The BBS# protocol

Technical details

Date: June 2024

Table des matières

Solving the eIDAS2 conundrum today: BBS style privacy with SOG-IS based hardware security

Summary

eIDAS 2.0 is a very ambitious regulation aimed at equipping European citizens with a digital wallet that not only needs to achieve a high level of security but also needs to be available as soon as possible for a large number of citizens and respect their privacy (as per GDPR). As of today (June 2024), it does not seem that this goal has been achieved in the European Digital Identity Architecture and Reference Framework (ARF). This document aims to propose the foundations of a digital identity wallet solution that could help move closer to this objective by leveraging the proven anonymous credentials protocol BBS (also known as BBS+) but modifying it to avoid the limitations that have hindered its widespread adoption, especially in certified infrastructures requiring hardware implementation. Thus, the solution we propose does not use bilinear pairings and only requires the implementation in hardware of well-known digital signature schemes such as ECDSA or ECSDSA (also known as ECSchnorr) using classical elliptic curves. Additionally, this solution can be implemented while adhering to the ISO mDL and SD-JWT protocol formats, allowing for its use not only in online contexts but also in face-to-face/proximity contexts. Furthermore, this protocol retains the well-known anonymity properties of BBS+, such as *unlinkability* and *everlasting/unconditional privacy*. We also introduce a simple method for *plausible deniability* for transactions related to services that do not require audits. We provide various improvements in terms of implementation efficiency (a single holder binding proof for *verifiable presentations* containing multiple (Q)EAA) and security (impossibility for a *Wallet Secure Cryptographic Device* (WSCD) to generate a proof without the holder's involvement), as well as a highly efficient new method to prove the non-revocation of (Q)EAA used in a transaction. We provide formal security and privacy proofs for our protocol (named BBS#); we believe that BBS# should be deployable quickly on certified infrastructures such as HSMs or SEs and should also simplify the implementation of eIDAS services that rely on the ARF. For example, proofs of non-revocation of all kinds can be done much more simply. Key management can also be greatly simplified, with only a single private key required on users' WSCD. We believe that this proposal fully addresses the challenges posed by the eIDAS 2 regulation by ensuring GDPR's obligation to be state-of-the-art in personal data protection, while maintaining great deployment flexibility for wide and immediate availability and a certain ability to be certified quickly.

Aim of this document: The objective of this document is to present the technical details of the BBS# anonymous credentials protocol, a variant of the BBS protocol (currently being standardized at the IETF), that meets the requirements of the future European digital identity wallet in terms of security and privacy. To illustrate this, we will focus on one of the main use cases of this type of digital identity wallet: *selective disclosure* of attributes. Specifically, we will show how the BBS# protocol preserves the privacy of digital identity wallet holders by allowing them to only disclose to third parties (service providers) the information that is strictly necessary for accessing their services (for example, they can prove that they hold a driver's license, so that they can access sites reserved for adults, without having to reveal their identity or date of birth). As we will see, BBS# is also compatible with the ISO/IEC 18013-5 standard (Mobile driving license application), which is one of the main digital identity management standards considered in the ARF, but it improves it by offering better privacy protection for holders of digital identity credentials (following the ISO/IEC 18013-5 format).

While the propositions of this document can be implemented immediately in today's existing smartphones (we've done it), we reckon that this document does not go into all the details and does not follow the form that could be expected from, e.g. a peer reviewed document in an applied crypto conference (which we do intend to do before the end of the year). Given the ongoing direction taken by the ARF and the associated debates and surrounding initiatives, it seemed important to us to have the word out as soon as possible and allow people to make up their mind with the raw material, thus offering alternative options in the design of privacy preserving eIDAS 2 wallets.

Structure of the document: This technical report is organized as follows. Firstly, we present the two main identity management models, *federated identity* and *self-sovereign identity*, and the privacy challenges that these models raise. We then recall how the ISO/IEC 18013-5 standard works, which is one of the main standards for managing digital identity credentials, and why it inadequately protects the privacy of credential holders. The following sections are dedicated to the BBS# protocol, which aims to address the privacy limitations of the ISO/IEC 18013-5 standard. **In Section 1**, we present the main building blocks of our cryptographic protocol (commitment schemes and zero-knowledge proofs) and recall the basic principles of a self-sovereign identity system (the actors and the main exchanges that can take place between these actors, such as the issuance of a digital identity credential and its presentation to a service provider). **In Section 2**, we describe the functioning of the BBS# protocol (in the specific case of selective disclosure of attributes). **In Section 3**, we prove that our protocol is secure (namely that BBS# credentials are unforgeable) relatively to a well-established computational assumption (the q-SDH assumption) and demonstrate that presentations of verifiable credentials, using the BBS# protocol, are perfectly anonymous (everlasting privacy): for example, in an online survey, it will be impossible for anyone, even with unlimited computational power, to obtain any information about the individuals who answered to the survey (and therefore to identify them); the only information revealed will be that these individuals were part of the chosen panel for the survey and that they answered only once. **In Section 4**, we compare the efficiency (in terms of key size and computation time for credential issuance and presentation) of BBS# with that of ISO mDL/SD-JWT (instantiated with the ECDSA signature scheme) and PQ-ABC, a recent post-quantum anonymous credential scheme that will appear at the forthcoming ACM CCS conference. **In Section 5**, we compare these three protocols (BBS#, ISO mDL with ECDSA, and PQ-ABC) in terms of the level of security and privacy they offer. We conclude this report **in Section 6**.

Introduction : the context

Many use cases of daily life (either online or through face-to-face interactions) require that users provide some of their identity credentials to implement the corresponding business process. Most of the time, in the physical world, that involves showing some documents issued by official authorities (id cards, driving licenses, etc…) or by regulated private businesses (electricity or telecom bills, medical certificates, etc…).

Often, after presenting only the necessary identity elements to access the service, the credential is returned to the user. By selecting the presented identity elements and retaining the (physical) credential, the user retains control over their identity elements and manages their dissemination. The development of services accessible via the internet requires the implementation of dedicated tools for digitizing the elements present on the aforementioned physical credentials, potentially in a structured manner, and allowing their sharing with the service provider after user consent. Major players on the internet have so far structured the domain of digital identity management in line with their business model. For example, through the OIDC (OpenID Connect) protocol, they have developed an offer that centralizes the user's identity data and, under the user's control, after identification/authentication and user consent, authorizes the service (called Relying Party) to access all or part of the user's identity data.. One of the main problems with this type of architecture (known as *Federated Identity*, see Figure 1) is that the *identity provider* knows, each time a user consumes one of the services (Relying Party), when the operation takes place and which identity elements the user shares with the service provider being accessed. The identity provider is thus able to compile the history of the user's access activities and the identity elements required by the various services visited. This data allows them to build a profile of the user's service consumption with service providers and to use it for often commercial purposes.

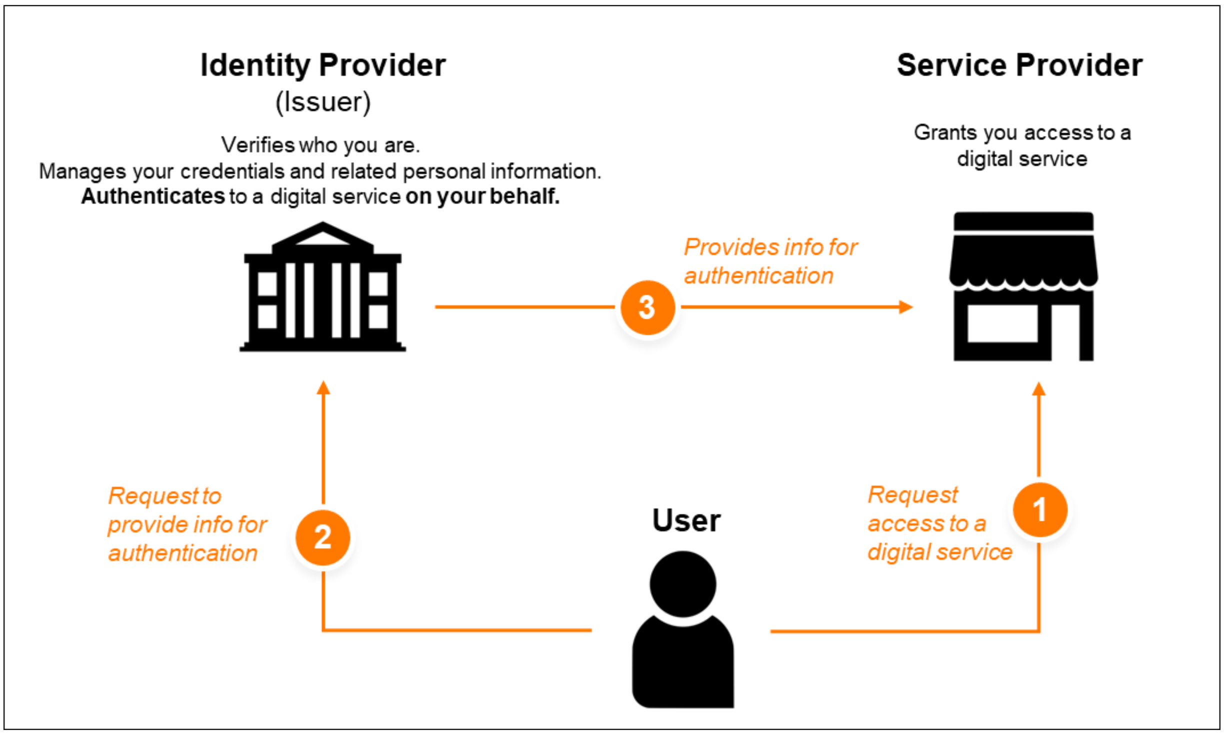

Identity Provider
(Issuer)
Verifies who you are.
Manages your credentials and related personal information.
Authenticates to a digital service **on your behalf.**
Service Provider
Grants you access to a digital service
Provides info for authentication
3
Request to provide info for authentication
2
User
Request access to a digital service
1

Figure 1: Federated identity model

Self-Sovereign Identity (SSI) is another approach to digital identity that gives users full control over their identity elements. Users can present these elements to access a service either to prove who they are or one of their qualities. Only the service provider and the user interact during the identity sharing phase. To generate trust between the user and the service, the user presents identity elements that have been certified by a third party (the identity provider) which the service provider trusts[1]. In this logic, the user holds identity elements about themselves that have been issued and certified[2] by one or more trusted entities, controls their dissemination to service providers (after consent), and protects their privacy from the identity provider by excluding them from the relationship they have with the service provider. The protocol is completely asynchronous.

In this digital identity model, an accreditation is a personal attestation allowing its holder to convince others (such as a service provider) that they have a particular authorization or qualification. Diplomas, student cards, or driving licenses are examples of traditional credentials used in everyday life. They are specific to an individual and issued by a trusted entity (an identity provider) using *digital signatures*. However, standard digital signature mechanisms require full disclosure of the signed/certified data, even to prove the authenticity of only a part of it, and enable users to be traced. Let us take the example of a transport card: to prove the validity of their card to a verifier (e.g. a turnstile), the user will have to provide all the certified information associated with their card, including their card identifier, which unfortunately makes it possible to trace their trips.

