5 (ISO/IEC 18013-5, 2021), and it is the basis for the IETF draft SD-JWT (Fett et al., 2023). Both are considered mandatory for the European digital identity wallet (DG CONNECT, 2023) developed in the context of the revised eIDAS regulation (EU, 2021).

JSON Web Proof (JWP) (Miller et al., 2023) is a proposed container format for VCs and VPs that aims to be agnostic to the proof mechanism. It is currently an IETF draft on the Standards Track. merTree has been proposed in (Steele and Prorock, 2021) as a possible mechanism for JWP. It also appears in the experimental Certificate Transparency 2.0 proposal (Laurie et al., 2021).

The BBS+ specification (Looker et al., 2023) is an IRTF draft. CL signatures are not specified independently but as part of the Identity Mixer (IBM, 2010) and Hyperledger Ursa (Khovratovich et al., 2022) anonymous credentials protocols.

5.2 Cryptographic Agility

cmtList and merTree offer the greatest agility: any cryptographic hash function can be used to construct them, and any digital signature can be chosen to sign the hash list or tree root.

BBS+ signatures can in theory be based on any pairing-friendly curve, of which several options have been identified (Sakemi et al., 2022) up to 256-bit security, and any correspondingly secure cryptographic hash function. Cipher suites have been drafted (Looker et al., 2023).

The idemix specification (IBM, 2010) for anonymous credentials with CL signatures contains a default value for 14 parameters and 7 "constraints which parameter choices must satisfy to ensure security and soundness" (Tables 2 and 3 therein), and it is left to the reader to adjust these as required. The default RSA modulus is 2048 bits, which corresponds to only 112 bits of security. Other than increasing the prime factor length, it is non-trivial to establish how parameters should change to increase the security level of the scheme as a whole.

The Ursa library also defaults to 2048-bit modulus; the Ursa specification (Khovratovich et al., 2022) lists individual parameter values scattered throughout, including 1536-bit RSA factors, but it is left to the reader to gather the information, to modify the source code, and to assume that all other parameters have been set to meet the same level of security.

5.3 Presentation Unlinkability

Unlinkability can be defined as ensuring that "no correlatable data are used in a digitally-signed payload" (Sporny and Longley, 2022). Sources of correlation include the signature itself and long-term identifiers, such as the credential subject, a credential identifier, revocation status information etc. Guaranteeing this property goes beyond selective disclosure only; here we focus on signature-based correlation.

Hiding Commitment Mechanisms. Since the Presentation Proof of a VP contains the issuer-signed commitment included in the associated VC, this identifier links each VP uniquely to one VC, and therefore to its Holder. This means that the Holder should use always different VCs to generate new VPs, therefore in the issuing phase the Issuer must provide the Holder with several distinct versions of the same VC where a distinct version of VC is built including the same set of attributes hidden using different salts.

When all the distinct versions of the same VC have been used, the Issuer must produce and send new ones to the Holder. There must therefore be an available channel between the Issuer and the Holder device storing the VCs that guarantees ready access to brand new VCs that can be used to create unlinkable presentations.

Selective Disclosure Signatures. As summarized in Table 2, the presentation proof contains only the Holder-generated proof, which is a randomized element.

Given a VC, the Holder can create a new presentation proof each time that is indistinguishable from random, and therefore cannot be correlated to other VPs. The Holder can use the same VC multiple times; therefore, interaction with the Issuer is required only when requesting a new credential or renewing an expired one.

5.4 Predicate Proofs

In some use cases, there may be an interest in asking a question ("predicate") about an attribute, without disclosing the attribute itself. For instance, a Verifier may need to know whether an mDL subject's age is over some threshold NN, or in some range, without needing to know their full date of birth. This feature would enhance privacy and follow the data minimization principle.

The cmtList mechanism used in mDL allows the Issuer to create range proofs only by treating them as individual attributes; for instance, in the

AAMVA mDL implementation guidelines (AAMVA, 2023) Issuers must identify every likely threshold value in their jurisdiction and encode a separate attribute `age_over_NN=True` or `False` for each `NN`.