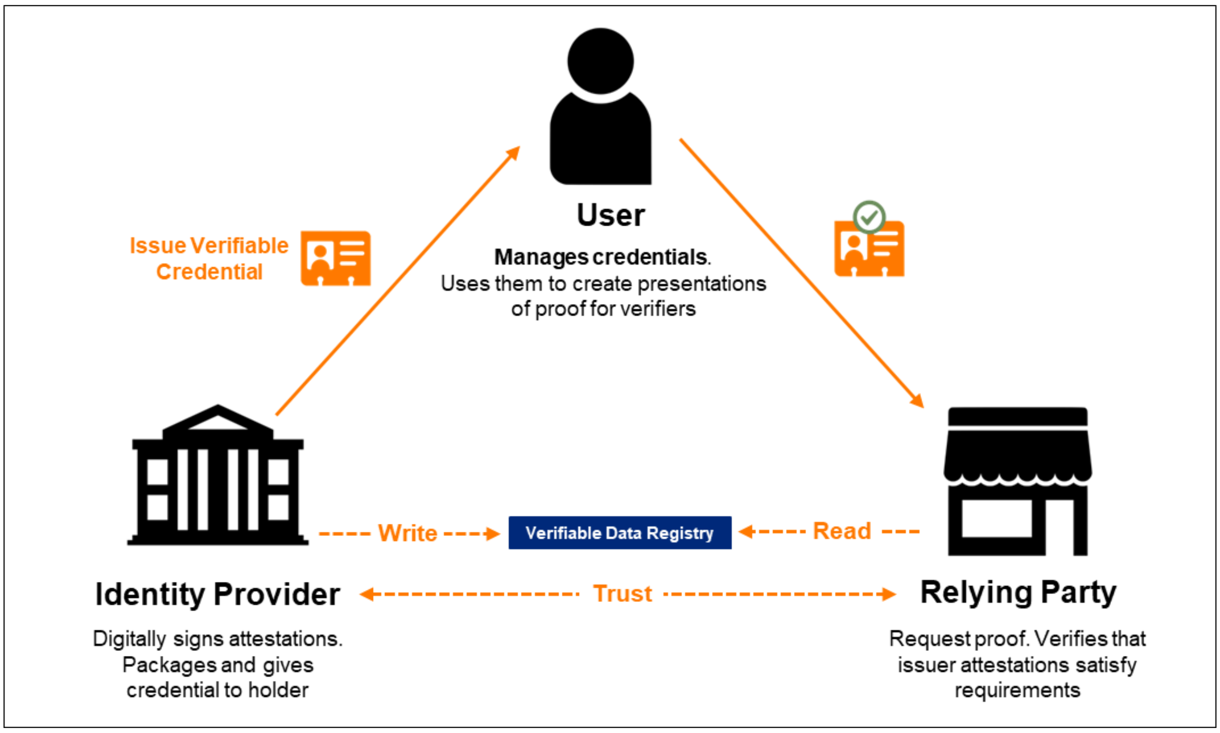

User
Manages credentials.
Uses them to create presentations of proof for verifiers
Issue Verifiable Credential
Identity Provider
Digitally signs attestations. Packages and gives credential to holder
Write
Verifiable Data Registry
Read
Trust
Relying Party
Request proof. Verifies that issuer attestations satisfy requirements

Figure 2: Self-sovereign or decentralized identity model

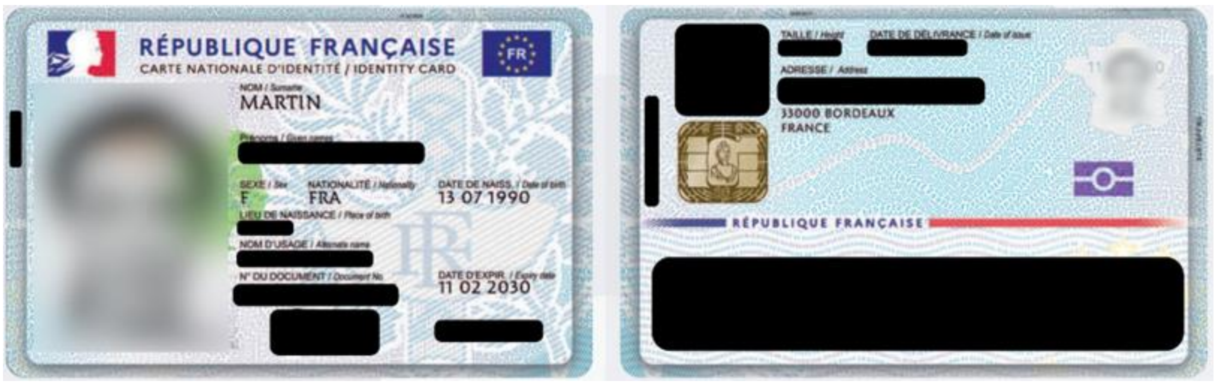
To best protect his or her privacy, a user should be able to reveal to a service provider only the credential data *strictly necessary* for the requested service. This property is known as

[1] In this case, we are referring to "Verifiable Presentation" (VP).

[2] In this case, we are referring to "Verifiable Credential" (VC). In the eIDAS 2.0 terminology, this is referred to as (Q)EEAs.

selective disclosure of attributes. For example, to benefit from a discounted train ticket as a student, a user should be able to prove their student status to the railway company booking website without having to disclose their full identity and/or the specific course they are studying.

Selective disclosure of attributes is easily achieved in the physical world. For example, concealing (or "redacting") information on one's credential with one's hand (or with a black felt-tip pen), leaving only certain identity data visible, is a selective disclosure of attributes. (see Figure 3). The ISO/IEC 18013-5 standard explains how to transpose this manual redaction technique (see Figure 3) to credentials.


RÉPUBLIQUE FRANÇAISE
CARTE NATIONALE D'IDENTITÉ / IDENTITY CARD
NOM / Surname
MARTIN
Prénoms / Given names [redacted]
SEXE / Sex F
NATIONALITÉ / Nationality FRA
DATE DE NAISS. / Date of birth 13 07 1990
LIEU DE NAISSANCE / Place of birth [redacted]
NOM D'USAGE / Alternate name [redacted]
N° DU DOCUMENT / Document No [redacted]
DATE D'EXPIR. / Expiry date 11 02 2030
[redacted] [redacted]

[redacted]
TAILLE / Height [redacted] DATE DE DÉLIVRANCE / Date of issue [redacted]
ADRESSE / Address [redacted]
33000 BORDEAUX
FRANCE
RÉPUBLIQUE FRANÇAISE
[redacted]

Figure 3 : Selective disclosure of identity data

The ISO/IEC 18013-5 standard is an international standard concerned with the design of a mobile driving license (mDL) credential carried in a native application running in a mobile device. It specifically defines an (extensible) data model[3] for such credentials, as well as a protocol enabling the selective disclosure of attributes associated with such a credential.

The working principle of the selective disclosure method used in the ISO/IEC 18013-5[4] standard is as follows (see also Figure 4)[5].

Issuance of a credential; the Identity Provider (IdP) or Issuer has a pair of keys, consisting of a private key (SK_I) and a public key (PK_I), of a digital signature algorithm such as ECDSA. The user also has their own key pair, consisting of a private key (SK) and a public key (PK) of a digital signature algorithm, which they will use to authenticate their VPs. The user must have their key pair and attributes related to their driving license certified by the IdP. We will denote the attributes as $(a_1, a_2, a_3, \dots, a_n)$.

During the issuance phase of such a (verifiable) credential (VC), the Identity Provider (IdP) will first authenticate the user. Then, the IdP will randomly pick n secret values, denoted as $\{K_i\}_{i=1}^{n}$. Next, the IdP will compute n *digests* (cryptographic hashes), one for each attribute. Each of these n digests will be labeled with a unique digest identifier, denoted as HID_i. The digest, denoted as H_i, for each attribute is calculated using its digest identifier (HID_i), attribute identifier (denoted by ID_{a_i}), attribute value (a_i) and the secret key generated and associated with that attribute during the creation of the identity credential: $H_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$

[3] This model can, of course, be extended to any other form of digital identity document on a mobile phone (social security card, national identity card, passport, etc.).
[4] It is also referred to as ISO mDL in the subsequent context.
[5] We will voluntarily use the terminology commonly used in the context of SSI (VC, VP, user, identity provider, service provider, etc.) for this description, rather than the terminology used in the ISO/IEC 18013-5 standard (mDL, mdoc, MSO, etc.).

where $\mathcal{H}$ is a cryptographic hash function (such as SHA-256; the notation $str_1 \parallel str_2$ denotes the concatenation of the strings str_1 and str_2).

The Identity Provider (IdP) then computes a digital signature using its private key SK_I on the data structure comprising the user's public key PK and the n digests H_i. Let's denote $H = PK \parallel H_1 \parallel H_2 \parallel \ldots \parallel H_n$ and $\sigma_{Issuer} = Sign_{SK_I}(H)$ as the signature of the IdP on H.

The IdP then transmits σ_{Issuer} and the keys K_i to the user.

The user's verifiable credential (VC) (or Mobile Security Object - MSO in the terminology of the ISO/IEC 18013-5 standard) consists of their public key PK, the digests $\{H_i\}_{i=1}^{n}$ and the certificate σ_{Issuer} on this data : $VC = (PK, \{H_i\}_{i=1}^{n}, \sigma_{Issuer})$. The secret data associated with this VC are SK and the secret keys $\{K_i\}_{i=1}^{n}$.

Verifiable presentation of a digital identity document with selective disclosure: we will denote by $\mathfrak{D}$, the list of indices of the attributes requested by the RP. For example, $\mathcal{D} = \{1, 5, 7\}$ will mean that the RP wants the user to disclose attributes a_1, a_5, and a_7.

1. The user anonymously connects to the RP's website. The RP sends them the access conditions for their site (i.e., the list $\mathcal{D}$ of required attributes) along with a random value (nonce), specific to this session, to prevent (VP) replay attacks.
2. The user calculates a signature (σ_{User}) using their private key SK on a set of data called "DeviceAuthenticationBytes" in the ISO/IEC 18013-5 standard, denoted as m_{DAB} in the remainder of this document. The DeviceAuthenticationBytes[6] includes the nonce (or any other equivalent element specific to the current session with the RP to prevent VP replay attacks), optionally the set of data disclosed to the RP, and other contextual data: $\sigma_{User} = Sign_{SK}(m_{DAB})$.

The VP consists of the public key PK_I of the IdP, the public key PK of the user, the digests $\{H_i\}_{i=1}^{n}$, the set of data disclosed to the RP ($\mathcal{D}_{disclosed} = \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}$), and the signatures σ_{Issuer} and σ_{User} : $VP = (PK_I, PK, \{H_i\}_{i=1}^{n}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}, \sigma_{Issuer}, \sigma_{User})$.

3. The user transmits the VP to the RP.

The RP first calculates $H'_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ for each $i \in \mathcal{D}$.
It then verifies that:

1) $H'_i = H_i$ for each $i \in \mathcal{D}$;
2) σ_{Issuer} is a valid signature on $H = PK \parallel H_1 \parallel H_2 \parallel \ldots \parallel H_n$ using the public key PK_I ;
3) σ_{User} is a valid signature on m_{DAB} (which the RP can reconstruct) using the public key PK certified by the issuer.

If all these conditions are met, the RP grants access to its site.

***Note* 1**: the signature σ_{Issuer} authenticates all the data disclosed by the user ($\{a_i\}_{i \in \mathcal{D}}$)) as well as the public key PK. The signature σ_{User}, on the other hand, convinces the RP that this VP originates from the holder of the $VC = (PK, \{H_i\}_{i=1}^{n}, \sigma_{Issuer})$ and is not a replay of a previous VP.

Note 2: the secret keys (K_i), included in the computation of the digests H_i, are used to prevent exhaustive search attacks aimed at recovering the undisclosed attributes. Without these secret keys, brute-force attacks would easily succeed against attributes with low entropy, such as those related to the user's "date of birth" or "identity document serial number". Indeed, if H_i

[6] The ISO/IEC 18013-5 standard is not very explicit about the content of "DeviceAuthenticationBytes".

had been computed as follows: $H_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i)$, without the secret key K_i, it would be easy to retrieve the attribute "date of birth" (a_i) by testing all possible values (a'_i) until obtaining the correct one, i.e., the one that satisfies the equality $H_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a'_i)$ (where HID_i and ID_{a_i} are public data accessible to everyone).

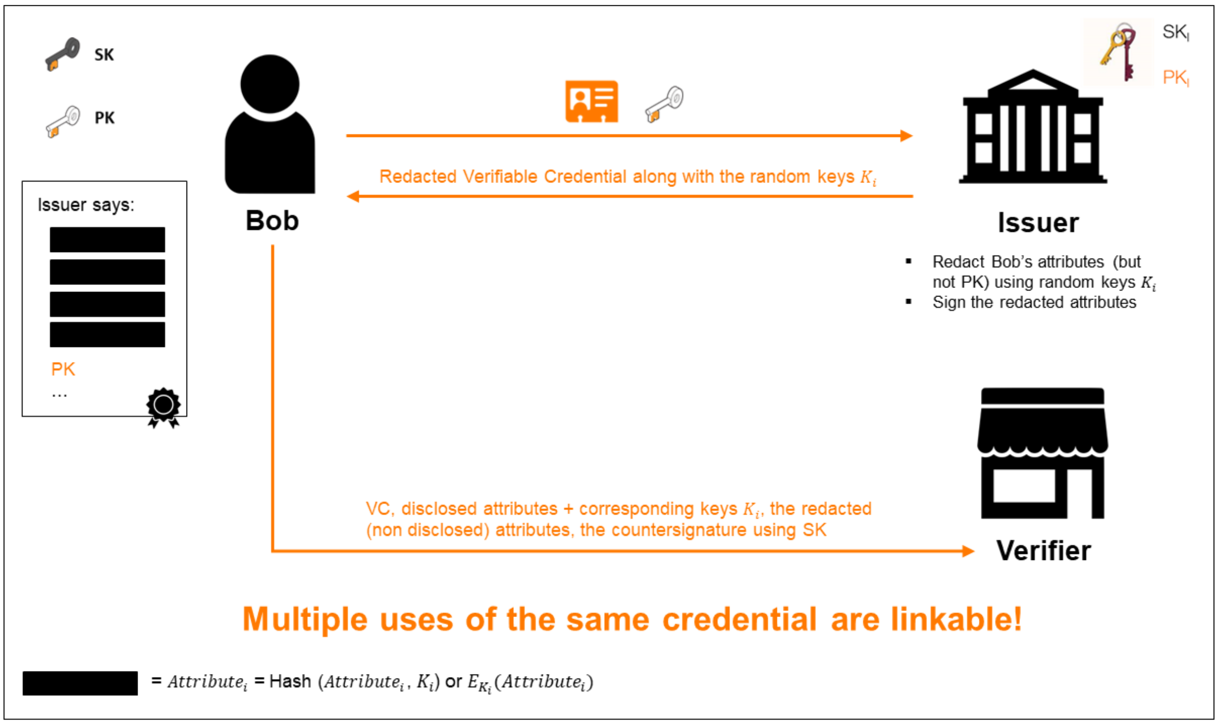

SK
PK
Issuer says:
PK
...
Bob
Redacted Verifiable Credential along with the random keys K_i
SK$_I$
PK$_I$
Issuer
Redact Bob's attributes (but not PK) using random keys K_i
Sign the redacted attributes
Verifier
VC, disclosed attributes + corresponding keys K_i, the redacted (non disclosed) attributes, the countersignature using SK
Multiple uses of the same credential are linkable!
= $Attribute_i$ = Hash ($Attribute_i$, K_i) or $E_{K_i}(Attribute_i)$

Figure 4: Selective disclosure of *attributes*

Drawback 1: From a data protection perspective, this solution has a significant flaw: the user is traceable by the RP (and also by the IdP). Indeed, all VPs of this user will systematically contain the signature σ_{Issuer} and the public key PK, indicating that they have been produced by a single user. Worse still, if the RP and the IdP collude, they can certainly identify who produced a given VP (indeed the IdP knows to whom it issued the signature σ_{Issuer} and can therefore identify the author of any VP).

Drawback 2: the user's private key SK, which is used to authenticate their VPs (via the signature σ_{User}), is an extremely sensitive key that must be properly protected; indeed, the compromise of this key would risk the user's identity being usurped. ISO/IEC 18013-5 recommends that this key be stored in a "secure hardware element with appropriate certifications" for this purpose. However, such devices (Secure Element or SE) have limited computational power and do not support all the mathematical operations used by certain classical digital signature algorithms (especially those, like BBS+, involving *bilinear* pairings in the generation of a signature and/or the verification of a signature). **This constraint therefore strongly limits the type of digital signature algorithms that can be used to issue the signatures $\boldsymbol{\sigma_{User}}$ and/or verify the signatures $\boldsymbol{\sigma_{Issuer}}$.**

1. Foundations of the BBS# protocol

We propose a solution to these two drawbacks (traceability, by the IdP and the RP, of the signature issued by the IdP and the user's public key), while remaining compliant with the recent ISO/IEC 18013-5[7] standard. In other words, the user will be able to reveal to a service provider only the attributes strictly necessary to access the services the latter offers, but without being able to be traced by the IdP and/or the RP from their VPs. The selective disclosure method we use is fully compliant with that described in ISO/IEC 18013-5 (and briefly described above).

1.1 Cryptographic tools

For our protocol, we will rely on a set of well-proven cryptographic techniques: an additive homomorphic *commitment scheme* and zero-knowledge proofs (ZKP). We will briefly explain these concepts, trying to minimize as much as possible the underlying mathematical formalism. Readers familiar with these concepts can skip this section and proceed to the next one.

1.1.1 Commitment schemes

The commitment of a value is a cryptographic process that allows a sender (Alice) to commit to a value (such as a secret) to a recipient (Bob) without initially revealing it, ensuring that this commitment cannot be changed afterwards. The value can be revealed by the sender at a later stage if necessary.

Thus, the recipient can be assured that once the commitment is made, the sender can no longer change their mind about the committed value. To illustrate this, a commitment can be seen as a closed safe, containing the secret, that is given to Bob. Later Alice can provide Bob with the key to open it.

For example, a simple way to build a commitment scheme is to use a hash function H. It is important to note that such a function should be *collision-resistant*, meaning it should be infeasible (in practice) to find two messages m and m' such that $H(m) = H(m')$. Additionally, the function should be computationally difficult to *invert*: given a hash C, it should be difficult to find a message m such that $C = H(m)$.

To commit to a value x (such as the winning combination for the next horse race or lottery draw) with Bob, Alice chooses a random value r and calculates the commitment $C = H(x||r)$, which she sends to Bob.[8]. By doing so, Alice can no longer change her mind. Otherwise, it would mean she could find another value $x' \neq x$ (along with another random value r') such that $C = H(x'||r')$, which would contradict the assumption that the hash function H is collision-resistant.

[7] In particular, we adhere to the data model and the protocols for issuing VCs and selective disclosure as defined in this standard. Additionally, the cryptographic tools employed in our solution involve only classical mathematical operations such as point addition or scalar multiplications on elliptic curves, which are efficient enough to be executed by constrained devices such as secure elements (smart cards) as recommended by the ISO/IEC 18013-5 standard.

[8] The symbol || refers to the concatenation symbol.

Furthermore, Bob who was given C is unable to recover the value that has been committed. Otherwise, it would mean that he can *invert* the hash function H.[9]

At the end of the race, Alice can prove to Bob that she had the trifecta r by revealing the pair (x, r). Bob can check that Alice did not lie by verifying that $C = H(x || r)$.

For our protocol, we will use the commitment scheme proposed by Pedersen in 1992 [Ped92]. This scheme provides commitments that are *perfectly hiding*[10] and *computationally binding*[11].

1.1.2 Zero Knowledge Proofs

A *Zero-Knowledge Proof* (ZKP) [GMR85] is an *interactive protocol* that allows a verifier to be convinced that a certain prover knows a secret S that satisfies a given predicate P - for example, the factorization (p, q) of a very large public number N which is the product of these two prime numbers p and q $(N = p * q)$. Furthermore, the proof reveals no information about the secret S to the verifier (the factorization of N in our example), except for the fact that it satisfies the given predicate P. In the sequel, to represent Zero-Knowledge Proofs, we will sometimes use the standard notation introduced by Camenisch and Stadler [CS97]: PoK{α,β,· · · : predicate on α,β,· · · }, where the Greek letters correspond to the secrets known by the prover. For example, PoK{α : $y = g^{\alpha}$} will denote a proof of knowledge of the discrete logarithm of (the public value) y in the base (public value) g. In our constructions, we will rather use the non-interactive versions of Zero-Knowledge Proofs (also called Signatures of Knowledge”), obtained through heuristic transformations such as the one due to Fiat-Shamir [FS86]. More precisely, a ”Signature of Knowledge” is a non-interactive protocol that allows a prover to convince a third party that they know a secret S that satisfies a predicate P, and which also guarantees the authenticity and integrity of a message m. We denote such a Signature of Knowledge with secret values (α,β,γ,..) that satisfy a given predicate and authenticate a message m as follows SoK{α,β, γ,· · · : predicate on α,β, γ,· · · }(m), where the Greek letters correspond to the secrets known by the prover and m is the authenticated message. For example, SoK{α : $y = g^{\alpha}$}(m) will denote a signature of knowledge of the discrete logarithm of (the public value) y in the base (public value) g, authenticating the message m. A signature of knowledge on a message m has the same characteristics as a digital signature: it guarantees both the authenticity and integrity of the message m.

1.2 Stakeholders presentation

In an architecture following the self-sovereign digital identity model, several entities are involved:

- Identity providers (sometimes called "Issuers")
- Service providers (also known as Relying Party or RP or verifier)
- Users

[9] The use of the random value r is crucial. Given that there is a limited number of possible combinations, Bob could have found the winning combination x by a brute-force attack if Alice had simply sent him $C = H(x)$ instead of $C = H(x, r)$.

[10] An attacker in possession of a commitment C will not learn any information about the secret value (the winning combination for the next horse race) that was committed, even if they have unlimited computational power.

[11] Given a commitment C of a message m, it is hard to find another message m' such that C is also a commitment to the message m'.

1.2.1 The identity provider

Its role is to issue VCs to users, containing their attributes, after authenticating them. Users can then present these VCs when accessing a service to prove who they are or one of their qualities. In the following, the IdP will use the BBS+ protocol (described later) to certify the identity elements (attributes) of users, instead of a more classical digital signature scheme.

1.2.2 The service provider / RP

It delivers services to users after ensuring that they meet the conditions for accessing its services. For example, to access a web site with adult content, the service provider has a legal obligation to verify that the user is of legal age. The user must therefore provide proof of age by means of a Verifiable Presentation (VP) generated from a Verifiable Credential (VC), certifying their age, issued by an Identity Provider (IdP) that the Service Provider (SP) trusts (such as the State agency in charge of issuing official documents for a driving license).

1.2.3 The users

They have a set of VCs that allow them to prove their identity or qualities (such as majority, seniority, degrees, etc.) to service providers in order to access the services they offer. They have a digital identity wallet (ID Wallet) that allows them to store and manage their various VCs and generate VPs for access to SP services. Their private keys and secrets (which are used to prove ownership of the VCs they present) are stored (ideally) in secure devices such as Secure Elements, HSMs, or TEEs.

1.3 Main phases

We will consider the following three main phases in our digital identity system:

- Issuance of VCs
- Access to a service provider and generation of a VP taking into account selective attribute disclosure
- Verification of a VP

1.3.1 VCs issuance

This is an interactive protocol between an identity provider and a user. After authenticating the user, the identity provider issues a VC (a signed credential in ISO/IEC 18013-5 format) containing the user's identity elements, whether they are official or non-official. With such a VC, the user can, if necessary, convince a service provider that they meet all the conditions to access the services offered by the latter (for example, proving that they are of legal age to access an adult content website, without revealing their date of birth or any other identifying information).

1.3.2 Access to the RP and VP generation

This is an interactive protocol between a service provider and a user. The user who wishes to access the services offered by a service provider receives its access policy. This policy may include specific attributes such as the user's city of residence for a citizen consultation, their student status to benefit from a preferential rate, or their date of birth or proof of legal age for access to an online betting site. If the user meets the SP’s access policy, they select one or more of their VCs (in ISO/IEC 18013-5 format) attesting to this (for example, their driving license for a proof of majority) and instructs their Wallet to generate a VP (also in ISO/IEC 18013-5 format), based on one or more of their VCs, demonstrating that they indeed meet the access policy of the service provider. Of course, in compliance with GDPR, a VP only reveals the information requested by the service provider and nothing more. In the case of access to an adult content website, for example, the service provider will only know that the user is of legal age; it will not know their date of birth or any other information that could be used to identify or trace them.

1.3.3 VP verification

The process of verifying a VP is the same as described in the ISO/IEC 18013-5 standard.