The disadvantages of implementing this feature with this mechanism are: (a) an increased size of every VC and VP, (b) requiring the Issuer to keep track of when each threshold is crossed to issue a new VC, (c) interoperability issues - a Verifier may not find all the same thresholds represented every separate jurisdiction for the same VC type (e.g., age above 18, 21, 65 etc). All hiding commitment based `cm` including `merTree` suffer the same disadvantages.

By contrast, selective disclosure signature predicate proofs enable a Verifier to request an evaluation of attributes during the Presentation Phase, without prior involvement of the Issuer: they are built by the Holder as NIZKP of predicates about the attributes included in the VC. For example, *range proofs* and *set membership proof*s (Camenisch et al., 2008) allow the Prover to prove that an attribute a lies within a range $v < a < u$, or in a given set of values $a \in \mathcal{A}$, respectively. Examples of predicate proofs for the `CL` mechanism can be found in Section 6 of (IBM, 2010).

5.5 Experimental Evaluation

Our use case of interest is a proximity flow for EUDI Wallets, in which the Holder and Verifier are physically close and the attestation exchange and disclosure occurs using proximity protocols (NFC, Bluetooth, QR-Code, etc.), possibly without the Holder having internet connectivity. A concrete instance involves checking a mobile driving license (mDL), as considered in Section 1. In this scenario, the Holder device may be resource-constrained in both computational capability and presentation exchange bandwidth; we therefore measure the speed of computation, particularly `genHolderProof`, in Section 5.5.2 and the size of VP elements for each `cm` in Section 5.5.3. Based on ISO/IEC 18013-5 (ISO/IEC 18013-5, 2021), in which an mDL consists of 11 mandatory and 22 optional attributes, we use credentials with $n_a \leq 33$ total attributes.

5.5.1 Experimental Set-up

For `BBS+` and `CL` signature performance, we test the Hyperledger Ursa rust implementation,[4] the only library with both algorithms, to the best of our knowledge. For a fair comparison, we aim for an equivalent level of security of 128 bits in all tested mechanisms - see SP 800-57 (Barker, 2020), Table 2. Enforcing this common security level is non-trivial, as mentioned in Section 5.2.

For the sake of reproducibility, the parameters and algorithms used in our experiments are curve BLS12-381 and BLAKE2 for `BBS+`, 3072-bit RSA modulus and SHA-256 for `CL`, and ed25519 and SHA-256 for hiding commitment mechanisms. For `cmtList` and `merTree`[5] digital signatures we use digital signature EdDSA over ed25519 using the same rust crate[6] on which Ursa depends.

In order to test performance on both desktop PCs and constrained devices with ARM CPUs more closely resembling mobile phones - our use case for Holder devices - experiments are run on i5-4690K (3.50GHz, 4 cores), raspberry pi 3B+ 1GB RAM, pi 4B 4GB RAM, and Ryzen 7 5800X.

5.5.2 Speed

We measure the speed of key generation, and signature and presentation proof generation and verification - see Table 3 and Figure 2. The presentation phase is of particular interest since it is expected to occur frequently on constrained devices; we show results in Tables 4 and 5. There is approximately an order of magnitude difference in performance between a modern desktop CPU (Ryzen 5800X) and an ARM raspberry pi 4B, and another between the pi 4B and pi 3B+.

The speed of hashing and of generating Merkle inclusion paths is negligible compared with generating and verifying the digital signature in `cmtList` and `merTree`; we therefore only report the results of `merTree` speed tests.

Table 3: Speed by mechanism for each function. Approximate orders of magnitude in ms - lower is faster.