2. BBS# and selective disclosure : compliance with the ISO mDL standard

In the following sections of this document, we will describe the procedures for issuing a VC and generating a VP, in accordance with the ISO/IEC 18013-5 standard, based on a variant (which we introduce) of the BBS+ protocol [BBS24][12]. **The BBS+ protocol has two drawbacks for being used as is within the context of the ISO/IEC 18013-5 standard.** The first drawback is that in order to verify the validity of a VC issued by the Identity Provider, the user's device must be able to handle bilinear pairings (which unfortunately are not supported by current Secure Elements). The second drawback is that the computation time for generating a VP with BBS+ is linear in the number of attributes (n) of the underlying VC. Specifically, the user's device needs to compute a linear number of scalar multiplications on elliptic curves. However, such operations (scalar multiplications on elliptic curves) are extremely time-consuming for devices such as Secure Elements. Therefore, we will modify the BBS+ protocol, without compromising its security or privacy properties, so that the user's device (Secure Element) does not need to compute any bilinear pairings and only needs to perform a constant number of scalar multiplications on elliptic curves (specifically, only one scalar multiplication).

The various proofs of knowledge that will be implemented during the generation of a VP are relatively standard[13] and for this reason will not be detailed in this document.

2.1 Cryptographic preliminaries

In this section, we introduce our notations and provide a brief description of the cryptographic building blocks that are at the core of our BBS+ construction (such as the Pedersen commitment scheme [Ped92]), as well as the main computational assumption that underlies the security of our cryptographic protocol.

2.1.1 Notations

In the following, we will use the notation $\mathrm{PoK}(\alpha_1, \alpha_2,\dots, \alpha_n : \mathcal{R}(\alpha_1, \alpha_2,\dots, \alpha_n))$ to denote a Zero-Knowledge Proof of Knowledge (ZKPK) of elements $\alpha_1, \alpha_2,\dots, \alpha_n$ satisfying the relation $\mathcal{R}$. For example, a proof of knowledge of the two prime factors of a public RSA modulus N would be denoted as : $\mathrm{PoK}(\alpha_1, \alpha_2 : N = \alpha_1 \cdot \alpha_2 \wedge (\alpha_1 \neq 1) \wedge (\alpha_2 \neq 1))$. Similarly, we will use the notation $\mathrm{SoK}(\alpha_1, \alpha_2,\dots, \alpha_n : \mathcal{R}(\alpha_1, \alpha_2,\dots, \alpha_n))(m)$ to denote a signature of knowledge (ZKPK) of elements $\alpha_1, \alpha_2,\dots, \alpha_n$ satisfying the relation $\mathcal{R}$ and authenticating the message m. Z_p denotes the set {0, 1, 2,…, p-1}, where p is a prime integer and Z_p^* the set {1, 2,…, p-1}.

[12] This protocol, initially named BBS+, is now referred to as BBS, which could potentially cause confusion. The BBS algorithm typically refers to the group signature scheme introduced by Boneh, Boyen, and Shacham at Crypto 2004, while BBS+ refers to its generalization to an anonymous credential system, introduced by Au, Susilo, and Mu in 2006 [ASM06]. BBS+ has since then been improved in terms of performance in 2016 [BBDT16, CDL16], and its security has been further strengthened in 2023 [TZ23].

[13] The article [Bra97] is indeed an excellent reference for building such proofs.

2.1.2 Bilinear maps

Bilinear groups[14], are a set of three cyclic groups[15] G_1, G_2 and $G_T$of order p, with p being a prime numberalong with a bilinear map, called e: $G_1 \times G_2 \rightarrow G_T$ satisfying the following three properties :

1. (Bilinearity) $\forall\, g_1 \in G_1, \forall\, g_2 \in G_2$ and $\forall\, (a,b) \in Z_p^2$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. (Non-degeneracy) For $g_1 \neq 1_{G_1}$ and $g_2 \neq 1_{G_2}$, $e(g_1, g_2) \neq 1_{G_T}$.
3. (Computability) $\forall\, g_1 \in G_1$, $\forall\, g_2 \in G_2$, there exists an efficient algorithm to compute $e(g_1, g_2)$.

In practice, the groups G_1, G_2 and G_T will be chosen such that there is no efficiently computable homomorphism between G_1 and G_2.[16]

2.1.3 Computational hardness assumptions

The security of the cryptographic mechanisms we use relies in part on the assumption that the following problem is hard. In other words, if an attacker is able to compromise the security of these mechanisms, it means that they are also capable of solving this problem, which is known to be "hard".

q-SDH problem. Given a cyclic group G of prime order p and an element $x \in Z_p$, the *q-strong Diffie-Hellman* (q-SDH) problem consists of computing a pair of the form $(g^{1/_{x+c}}, c) \in G \times Z_p$ given $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$.

q-DL problem. Given a cyclic group G of prime order p and an element $x \in Z_p$, , the *q-Discrete Logarithm* (*q-DL*) problem consists of finding x given $(g, g^x, g^{x^2}, g^{x^3}, \ldots, g^{x^q})$.

The q-SDH problem was introduced by Boneh, Boyen, and Sacham to prove the security of their group signature scheme BBS [BBS04]. The difficulty of this mathematical problem has been further studied by Cheon [Che06] and Jao and Yoshida [JY09]. Moreover, the q-SDH assumption trivially implies the slightly more standard q-DL assumption. The converse is not known to be true in general, but it is true for *algebraic adversaries* [BFL20].

2.1.4 The Pedersen commitment scheme

The most commonly used commitment scheme is the one introduced by Pedersen [Ped92]. Below, we provide a brief description of this scheme.

Let G be a cyclic group of prime order p and let g and h be two arbitrary random generators of G. We assume that the discrete logarithm of h to the base g is unknown.

Commitment: To commit to a chosen message $m \in Z_p$, the sender selects a random value $r \in Z_p$ and calculates $C = g^m h^r$, then sends it to the recipient.

[14] In fact, this document describes the two versions of the BBS# protocol (i.e., with or without pairings). Current SE/HSM/TEE devices do not support such mathematical operations nor the associated "pairing-friendly" curves, so the version without pairings was preferred for our various implementations on WSCD (SE/TEE) of the BBS# protocol (see section 4). This version without pairings relies on the protocol called MAC_BBS introduced by Barki et al. at the SAC 2016 conference [BBDT16].

[15] For convenience, we will assume that the group law in G_1 and G_2 is multiplication.

[16] Such pairings are known as Type 3 pairings in the literature.

Opening: The sender sends the values m and r to the recipient, who verifies the equality: $C = g^m h^r$.

In the sequel, we will rather use in fact the generalization of this scheme to several committed values (see figure 5, for the computation of the value A)

2.1.5 Issuance of VCs with BBS+:

First, the identity provider defines the public parameters of the BBS+ system: $pp = (G, G_2, G_T, e, p, n, g, g_0, g_1, g_2, .., g_n, \tilde{g}, f)$ where

- G is a cyclic group, and we assume that the q-SDH problem is hard in this group.
- p is the order of the group G (a prime integer)
- n is the number of attributes to be certified.
- $g, g_0, g_1, g_2, .., g_n, \tilde{g}$ are n+3 randomly chosen (**in a verifiable way**) generators of G (**such that no one knows the discrete logarithm of one generator with respect to another**). Each $\{g_i\}_{i=1}^n$ is associated with a specific attribute type (e.g., g_1 is associated with the "name" attribute, g_2 with the « first name » attribute, g_3 with the « age » attribute, g_4 with the « gender » attribute, etc.). This allows us to differentiate attributes and avoid any ambiguity.
- f is a generator of G_2

In the following, all computations involving exponents will be performed modulo p (mod p).

Keys generation :

The identity provider (IdP) randomly generates an integer sk_I from the set {1, 2,…, p-1} and computes $PK_I = \tilde{g}^{sk_I}$ and $PK'_I = f^{sk_I}$ (along with a zero-knowledge proof Π that its key pair (PK_I, PK'_I) has been correctly computed : Π = PoK $\{\alpha: PK_I = \tilde{g}^{\alpha} \wedge PK'_I = f^{\alpha}\}$[17]. Its private key will be sk_I and its public key (PK_I, PK'_I).

We assume that each user *U* also has a pair of keys, private and public, generated and managed exclusively by their WSCD: $(sk, PK = g^{sk})$.

We also assume that the IdP has published n public values[18], randomly chosen from Z_p and denoted $\{K_i\}_{i=1}^n$ as well as another integer (also public), randomly selected from Z_p and denoted as $\mathfrak{U}d$ (for « Undisclosed ») in the following.

To obtain a VC on its attributes $(a_1, a_2, a_3, \ldots, a_n)$ known to the IdP and on its public key PK, the user *U* and the IdP must execute the following protocol (Figure 5), after *U*'s authentication by the IdP (or at least their WSCD). **For security reasons, the public key PK transmitted to the IdP must be authenticated as coming from a WSCD (typically through a *signed attestation*)[19].**

First, the IdP will compute n *digests* (cryptographic hashes), one for each attribute. Each of these n digests will be labeled with a unique digest identifier denoted as HID_i. The digest,

[17] The groups G_2 and G_T, the bilinear map e, the generator f, the key PK'_I, as well as the proof Π are obviously not necessary in the version of BBS# without pairings

[18] These public values will be used for all VC issuances carried out by this IdP.

[19] **For obvious security reasons, the corresponding private key sk will be fully managed by the WSCD and known only to it. It will be unknown to the user and their Wallet application.**

denoted H_i, is computed for each attribute using its digest identifier (HID_i), the attribute identifier (denoted as ID_{a_i}), the value of the attribute (a_i) and the secret key generated and associated with the attribute during the creation of the identity credential[20]: $H_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ where $\mathcal{H}$ denotes a cryptographic hash function producing digests in Z_p (such as SHA-256, for example).

The IdP then computes a BBS+ digital signature using its private key SK_I, on the data structure consisting of the user's public key PK and the n digests H_i. Let $H = PK \parallel H_1 \parallel H_2 \parallel \dots \parallel H_n$ and $\sigma_{Issuer} = Sign_{SK_I}^{BBS+}(H)$ be the IdP's signature on H.

The IdP then transmits σ_{Issuer} to the user. The user's digital identity credential (VC) (or Mobile Security Object - MSO in the terminology of ISO/IEC 18013-5) consists of their public key PK, the digests $\{H_i\}_{i=1}^{n}$ and the certificate σ_{Issuer} on this data : $VC = (\,PK, \{H_i\}_{i=1}^{n}, \sigma_{Issuer})$. The secret data associated with this VC is sk.

Note 3 : In the following, we will assume that the private key sk is stored and managed exclusively by the user's Secure Element (e.g., their SIM card) and that the attributes $(a_1, a_2, a_3, \dots, a_n)$ as well as the $VC = (\,PK, \{H_i\}_{i=1}^{n}, \sigma_{Issuer})$ are stored and managed by a dedicated native application on their mobile phone.

[20] The ISO/IEC 18013-5 standard requires this representation of attributes. It is understood that our solution would also apply to any other representation of attributes.

U		IdP
Public input: pp, $PK_I, PK'_I, \mathfrak{Ad}, \{K_i\}_{i=1}^{n}$, $\{HID_i\}_{i=1}^{n}$, $\{ID_{a_i}\}_{i=1}^{n}$ **Private input**: sk		**Public input**: pp, PK_I, $PK'_I, a_1, a_2, a_3, \dots, a_n$, $\mathfrak{Ad}$, $\{K_i\}_{i=1}^{n}$, $\{HID_i\}_{i=1}^{n}$, $\{ID_{a_i}\}_{i=1}^{n}$ **Private input**: sk_I
	$\xleftarrow{ch}$	
Computes : • π_U= SoK $\{\alpha\colon PK = g^{\alpha}\}(ch)$[21]		
	$\xrightarrow{PK,\pi_{U,Attestation}}$	
		Verifies the proof π_U (terminates the protocol if it is invalid) Randomly chooses : $e \in Z_p^*$ Computes $H_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ for i in [1, n], then : $A = (g_0 PK \prod_{i=1}^{n} g_i^{H_i})^{\frac{1}{sk_I+e}}$ % Let $B = g_0 PK \prod_{i=1}^{n} g_i^{H_i}$ % this implies that $A^{sk_I+e} = B$ % and then that $A^{sk_I} = BA^{-e}$ % Let $C = A^{sk_I} = BA^{-e}$ IdP generates a proof Π_I = PoK $\{\alpha\colon C = A^{\alpha} \wedge PK_I = \tilde{g}^{\alpha}\}$.
	$\xleftarrow{A,e,\pi_I}$	
Computes • $B = g_0 PK \prod_{i=1}^{n} g_i^{H_i}$ • $C = BA^{-e} = A^{sk_I}$ Verifies the proof Π_I If the proof is valid, records in their wallet their VC signature (A, e) on their		

[21] The π_U proof is a proof of knowledge of the discrete logarithm of PK in the base g (i.e., the private key sk). The algorithm called ECSchnorr or ECDSA [ISO14888] could be judiciously used for this proof. However, any other algorithm that can prove knowledge of the private key sk could also be implemented. In the context of SSI, it is generally the ECDSA algorithm [ISO14888] that is used for the π_U proof, although strictly speaking, an ECDSA signature does not constitute a proof of knowledge of the signing private key.

attributes $a_1, \dots, a_n$ and their public key PK.[22]
(A, e) is actually a BBS+ signature [BBS24] on PK and $\{H_i\}_{i=1}^{n}$

Figure 5 : BBS+ VC issuance

2.1.6 Verifiable presentation of a digital identity credential in BBS+ format with selective attribute disclosure following the ISO mDL standard.

In the following, we will denote by $\mathfrak{D}$ the list of indices of the attributes requested by the RP. For example, $\mathcal{D} = \{1, 5, 7\}$ will mean that the RP wants the user to reveal the attributes a_1, a_5, and a_7. For each attribute not belonging to $\mathcal{D}$, the user will send the value $\mathfrak{A}d$ to the RP.

- The user anonymously connects to the RP's website. The RP sends to the user its access policy (i.e. the list $\mathcal{D}$ of required attributes) as well as a specific random value (nonce) for this session to prevent a replay attack of a VP.
- The user will first "anonymize" their public key (so that neither the issuer nor the RP can trace them from this key). To do this, they will randomly pick an integer r in Z_p and compute using this value: $PK_{Blind} = g^{sk+r}$. They will then "anonymize" the BBS+ signature of their VC, to also prevent the issuer and the RP from tracing them from this element, and "adapt" it to be on PK_{Blind} and the $\{H_i\}_{i=1}^{n}$ (and not on PK and the $\{H_i\}_{i=1}^{n}$)[23]. To do this, they will choose integers $r_1, r_2 \in Z_p^*$ and compute :
  - $\hat{A} = A^{r_1 \times r_2}$
  - $D = B^{r_2}$
  - $\bar{B} = \hat{A}^{-e} D^{r_1} = \hat{A}^{sk_I}$
  - $r_3 = r_2^{-1} (\text{mod } p)$
  - $\Pi_{validity}$= PoK$\{\alpha, \beta, \gamma, \delta, \{\tau_i\}_{i \notin \mathcal{D}}$: $\bar{B} = \hat{A}^{-\alpha} D^{\beta} \wedge g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{\gamma} \prod_{i \notin \mathcal{D}} g_i^{-\tau_i} g^{\delta}\}$.[24] This ZKP ($\Pi_{validity}$) being relatively standard, we will not detail it in this document.
- Let $\sigma_{Issuer}^{Blind} = (\hat{A}, \bar{B}, D, \Pi_{validity})$

[22] The particularity of this proof Π_I is that it can be verified by constrained devices such as Secure Elements (SE) because it only involves operations in the group G (which are supported by these devices), unlike the proofs proposed by the state of the art that require pairing calculations which are not supported by such devices.

[23] This is possible with BBS+ signatures without compromising the security (unforgeability) of such signatures (see section 3).

[24] We have the following two equalities, hence the ZKP $\Pi_{validity}$: $\bar{B} = \hat{A}^{-e} D^{r_1}$ and $g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{r_3} \prod_{i \notin \mathcal{D}} g_i^{-H_i} g^{r}$

Plausible Deniability: for use cases that allow it, instead of using a standard ZKP ($\Pi_{validity}$), we could use a "Designated Verifier Proof" (DVP) [JSI96] to provide the user with the ability to deny having generated a given VP (Plausible Deniability). A DVP, as the name suggests, is a zero-knowledge proof generated for a designated verifier chosen in advance (the RP in our case). They will be the only one able to verify and be convinced by this particular proof. For everyone else, obtaining this proof is useless and provides no information.

In our context, it would suffice to replace the proof $\Pi_{validity}$= PoK$\{\alpha, \beta, \gamma, \delta, \{\tau_i\}_{i \notin \mathcal{D}}: \bar{B} = \hat{A}^{-\alpha} D^{\beta} \wedge g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{\gamma} \prod_{i \notin \mathcal{D}} g_i^{-\tau_i} g^{\delta}\}$ with the following DVP: $\mathrm{DVP}_{validity}$= PoK$\{\alpha, \beta, \gamma, \delta, \mu, \{\tau_i\}_{i \notin \mathcal{D}}: \bar{B} = \hat{A}^{-\alpha} D^{\beta} \wedge g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{\gamma} \prod_{i \notin \mathcal{D}} g_i^{-\tau_i} g^{\delta} \vee PK_{RP} = g^{\mu}\}$ where PK_{RP} would represent the certified public key of the RP (for which only it would normally know the corresponding private key). $\mathrm{DVP}_{validity}$ is therefore a proof of knowledge of secrets $\{\alpha, \beta, \gamma, \delta, \mu, \{\tau_i\}_{i \notin \mathcal{D}}\}$ satisfying the predicate $P = P_1 \vee P_2$ where P_1= $\Pi_{validity}$ and $P_2 =$ PoK$\{\mu: PK_{RP} = g^{\mu}\}$. The RP will be convinced that if $\mathrm{DVP}_{validity}$ is valid, it is necessarily because P_1 is true (since the RP is assumed to be the only one who knows the private key associated with PK_{RP}, P_2 cannot therefore be true). On the other hand, it will not be able to convince others that P_1 is true. Indeed, as the RP can itself generate a proof that P_2 is true, it could be that $\mathrm{DVP}_{validity}$ is valid, simply because P_2 is true.

Similarly, the IdP could, when issuing the user's VC, replace the proof Π_I (see figure 5) with a DVP (in this case, the "designated verifier" would be the user). By doing so, the user would no longer have the ability to convince others of the validity of their VC (only the IdP would be able to do so). This property (deniability of a VC) is essential for designing "coercion-resistant" online voting systems (see for example [ABBT16]); that is, to thwart and render useless any attempt to buy votes or coerce a voter.)

- The user computes, using the random value r and its private key sk, a signature of knowledge (σ_{User}) on a set of data, referred to as « DeviceAuthenticationBytes »[25] in the ISO/IEC 18013-5 standard and denoted m_{DAB} in the following: $\sigma_{User} = SoK\{\alpha: PK_{Blind} = g^{\alpha}\}(m_{DAB})$. The signature σ_{User} is a *signature of knowledge* of the discrete logarithm of PK_{Blind} in the base g (i.e. of the private key, $sk + r$). The algorithm called ECSchnorr could therefore be judiciously used for this signature of knowledge. However, any other algorithm that allows proving knowledge of the private key sk could also be implemented. In the context of SSI, the ECDSA algorithm is generally used for such a signature (σ_{User}), although, strictly speaking, ECDSA is not a proof of knowledge of the private signing key.

Note 4: In practice, Secure Elements (SE) are relatively closed devices. While most of them support common digital signature algorithms like ECDSA, developers do not have the ability to implement new cryptographic functionalities for security reasons. Therefore, it is difficult to use these SEs for purposes other than what they were initially designed for (such as generating ECDSA signatures, for example). As a result, an SE cannot "anonymize" its own public and private keys because it has not been programmed to perform such operations. It cannot carry

[25] The DeviceAuthenticationBytes includes the nonce (or any other equivalent element specific to the current session with the RP, which helps prevent replay attacks of VPs), possibly the set of data disclosed to the RP, and other contextual data. However, the ISO/IEC 18013-5 standard is not very explicit about the exact content of the "DeviceAuthenticationBytes".

out these basic operations, even though they may seem straightforward : generate a random value r and calculate sk_{Blind} = $sk + r\ (mod\ p)$, its anonymized private key. Similarly, an SE cannot generate the signature σ_{User}, i.e., a signature of knowledge of the discrete logarithm of PK_{Blind} in the base g (i.e. of the private key, $sk + r\ (mod\ p)$).

Below, we explain how the Secure Element (SE) and the associated mobile application (referred to as M-Wallet) can jointly anonymize the public key PK_{Blind} and calculate the signature σ_{User}. It is important to note that only the SE knows the private key sk corresponding to the user's public key PK.

We propose two variants of BBS#: in the first one, we assume that the digital signature algorithm supported by the SE (Secure Element) is ECSchnorr, while in the second one, we assume it is ECDSA.

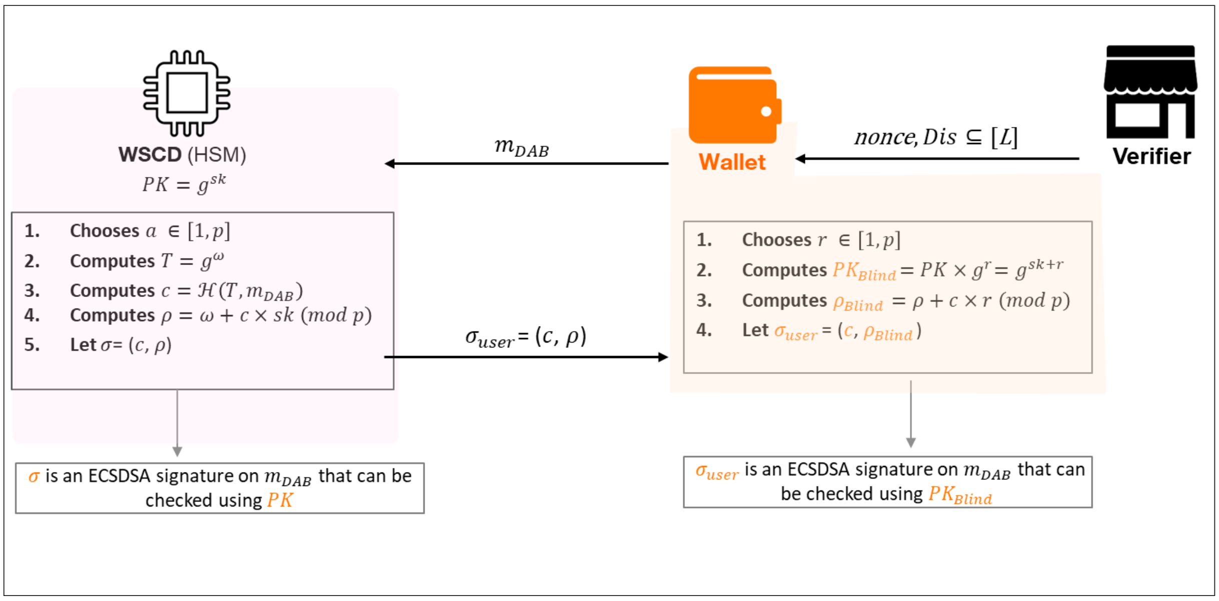
Joint computation of PK_{Blind} and σ_{User} with ECSChnorr : We assume that the SE supports the classical ECSchnorr digital signature algorithm, also known as ECSDSA in ISO/IEC 14888-3 standard. It should be noted that in this standard, the so-called "Weak" version of the Fiat-Shamir heuristic is implemented; however, in certain contexts, this version is vulnerable to an attack introduced at Asiacrypt 2012 by Bernhard et al. [BPW12]. Although this attack does not apply in our context, we will nevertheless indicate how to use the so-called Strong version of the Fiat-Shamir heuristic (Strong FS) with ECSDSA (as specified in ISO/IEC 14888-3 standard). The attack by Bernhard et al. [BPW12] does not apply to non-interactive proofs using the Strong version of the Fiat-Shamir heuristic.