Function	merTree	CL	BBS+
keyGen	-2	4	0
genIssuerProof	-1	2	0
verIssuerProof	-1	2	1
genHolderProof	-3	2	1
verPresentProof	-1	2	1

Table 4: Presentation Proof generation test by CPU with $n_A = 33$. Times in ms are median over $n_D \leq n_A$.

cm	5800X	4690K	pi 4B	pi 3B+
merTree	0.0007	0.0015	0.0044	0.0143
CL	55.270	74.758	394.95	1475.9
BBS+	7.127	15.109	52.03	374.7

[4] https://docs.rs/ursa/

[5] https://docs.rs/rs_merkle/

[6] https://docs.rs/ed25519-dalek/

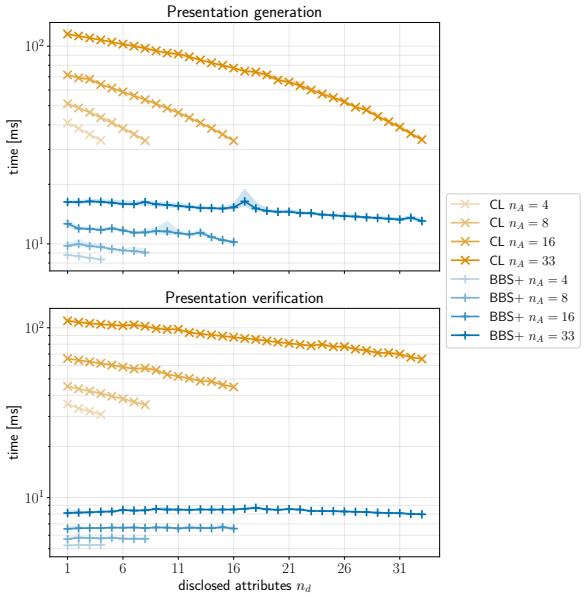

Figure 2: Presentation Proof generation and verification performance test results for CL and BBS+. merTree generation and verification speeds are 4 and 2 orders of magnitude faster than BBS+ and not shown. Solid lines are median times, shaded areas are 25^{th} and 75^{th} percentile.

5.5.3 Presentation Size

Presentation Proof. We compare VP size contributions for each cm, with trends summarized in Figure 3. Attribute size is arbitrary, so DA is not included. For commitment-based mechanisms, one disclosed salt DS per disclosed attribute must be included; therefore, VP size tends to grow with n_D for cmtList and merTree, while it decreases for CL and BBS+ due to one zero-knowledge proof per undisclosed attribute.

Presentation proof size is calculated as follows:

- cmtList: one digest per attribute in the credential, a signature of the list of digests, one disclosed salt per disclosed attribute:

$$|\texttt{CMT}| + |\sigma| + |\texttt{DS}| = dn_A + |\sigma| + sn_D \qquad (11)$$

- merTree: one tree root (of digest size), a signature of the tree root, one disclosed salt per disclosed attribute, an inclusion proof for disclosed attributes. The size of an inclusion proof for a single attribute is equal to the tree height $\lceil \log_2(n_A) \rceil$ times the digest size; a simple implementation may return a separate proof per disclosed attribute, an optimized implementation will be more compact. An upper bound is therefore:

$$|\texttt{CMT}| + |\sigma| + |\texttt{DS}| + |P| = \\ = d + |\sigma| + dn_D + \lceil \log_2(n_A) \rceil dn_D \qquad (12)$$

- CL: SDSig: in order to make a fair comparison between the algorithms, we consider as size of n 3072 bits to have a security level of 128 bits. Therefore, a NIZKP of knowledge of a signature and of the undisclosed attributes (Eq. (8)) is given by:
  - a digest $c \in \{0,1\}^{256}$ (32 bytes);
  - an element $A' \in \mathbb{Z}_n$ (384 bytes), an element $\widehat{e} \in \{0,1\}^{457}$ (58 bytes), and $\widehat{v}' \in \{0,1\}^{3744}$ (468 bytes)
  - an element $\widehat{a}_i \in \{0,1\}^{593}$ (75 bytes) for each undisclosed attribute.
- BBS+: SDSig: a NIZKP of knowledge of a signature and of the undisclosed values (Eq. (10)) is given by:
  - three elements $A', \overline{A}, d \in \mathbb{G}_1$;
  - five elements $c, \widehat{e}, \widehat{r}_2, \widehat{r}_3, \widehat{s}' \in \mathbb{Z}_p$;
  - one $\widehat{a}_i \in \mathbb{Z}_p$ for each undisclosed attribute.

  BBS+ can be implemented using the pairing-friendly elliptic curve BLS12-381, with the prime order of the subgroup of $\mathbb{G}_1$ being $p \in \{0,1\}^{256}$. Therefore, the elements in $\mathbb{G}_1$ - i.e., $A', \overline{A}, d$ - can be represented as 48-byte strings and the integer elements as 32-byte strings.

Table 5: Presentation Proof verification test by CPU with $n_A = 33$. Times in ms are median over $n_D \leq n_A$.

cm	5800X	4690K	pi 4B	pi 3B+
merTree	0.040	0.071	0.29	1.1
CL	64.969	86.393	456.00	1687.3
BBS+	4.875	8.383	30.56	187.9

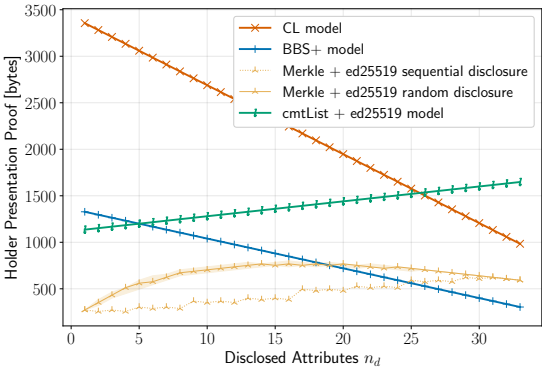

Figure 3: VP proof size comparison for $n_A = 33$. Values used for comparison, in bytes: salt size $s = 16$; digest size $d = 32$; cmtList and merTree signature size $|\sigma| = 64$ for EdDSA with curve ed25519. Model equations for cmtList (11), CL (8), BBS+ (10) shown alongside measured Merkle implementation values for sequential attribute disclosure (best-case) and random sampling of attribute combinations (median with 25th and 75th percentile shaded).

Public Keys. A VP header may contain either $\texttt{pk}_{\texttt{Iss}}$ or a reference to it. For instance, a JWS (Jones et al., 2015) header may contain $\texttt{pk}_{\texttt{Iss}}$ as JWK or X.509 certificate, or a url as JKU, or a certificate thumbprint, etc. $\texttt{pk}_{\texttt{Iss}}$ size may be calculated as follows.

- `merTree`, `cmtList`: the public key $\texttt{pk}_{\texttt{Iss}}$ of EdDSA digital signature is a 32-byte point of the curve ed25519.
- `CL`: the public key $\texttt{pk}_{\texttt{Iss}}$ of the CL signature algorithm is $(n, R_1, \ldots, R_m, S, Z) \in \mathbb{Z}_{\texttt{n}}^{\texttt{m}+3}$ of size $384(m+3)$ bytes.
- `BBS+`: the public key $\texttt{pk}_{\texttt{Iss}}$ of the BBS+ signature algorithm is $(w = g_2^x, h_0, \ldots, h_m) \in \mathbb{G}_2 \times \mathbb{G}_1^{m+1}$ of size $96 + 48(m+1)$ bytes, where $\mathbb{G}_1, \mathbb{G}_2$ are obtained using curve BLS12-381.

5.6 Assessment Summary

We find that `cmtList` and `merTree` are more standardized, easier to implement but more cumbersome for the Issuer to manage, very fast to compute but only `merTree` is reliably small in size. Predicates must be defined by the issuer, and unlinkability requires the issuer to provide a supply of single-use credentials with new attribute salts and signatures in advance. While `CL` is particularly computationally expensive and large in size, `BBS+` is computationally feasible and compact; in both cases, predicate proofs can be provided by the holder, and the randomness for unlinkability is also generated by the holder based on a single selective disclosure signature. Our assessment is summarized qualitatively in Table 6.