The joint computation of the signature $\sigma_{User} = SoK\{\alpha: PK_{Blind} = g^{\alpha}\}(m_{DAB})$ could be performed in the following way (see figure 6) :

1. The SE will first calculate a signature of knowledge (denoted σ) of the discrete logarithm of PK in the base g (i.e. of its private key sk) : $\sigma = SoK\{\alpha: PK = g^{\alpha}\}(m_{DAB})$[26]. The algorithm called ECSDSA will be used to compute this signature. This signature of knowledge is computed as follows using the ECSDSA algorithm. The SE generates a random value ω and computes $T = g^{\omega}$, $c = \mathcal{H}(T \parallel m_{DAB})$ and $\rho = \omega + c\ sk$ (mod p) where $\mathcal{H}$ denotes a cryptographic hash function (e.g., SHA-256). The signature σ consists of the pair (c, ρ) : $\sigma = (c, \rho)$. It is valid if $c' = \mathcal{H}(g^{\rho} \times PK^{-c}, m_{DAB}) = c$ and invalid otherwise.
2. The M-Wallet chooses a random value r and computes $PK_{Blind} = g^{r} \times PK = g^{sk+r}$ and ρ_{Blind} = $\rho + c \times r = \omega + c \times sk + c \times r$ = $\omega + c \times (sk + r)$ (mod p).

The signature $\sigma_{User} = (c, \rho_{Blind})$ is a valid ECSDSA signature on m_{DAB} with respect to the public key PK_{Blind} (Q.E.D).

[26] m_{DAB} is typically transmitted to the SE by the M-Wallet. The M-Wallet could include PK_{Blind} in it if we prefer to use the Strong version of the Fiat-Shamir heuristic instead of the weak version. However, it is important to note that the attack by Bernhard et al. would not apply in our context with the weak version of the Fiat-Shamir heuristic.



WSCD (HSM)
$PK = g^{sk}$
1. Chooses $\alpha \in [1,p]$
2. Computes $T = g^{\omega}$
3. Computes $c = \mathcal{H}(T, m_{DAB})$
4. Computes $\rho = \omega + c \times sk \ (mod\ p)$
5. Let $\sigma = (c, \rho)$
m_{DAB}
$\sigma_{user} = (c, \rho)$
Wallet
$nonce, Dis \subseteq [L]$
Verifier
1. Chooses $r \in [1,p]$
2. Computes $PK_{Blind} = PK \times g^r = g^{sk+r}$
3. Computes $\rho_{Blind} = \rho + c \times r \ (mod\ p)$
4. Let $\sigma_{user} = (c, \rho_{Blind})$
σ is an ECSDSA signature on m_{DAB} that can be checked using PK
σ_{user} is an ECSDSA signature on m_{DAB} that can be checked using PK_{Blind}

Figure 6 : Joint computation of PK_{Blind} and σ_{User} with ECSChnorr (ECSDSA)

The VP consists of the public key of the IdP, the public key PK_{Blind}, the $\{\mathfrak{A}d\}_{i \notin \mathcal{D}}$, the set of attributes disclosed to the RP ($\mathcal{D}_{disclosed} = \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}$), and the signatures σ_{Issuer}^{Blind} and σ_{User} : $VP = (PK_I, PK'_I, PK_{Blind}, \{\mathfrak{A}d\}_{i \notin \mathcal{D}}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}, \sigma_{Issuer}^{Blind}, \sigma_{User})$.

- The user transmits VP to the RP.

The RP first computes $H'_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ for each $i \in \mathcal{D}$.
It then verifies that :

1) $H'_i = H_i$ for each $i \in \mathcal{D}$;
2) σ_{User} is a valid ECSDSA signature on m_{DAB} (which the RP can reconstruct) using the public key PK_{Blind} certified by the issuer. The verification algorithm for this SoK ($SoK\{\alpha : PK_{Blind} = g^{\alpha}\}(m_{DAB})$) being relatively standard, we will not detail it in this document.
3) σ_{Issuer}^{Blind} is a valid BBS+ signature on PK_{Blind}, $\{H_i\}_{i \in \mathcal{D}}$ and $\{H'_i = \mathfrak{A}d\}_{i \notin \mathcal{D}}$, using the public key of the IdP. To do this, it must first ensure that the following equality $\bar{B} = \hat{A}^{sk_I}$ is satisfied[27] and secondly that $\Pi_{validity}$ is valid. The verification algorithm for this ZKP ($\Pi_{validity}$) being relatively standard, we will not detail it in this document.

If all these conditions are met, the RP grants access to its site.

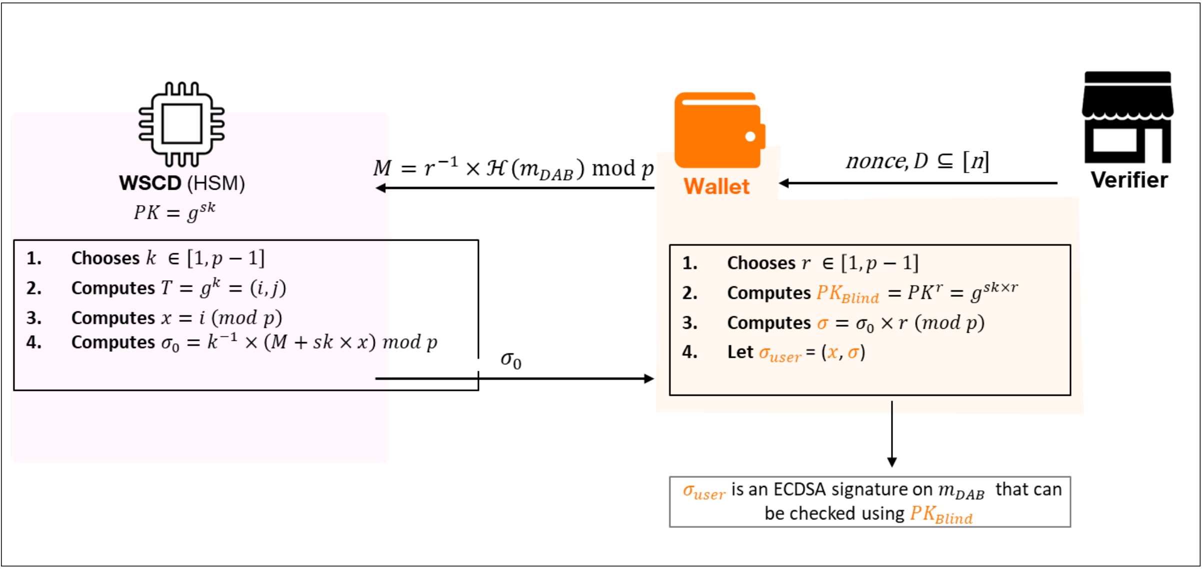
[27] There are several methods to do this, with [BBS24] or without bilinear pairings (see for example [BBDT16] or the solution based on Blind Tokens, described below for the "semi-offline" mode). If the IdP has implemented the BBS+ version using bilinear pairings, then this verification could be done as follows: $e(\hat{A}, PK'_I) = e(\bar{B}, f)$

Joint computation of PK_{Blind} and σ_{User} with ECDSA: This time, we will assume that the SE supports the classic digital signature algorithm ECSDSA [FIPS186-4, ISO/IEC 14888 3].

The joint computation of the signature ECDSA (σ_{User}) on the message m_{DAB} using the private key $sk_{Blind} = sk \times r\ (mod\ p)$, could be performed in the following way (see figure 7) :

1. The M-Wallet calculates $M = r^{-1} \times \mathcal{H}(m_{DAB})$[28] mod p and transmits M to the SE after authenticating itself with the latter.
2. The SE chooses a random value $k \in Z_p^*$ and calculates $g^k = (i, j)$[29] and $x = i$ (mod p).
3. If x = 0 then go back to step 2.
4. The SE calculates $\sigma_0 = k^{-1}(M + sk \times x)$ mod p.
5. If σ_0 = 0 go back to step 2. Otherwise, the SE transmits σ_0 to the M-Wallet.
6. The M-Wallet computes $\sigma = r \times \sigma_0 = k^{-1}(\mathcal{H}(m_{DAB}) + sk_{Blind} \times x)$ mod p

   The signature $\sigma_{User} = (x, \sigma)$ is a valid ECDSA signature on m_{DAB} with respect to the public key PK_{Blind} (Q.E.D).



WSCD (HSM)
$PK = g^{sk}$
1. Chooses $k \in [1, p-1]$
2. Computes $T = g^k = (i, j)$
3. Computes $x = i\ (mod\ p)$
4. Computes $\sigma_0 = k^{-1} \times (M + sk \times x)\ mod\ p$
$M = r^{-1} \times \mathcal{H}(m_{DAB})$ mod p
σ_0
Wallet
$nonce, D \subseteq [n]$
Verifier
1. Chooses $r \in [1, p-1]$
2. Computes $PK_{Blind} = PK^r = g^{sk \times r}$
3. Computes $\sigma = \sigma_0 \times r\ (mod\ p)$
4. Let $\sigma_{user} = (x, \sigma)$
σ_{user} is an ECDSA signature on m_{DAB} that can be checked using PK_{Blind}

Figure 7: Joint calculation of PK_{Blind} and σ_{User} with ECDSA

The VP consists of the public key of the IdP, the public key PK_{Blind}, the $\{\mathfrak{A}d\}_{i \notin \mathcal{D}}$, the set of attributes disclosed to the RP ($\mathcal{D}_{disclosed} = \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}$), and the signatures σ_{Issuer}^{Blind} and σ_{User} : $VP = (PK_I, PK'_I, PK_{Blind}, \{\mathfrak{A}d\}_{i \notin \mathcal{D}}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}, \sigma_{Issuer}^{Blind}, \sigma_{User})$.

- The user transmits VP to the RP.

The RP first computes $H'_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ for each $i \in \mathcal{D}$.
It then verifies that:

1) $H'_i = H_i$ for each $i \in \mathcal{D}$;
2) σ_{User} is a valid ECDSA signature on m_{DAB} (which the RP can reconstruct) using the public key PK_{Blind} certified by the issuer. The verification algorithm for an ECDSA

[28] m_{DAB} could include PK_{Blind} if one wishes to protect against a similar attack to that of Bernhard et al. (against the ECSchnorr algorithm using the weak version of the Fiat-Shamir heuristic) that also applies to ECDSA.
[29] Here, we are abusively using multiplication (instead of addition) to denote the group operation in G. The element g^k is indeed a point on the underlying elliptic curve.

signature being relatively standard (see for instance [FIPS186-4, ISO/IEC 14888-3]), we will not detail it in this document.

3) σ_{Issuer}^{Blind} is a valid BBS+ signature on PK_{Blind}, $\{H_i\}_{i\in\mathcal{D}}$ and $\{H'_i = \mathfrak{A}d\}_{i\notin\mathcal{D}}$, using the public key of the IdP. To do this, it must first ensure that the following equality $\bar{B} = \hat{A}^{sk_I}$ is satisfied[30] and secondly that $\Pi_{validity}$ is valid[31]. The verification algorithm for this ZKP ($\Pi_{validity}$) being relatively standard, we will not detail it in this document.

If all these conditions are met, the RP grants access to its site.

Note 5: If all the conditions are met, it can be easily demonstrated that the attributes $\{a_i\}_{i\in\mathcal{D}}$ have been certified by the IdP and that the VP indeed originates from the user whose attributes are the $\{a_i\}_{i\in\mathcal{D}}$.

Note 6: the σ_{User} signature would be generated by the user's SE, while the σ_{Issuer}^{Blind} signature would be computed by the dedicated mobile application running on the user's mobile phone. With our solution, the SE would not need to perform any bilinear pairings and would only need to perform a single scalar multiplication on the elliptic curve to generate the σ_{User} signature (compared to a linear number in n with the original version of BBS+ currently being standardized at IETF [BBS24]). For certain versions of BBS+, the user's secure device would only need a constant number of scalar multiplications on the elliptic curve to generate the σ_{User} signature but would need to interact multiple times with the mobile application to compute this signature, which could be problematic if a remote secure device like an HSM is implemented (HSMs generally do not handle session management very well).

It can be easily proved that neither the issuer nor the RP can trace a user based on their VP (provided, of course, that the user has not disclosed any personally identifiable attributes or attributes that can be used to identify them), see section 3. In other words, with the solution presented above, a user will be able to only disclose the attributes required by the service provider to access its services, without being traceable by the IdP and/or the RP. Furthermore, the selective disclosure method we use fully complies with the ISO/IEC 18013-5 standard (which unfortunately, in its "standard usage" i.e., without the method described in this paper, would allow colluding IdPs and RPs, to trace the usages of their common customers).

[30] There are several methods to do this, with [BBS24] or without bilinear pairings (see for example [BBDT16] or the solution based on Blind Tokens, described below for the "half-offline" mode). If the IdP has implemented the BBS+ version using bilinear pairings, then this verification could be done as follows : $e(\hat{A}, PK'_I) = e(\bar{B}, f)$

[31] In the ECDSA case, the proof $\Pi_{validity}$ would be the following : $\Pi_{validity} = \mathrm{PoK}\{\alpha,\beta,\gamma,\delta,\{\tau_i\}_{i\notin\mathcal{D}}: \bar{B} = \hat{A}^{-\alpha}D^{\beta} \wedge F^{\delta}PK_{Blind} = D^{\gamma}\prod_{i\notin\mathcal{D}} g_i^{-\tau_i} \wedge \delta \neq 0\ (mod\ p)\}$ where $F = g_0\prod_{i\in\mathcal{D}} g_i^{H_i}$. We have the following two equalities, hence the validity proof $\Pi_{validity}$: $\bar{B} = \hat{A}^{-e}D^{r_1}$ and $F^{r}PK_{Blind} = D^{r\times r_3}\prod_{i\notin\mathcal{D}} g_i^{-r\times H_i}$

Half offline mode : In the ISO 18013-5 standard, two verification modes for a VP are proposed. One mode, which we will refer to as "offline," does not require the user or the RP to be connected to generate and verify a VP. The other mode, which we will refer to as "half-offline" (HOL), requires one of these two actors to be connected for the transaction (VP) to be successfully completed. The offline mode is the one described in the previous sections. The HOL mode corresponds to the synchronous mode (or federated identity model) described in Figure 1, during which the user, after authenticating with the IdP, will receive a *Token* from the IdP attesting that they meet the access policy of the RP.

In the following, we also propose a solution for the HOL mode, which does not have the aforementioned drawbacks of the federated identity model. In particular, the IdP will not know which RP the user is using the Token with and therefore cannot trace the user's usage. **Please note that this works both when the splitting is done with ECSDSA as when it is done with ECDSA.**

In the following, we assume that the user already possesses the VC signature = (A, e) (see figure 5) that they intend to use to access the services of the RP.

- The user will first "anonymize" their public key (to prevent the issuer and the RP from tracing them based on this key). To do this, they will randomly pick an integer r in Z_p and compute: $PK_{Blind} = g^{sk+r}$ (see Note 6 above for the joint computation of this public key).
- They will then "anonymize" the BBS+ signature of their VC to also prevent the issuer and the RP from tracing them based on this element, and "adapt" it to be on PK_{Blind} and the $\{H_i\}_{i=1}^{n}$ (rather than on PK and the $\{H_i\}_{i=1}^{n}$)[32]. To do this, they will choose integers $r_1, r_2 \in Z_p^*$ and compute :
    - $\hat{A} = A^{r_1 \times r_2}$
    - $D = B^{r_2}$
    - $\bar{B} = \hat{A}^{-e} D^{r_1} = \hat{A}^{sk_I}$
    - $r_3 = r_2^{-1} (\text{mod } p)$
    - $\Pi_{validity} =$ PoK$\{\alpha, \beta, \gamma, \delta, \{\tau_i\}_{i \notin \mathcal{D}} : \bar{B} = \hat{A}^{-\alpha} D^{\beta} \wedge g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{\gamma} \prod_{i \notin \mathcal{D}} g_i^{-\tau_i} g^{\delta}\}$.[33] This ZKP ($\Pi_{validity}$) being relatively standard, we will not detail it in this document.
    - The user will then attempt to obtain, **online**, from the IdP, a **blind** proof that $\bar{B} = \widehat{A}^{sk_I}$ where $\hat{A} = A^{r_1 \times r_2}$. We will refer to this proof as π_{EQ} (SoK). By blind, we mean that the IdP, although contributing to the generation of this proof (as only they know sk_I), will be unable, in the presence of such a proof, to determine which user it was intended for. This proof described in Figure 8 can be obtained using the blind signature protocol of Chaum-Pedersen [CP92][34], which is

[32] This is possible with BBS+ signatures without compromising the security (unforgeability) of such signatures.

[33] We have the following two equalities, hence the validity proof $\Pi_{validity}$: $\bar{B} = \hat{A}^{-e} D^{r_1}$ and $g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{r_3} \prod_{i \notin \mathcal{D}} g_i^{-H_i} g^{r}$

[34] Benhamouda et al. [BLL+22] have shown that certain blind signature schemes are vulnerable to *a parallel attack (ROS attack)*. Specifically, a user who concurrently executes a large number (l) of sessions of the Chaum-Pedersen blind signature protocol with the IdP could, after these l sessions, succeed in forging an additional valid

standardized in ISO/IEC ([ISO18370], mechanism 4). In fact, it is this protocol that is implemented in Figure 8.

The VP consists of the public key of the IdP, the public key PK_{Blind}, the $\{\mathfrak{A}d\}_{i \notin \mathcal{D}}$, the set of data disclosed to the RP ($\mathcal{D}_{disclosed} = \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}$)[35], the signatures σ_{Issuer}^{Blind} and σ_{User} and the proof π_{EQ}: $VP = (PK_I,\ PK'_I, PK_{Blind}, \{\mathfrak{A}d\}_{i \notin \mathcal{D}}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}, \sigma_{Issuer}^{Blind}, \sigma_{User}, \pi_{EQ})$.

- The user transmits VP to the RP.

The RP first computes $H'_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i)$ for each $i \in \mathcal{D}$.
It then verifies that :

1) $H'_i = H_i$ for each $i \in \mathcal{D}$;
2) σ_{User} is a valid ECSDSA signature on m_{DAB} (which the RP can reconstruct) using the public key PK_{Blind} certified by the issuer. The verification algorithm for this SoK ($SoK\{\alpha: PK_{Blind} = g^{\alpha}\}(m_{DAB})$) is described above (see Note 6).
3) σ_{Issuer}^{Blind} is a valid BBS+ signature on PK_{Blind}, $\{H_i\}_{i \in \mathcal{D}}$ and $\{H'_i = \mathfrak{A}d\}_{i \notin \mathcal{D}}$, using the public key of the IdP. To do this, it must first ensure that the π_{EQ} proof is valid[36] (see figure 8) and secondly that $\Pi_{validity}$ is valid. The verification algorithm for this ZKP ($\Pi_{validity}$) being relatively standard, we will not detail it in this document.

If all these conditions are met, the RP grants access to its site.

Note 7 : The particularity of our solution, for the HOL mode, is that it does not require the use of pairings, which makes it much more efficient than solutions that implement these kind of computations. Its security also relies on computational problems that are more widely accepted () by the scientific community.

signature (l+1 signatures instead of l). However, this attack would not apply to our context; in fact, the requests (per user) would only be made in sequential mode (or in a very limited number in concurrent mode).

[35] To illustrate selective disclosure with BBS#, we deliberately relied on the ISO/IEC 18013-5 standard. However, it is important to note that any other representation of attributes and any other method enabling selective disclosure (such as using commitments and zero-knowledge proofs) could be implemented with BBS#.

[36] This π_{EQ} proof effectively constitutes a proof that the user's VC has not been revoked. Indeed, a IdP will refuse to provide this proof if the user's VC signature = (A, e) has been revoked. This technique is therefore a very simple (comparable to using a protocol such as OCSP - Online Certificate Status Protocol) and privacy-preserving method to prove the non-revocation of a (Q)EEA involved in a transaction/VP.

U		IdP
Public input: pp, $PK_I, PK'_I, \{K_i\}_{i=1}^{n}$, $\{HID_i\}_{i=1}^{n}$, $\{ID_{a_i}\}_{i=1}^{n}$, PK, A, e **Private input**: sk, $l = r_1 \times r_2$, $\hat{A} = A^{r_1 \times r_2}, \bar{B}$		**Public input**: pp, PK_I, $PK'_I, a_1, a_2, a_3, \dots, a_n$, $\{K_i\}_{i=1}^{n}$, $\{HID_i\}_{i=1}^{n}$, $\{ID_{a_i}\}_{i=1}^{n}$, PK, A, e **Private input**: sk_I
	$\xleftarrow{ch}$	
Computes: • π_U= SoK $\{\alpha: PK = g^{\alpha}\}(ch)$		
	$\xrightarrow{\pi_U}$	
		Verifies the proof π_U (aborts if it is invalid)
		Randomly chooses : $s \in Z_p^*$ % Let $H_i = \mathcal{H}(HID_i$ ‖ ID_{a_i} ‖ a_i ‖ $K_i)$ for i in [1, n], % Let $B = g_0 PK \prod_{i=1}^{n} g_i^{H_i}$ % We have $BA^{-e} = A^{sk_I}$ % Let $C = A^{sk_I} = BA^{-e}$ IdP computes: $A_0 = \tilde{g}^s$ $B_0 = A^s$
	$\xleftarrow{A_0, B_0}$	
Chooses : $u, v \in Z_p^*$ Computes : • $\ddot{A}_0 = (A_0 \times \tilde{g}^v)^u$ • $\ddot{B}_0 = (B_0^l \times \hat{A}^v)^u$ • $c = \mathcal{H}(\hat{A}$ ‖ $\bar{B}$ ‖ $\ddot{A}_0$ ‖ $\ddot{B}_0$ ‖ $m_{DAB})$ • $c_0 = c/u$ (mod p)		
	$\xrightarrow{c_0}$	
	$\xleftarrow{r_0 = s + c_0 \times sk_I \ (mod\ p)}$	

Calculates :

- $r = (r_0 + v)u$ (mod p)

The proof $\pi_{EQ} = (\hat{A}, \bar{B}, c, r)$.
It is a valid proof (SoK) on the message m_{DAB} if :

$$c' = \mathcal{H}(\hat{A} \parallel \bar{B} \parallel \tilde{g}^r \times PK_I^{-c} \parallel \hat{A}^r \times \bar{B}^{-c} \parallel m_{DAB}) = c$$

Figure 8: Blind computation of the π_{EQ} proof

3. Security

It is worth reminding that a VC issued using the BBS/BBS+ signature algorithm is unforgeable under the q-SDH assumption [TZ23]. Two other fundamental security properties for anonymous credential systems have been defined (see, for example, [BBDT16]): the unforgeability of VPs and the anonymity of such VPs.

Informally, the first property characterizes the fact that an adversary must be unable to generate a VP that will be accepted by an RP if they do not possess a valid VC (issued by an IdP) that satisfies the access policy of that RP.

The second property expresses the fact that a VP reveals no other information than the attributes that the user agreed to disclose to the RP. In the particular case where the user does not disclose any identifying attributes (or any attributes at all), they will be *perfectly anonymous* among the set of users sharing the same attributes as them. For example, if the disclosed attribute is the “year of birth”, the user will be perfectly anonymous among all individuals born in the same year and who obtained a VC from the same IdP.

Firstly, we will prove that the unforgeability of BBS# VPs relies on the q-SDH assumption. In other words, if a VP is accepted by an RP, then this implies that the user knows a valid VC that satisfies the access policy of that RP, as well as the private key associated with that VC.[37]

Unforgeability of VPs

Proof: Let $VP = (PK_I,\ PK'_I, PK_{Blind}, \{\mathfrak{A}d\}_{i \notin \mathcal{D}}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}},\ \sigma_{Issuer}^{Blind}, \sigma_{User})$ be a given VP where $\sigma_{Issuer}^{Blind} = (\hat{A}, \bar{B}, D, \Pi_{validity})$. We recall that $\sigma_{User} = SoK\{\alpha : PK_{Blind} = g^{\alpha}\}(m_{DAB})$. Therefore, the signature σ_{User} is a proof of knowledge of the discrete logarithm of PK_{Blind} in the base g. By using the *extractor* for this proof of knowledge, we can extract sk_{Blind} such that :

$$PK_{Blind} = g^{sk_{Blind}} \quad \textbf{(1)}$$

[37] It is worth noting that the private key associated with a VC is necessarily managed by a WSCD. Indeed, during the issuance of a VC, the IdP ensures, beforehand, before issuing a VC, that the request is legitimate and that the public key presented to it comes from a WSCD (via an attestation, for example). Furthermore, Tessaro and Zhu [TZ23] have demonstrated, under the q-SDH assumption, that the only way to obtain a valid VC is to request it from an IdP.