6 CONCLUSION

There exist several approaches exist to augment VCs with privacy-preserving selective disclosure of attributes, including cryptographic mechanisms based on hiding commitments and selective disclosure signatures. We analyzed four mechanisms: salted hash list (`cmtList`), salted hash tree (`merTree`), as well as `CL` and `BBS+` signatures. For each mechanism we defined the VP and VC structures, presented the operations to be performed to issue VCs and present VPs. Finally, we examined standardization status, cryptographic agility, and additional features such as predicate proofs and unlinkable VPs, and experimentally evaluated the performance of mechanisms with a view to a practical scenario of proximity flows with constrained devices.

Table 6: `cm` assessment summary.

Feature	`cmtList`	`merTree`	`CL`	`BBS+`
Standard	+	±	−	±
Agile	+	+	−−	+
Unlinkable	±	±	+	+
Predicates	±	±	+	+
Fast	+++	+++	−	±
Compact	−	+	−	+

We find that `merTree` is the fastest mechanism of those examined but requires constant management by the Issuer to provide selective disclosure together with unlinkability; `BBS+` is competitive in terms of size, does not require constant re-issuing of credentials on the Issuer's part, and is acceptable in speed even on constrained devices.

Future Work. As future work, we aim to describe additional selective disclosure mechanisms based on e.g., vector commitments (Catalano and Fiore, 2013) or other signatures such as PS (Pointcheval and Sanders, 2018).

We aim to provide greater detail on the algorithms for the creation and verification of the Holder generated proofs for `CL` and `BBS+`, together with the necessary preliminaries regarding the NIZKP.

ACKNOWLEDGEMENTS

The first author acknowledges support from Eustema S.p.A. through the PhD scholarship and is a member of GNSAGA of INdAM.

This work has been partially supported by "Futuro & Conoscenza Srl", jointly created by the FBK and the Italian Government Printing Office and Mint, Italy.

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

REFERENCES

AAMVA (2023). Mobile Driver's License (mDL) implementation guidelines, version 1.2. https://www.aamva.org/topics/mobile-driver-license.

Au, M. H., Susilo, W., and Mu, Y. (2006). Constant-size dynamic k-TAA. In *SCN 2006*, volume 4116 of *LNCS*, pages 111–125.

Barker, E. (2020). NIST SP 800-57r5 Recommendation for Key Management, Part 1: General.

Barker, E. and Roginsky, A. (2019). NIST SP 800-131A transitioning the use of cryptographic algorithms

and key lengths. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf.

Boneh, D., Boyen, X., and Shacham, H. (2004). Short group signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. https://doi.org/10.1007/978-3-540-28628-8_3.

Boneh, D. and Shoup, V. (2023). A graduate course in applied cryptography. https://toc.cryptobook.us/.

Camenisch, J., Chaabouni, R., and Shelat, A. (2008). Efficient protocols for set membership and range proofs. In *ASIACRYPT*, volume 5350 of *LNCS*, pages 234–252.

Camenisch, J., Drijvers, M., and Lehmann, A. (2016). Anonymous attestation using the strong Diffie Hellman assumption revisited. In *Trust 2016*, volume 9824 of *LNCS*, pages 1–20.

Camenisch, J. and Lysyanskaya, A. (2002). A signature scheme with efficient protocols. In *SCN 2002*, volume 2576 of *LNCS*, pages 268–289.

Catalano, D. and Fiore, D. (2013). Vector commitments and their applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 55–72.

DG CONNECT (2023). The European Digital Identity Wallet Architecture and Reference Framework, version 1.0.0. https://github.com/eu-digital-identity-wallet/architecture-and-reference-framework.

EU (2016). Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance). http://data.europa.eu/eli/reg/2016/679/2016-05-04.

EU (2021). Proposal for a Regulation of the European Parliament and of the Council amending Regulation (EU) no 910/2014 as regards establishing a framework for a European Digital Identity. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2021:281:FIN.

EU (2022). Proposal for a regulation of the european parliament and of the council laying down measures for a high level of public sector interoperability across the Union (Interoperable Europe Act). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52022PC0720.

EU (2023). Amendments by the European Parliament to the Commission proposal for a Regulation of the European Parliament and of the Council amending Regulation (EU) no 910/2014 as regards establishing a framework for a European Digital Identity. https://www.europarl.europa.eu/doceo/document/A-9-2023-0038_EN.html.

Fett, D., Yasuda, K., and Campbell, B. (2023). Selective disclosure for JWTs (SD-JWT). https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/.

Housley, R. (2015). Guidelines for cryptographic algorithm agility and selecting mandatory-to-implement algorithms. https://www.rfc-editor.org/rfc/rfc7696.

IBM (2010). Specification of the identity mixer cryptographic library version 2.3.0. https://dominoweb.draco.res.ibm.com/reports/rz3730_revised.pdf. Security Team, Computer Science Dept., IBM Research Zurich.

ISO/IEC 18013-5 (2021). ISO/IEC 18013-5 personal identification - ISO-compliant driving licence - part 5: Mobile driving licence (mDL) application.

Jones, M. B., Bradley, J., and Sakimura, N. (2015). JSON Web Signature (JWS). https://www.rfc-editor.org/rfc/rfc7515.html.

Khovratovich, D., Lodder, M., and Parra, C. (2022). Anonymous credentials with type-3 revocation, version 0.6. https://github.com/hyperledger/ursa-docs/tree/main/specs/anoncreds1.

Laurie, B., Messeri, E., and Stradling, R. (2021). Certificate transparency version 2.0. https://www.rfc-editor.org/rfc/rfc9162.

Lodder, M., Zundel, B., and Khovratovich, D. (2019). Pairings-based anonymous credentials with circuit-based revocation and permission policies, version 0.7. https://github.com/hyperledger/ursa-docs/tree/main/specs/anoncreds2.

Lodderstedt, T., Yasuda, K., and Looker, T. (2023). OpenID for verifiable credential issuance. https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html.

Looker, T., Kalos, V., Whitehead, A., and Lodder, M. (2023). The BBS signature scheme. https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html.

Miller, J., Waite, D., and Jones, M. B. (2023). JSON Web Proof. https://datatracker.ietf.org/doc/draft-ietf-jose-json-web-proof/.

Pointcheval, D. and Sanders, O. (2018). Reassessing security of randomizable signatures. In *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338.

Sakemi, Y., Kobayashi, T., Saito, T., and Wahby, R. S. (2022). Pairing-friendly curves. https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves/.

Sporny, M. and Longley, D. (2022). Verifiable Credentials data integrity 1.0. https://www.w3.org/TR/vc-data-integrity/.

Sporny, M., Longley, D., and Chadwick, D. (2022). Verifiable credentials data model. https://www.w3.org/TR/vc-data-model/.

Sporny, M., Longley, D., Chadwick, D., Terbu, O., Zagidulin, D., and Zundel, B. (2019). Verifiable credentials implementation guidelines 1.0. https://www.w3.org/TR/vc-imp-guide/.

Steele, O. and Prorock, M. (2021). JSON Web Proof for binary Merkle trees. https://w3c-ccg.github.io/Merkle-Disclosure-2021/jwp/.

Sullivan, B. (2010). Cryptographic agility. In *Black Hat USA*. https://www.blackhat.com/html/bh-us-10/bh-us-10-archives.html#Sullivan.

Tessaro, S. and Zhu, C. (2023). Revisiting BBS signatures. In *Eurocrypt 23 (forthcoming)*. https://ia.cr/2023/275.