By using the extractor for the $\Pi_{validity}$ proof, we can extract values e, r, r_1, r_3, $\{H_i\}_{i=1}^{n}$ such that:

$$\bar{B} = \hat{A}^{-e} D^{r_1} \quad \mathbf{(2)}$$

$$g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{r_3} \prod\nolimits_{i \notin \mathcal{D}} g_i^{-H_i} g^r \quad \mathbf{(3)}$$

We also know that if the VP is accepted by the RP, then:

$$\bar{B} = \hat{A}^{sk_I} \qquad \mathbf{(4)}$$

(2) and (4) therefore implies that:

$$\hat{A}^{sk_I + e} = D^{r_1} \quad \mathbf{(5)}$$

If $r_1 = 0$ (mod p), (5) implies that $sk_I = -e$ (mod p) and therefore our reduction has found the private key of the IdP. This is impossible under the discrete logarithm assumption. Therefore, we will assume that $r_1 \neq 0$ (mod p).

From (3) and (1), we can deduce that $D^{r_3} = g_0 g^{sk_{Blind} - r} \prod_{i=1}^{n} g_i^{H_i}$ **(6)**

- If $r_3 = 0$ (mod p) then $g_0 = g^{sk_{Blind} - r} \prod_{i=1}^{n} g_i^{H_i}$. We have therefore found a representation of g_0 in the base (g, $\{g_i\}_{i=1}^{n}$). This is also impossible under the discrete logarithm assumption. Therefore, we will assume that $r_3 \neq 0$ (mod p).
- If $r_3 \neq 0$ (mod p), from the equalities (5) and (6), we obtain $\left(\hat{A}^{r_3 r_1^{-1}}\right)^{sk_I + e} = g_0 g^{sk_{Blind} - r} \prod_{i=1}^{n} g_i^{H_i}$.

Therefore, we have extracted a valid BBS+ signature ($\hat{A}^{r_3 r_1^{-1}}$,e) on ($sk_{Blind} - r$,$\{H_i\}_{i=1}^{n}$). Under the q-SDH assumption, the user therefore holds a valid BBS+ signature (i.e., a VC) on the $\{H_i\}_{i \in D}$. The $\{H_i\}_{i \in D}$ disclosed to the RP have thus been certified by the IdP and presented by the user to whom the corresponding VC was issued (as the user knows the key $sk = sk_{Blind} - r$). ■

We will now demonstrate that BBS# VPs are *perfectly anonymous* (everlasting privacy).

Anonymity of VPs

Proof. Let $VP = (PK_I, PK'_I, PK_{Blind}, \{\mathfrak{Ad}\}_{i \notin \mathcal{D}}, \{HID_i \parallel ID_{a_i} \parallel a_i \parallel K_i\}_{i \in \mathcal{D}}, \sigma_{Issuer}^{Blind}, \sigma_{User})$ be the VP received by the RP where $\sigma_{Issuer}^{Blind} = (\hat{A}, \bar{B}, D, \Pi_{validity})$.
Let's show that this VP could have been generated by any user $\mathcal{U}$' (different from the actual user $\mathcal{U}$ who generated the above VP but sharing the same attributes disclosed by $\mathcal{U}$ to the RP) who obtained a VC from the IdP with the key pair (PK_I, PK'_I).

Let VC' = (A', e') be the VC obtained by $\mathcal{U}'$ on their public key $PK' = g^{sk'}$ (where their private key sk' is exclusively managed by their WSCD) and their attributes $(a'_1, a'_2, a'_3, \dots, a'_n)$.
As D is an element of a cyclic group of order p and D is different from 1 (the neutral element of this group, considered here as multiplicative), this implies that there exists an integer $r'_2 \in Z_p^*$ such that $D = B'^{r'_2}$ where $B' = g_0 PK \prod_{i=1}^{n} g_i^{H'_i}$ and $H'_i = \mathcal{H}(HID_i \parallel ID_{a_i} \parallel a'_i \parallel K_i)$.
For the same reasons, there exists an integer $r'_1 \in Z_p^*$ such that $\hat{A} = A'^{r'_1 \times r'_2}$
Let's show that $\bar{B} = \hat{A}^{-e'} D^{r'_1} = \hat{A}^{sk_I}$.

By definition : $A'^{sk_I + e'} = B'$ and thus $A'^{sk_I} = B' A'^{-e'}$.

Which therefore implies that :

$$\hat{A}^{sk_I} = A'^{r'_1 \times r'_2 \times sk_I} = B'^{r'_1 \times r'_2} A'^{r'_1 \times r'_2 \times (-e')} = D^{r'_1} \hat{A}^{-e'} = \bar{B}$$

Furthermore, there exists $r' \in Z_p^*$ such that $PK_{Blind} = \ g^{sk+r} = g^{sk'+r'}$. σ_{User} is therefore a valid ECSchnorr signature produced with the private key $sk' + r' = sk + r$ (mod p).

Let’s also show that $g_0 PK_{Blind} \prod_{i \in \mathcal{D}} g_i^{H_i} = D^{r'_3} \prod_{i \notin \mathcal{D}} g_i^{-H'_i} g^{r'}$ where $r'_3 = r'^{-1}_2$(mod p).

By definition, $D^{r'_3} = \ B' = g_0 PK' \prod_{i=1}^{n} g_i^{H'_i} = g_0 PK_{Blind} \prod_{i=1}^{n} g_i^{H'_i} g^{-r'}$. Therefore, we have the equality above since $H_i = \ H'_i$ for $i \in \mathcal{D}$ (since we have assumed that $\mathcal{U}'$ shares the same disclosed attributes to the RP as $\mathcal{U}$)

Given that the proof $\Pi_{validity}$ is *witness-indistinguishable*, it reveals no information (even to an attacker with unbounded computational power) about the elements $(\alpha, \beta, \gamma, \delta, \{\tau_i\}_{i \notin \mathcal{D})})$ used to produce the proof $\Pi_{validity}$.

Therefore, an attacker, even with unbounded computational power, has no way to determine whether $\mathcal{U}$ or $\mathcal{U}'$ generated the $VP = (PK_I,\ PK'_I, PK_{Blind}, \{\mathfrak{A}d\}_{i \notin \mathcal{D}}, \{HID_i \ \| \ ID_{a_i} \ \| \ a_i \ \| \ K_i\}_{i \in \mathcal{D}},$ $\sigma_{Issuer}^{Blind}, \sigma_{User})$. This concludes the demonstration that BBS# VPs offer perfect anonymity (Everlasting Privacy). ■

4. Performances

In this section, we compare the efficiency (in terms of key size and computation time for credential issuance and presentation) of BBS# with that of ISO mDL/SD-JWT (instantiated with the ECDSA signature scheme) and PQ-ABC, a recent post-quantum anonymous credential scheme that will appear at the forthcoming ACM CCS conference [AGJ+24].

Space Efficiency (bytes)				
	Private Key (Holder, Issuer)	**Public Key** (Holder, Issuer)	**Credential Size**	**Presentation Proof**
BBS#	(32, 32)	(32, 32)	128	416+$U \times 32$
SD-JWT and mDL[1]	(32, 32)	(32, 32)	64	64+$N \times 32$
PQ-ABC[2]	(0,25 KB, 10 KB)	(2,38 KB, 47, 53 KB)	6,81 KB	79,58 KB[3]

- N: number of signed attributes
- U: number of undisclosed attributes

1. with ECDSA used on both the Holder and Issuer's side
2. [AGJ+24]
3. For U = 10 tl

Time Efficiency*				
	Credential Issuance[1]	**Present WSCD**	**Present Wallet**	**Verify Presentation**
BBS#	$N\ \mathbb{E}_{G_1}$ (630 μs)	1 $\mathbb{E}_{G_1}$ (50 ms)	$(N$+9) $\mathbb{E}_{G_1}$ (3,8 ms)	$(N$+12) $\mathbb{E}_{G_1}$ (1,4 ms)
SD-JWT and mDL[2]	1 $\mathbb{E}_{G_1}$ (63 μs)	1 $\mathbb{E}_{G_1}$ (50 ms)	-	2 $\mathbb{E}_{G_1}$ (126 μs)
PQ-ABC[3,4]	**400ms**	**N.A**[5]	**355 ms**	**147 ms**

*We do not consider operations in Z_p since their cost is negligible compared to the other ones

- $\mathbb{E}_{G_1}$: cost of an exponentiation /scalar multiplication in G_1:
    - 63 μs on an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
    - 0,2 ms on a Samsung S10e over the secp256r1 curve
    - 50 ms on a Javacard 2.2.2 SIM card, Global Platform 2.2 compliant, over the secp256r1 curve
- N: number of signed attributes
- U: number of undisclosed attributes

1. N =10
2. with ECDSA used on both the Holder and Issuer's side
3. Benchmarked on an Intel Core i7 12800H CPU running at 4.6 GHz
4. [AGJ+24]
5. Current WSCD do not support the computations involved in [AGJ+24]

5. Anonymous credential protocols in a pre-quantum and post-quantum world

In this section, we compare the three aforementioned protocols (BBS#, ISO mDL with ECDSA, and PQ-ABC) in terms of the level of security and privacy they provide.

Security and Privacy				
	Credential Unforgeability	**VP Unlinkability Colluding RPs**	**VP Unlinkability Colluding RP-Issuer**	**VP Unforgeability**
BBS#	NPQ Assumption[1]	**Unconditional**[2] (Everlasting Privacy)	**Unconditional** (Everlasting Privacy)	NPQ Assumption
SD-JWT[3] and mDL	Unknown Assumption (NPQ security)	**No**	**No**	Unknown Assumption (NPQ Security)
PQ-ABC	PQ Assumption	**PQ Assumption**	**PQ Assumption**	PQ Assumption

1. Classical assumption (i.e. not PQ) , namely q-SDH
2. Even against an adversary with unbounded computational power
3. With ECDSA used on both the Holder and Issuer's side

6. Conclusion

What we aimed to demonstrate through this technical report is that it is possible to achieve SSI transactions, particularly for EUDIW eIDAS 2.0, which are not only secure and certifiable at the highest level but also provide strong (optimal) privacy protection for EUDIW users. Furthermore, what seems to be new is that this can be accomplished by leveraging existing security infrastructures that are already deployed everywhere, whether in smartphones or in the cloud. This result demonstrates that by using proven and conceptually simple cryptographic primitives (such as "classic" Schnorr-style ZKPs or "Sigma Protocols"), it is possible to combine the following seemingly contradictory properties: security (including hardware), full unlinkability, and everlasting privacy..

The proposed solution is, of course, not an end in itself but rather the foundation upon which privacy-respecting services, picularly those based on the principle of maximum minimization, could be built (by relying on relatively simple ZKPs). Therefore, we encourage the entire eIDAS 2.0 ecosystem to provide feedback on the proposed solution and to consider these ZKPs for their true value, especially in a privacy-by-design approach, both for the definition of the ARF and for eIDAS 2.0 services that are built on top of it.

7. References

[ABBT16] Roberto Araújo, Amira Barki, Solenn Brunet, Jacques Traoré: Remote Electronic Voting Can Be Efficient, Verifiable and Coercion-Resistant. Financial Cryptography Workshops 2016: 224-232

[AGJ+24] Sven Argo, Tim Güneysu, Corentin Jeudy, Georg Land, Adeline Roux-Langlois, Olivier Sanders: Practical Post-Quantum Signatures for Privacy. IACR Cryptol. ePrint Arch. 2024: 131 (2024)

[ASM06] Man Ho Au, Willy Susilo, Yi Mu: Constant-Size Dynamic k-TAA. SCN 2006: 111-125

[BBDT16] Amira Barki, Solenn Brunet, Nicolas Desmoulins, Jacques Traoré: Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials. SAC 2016: 360-380

[BBS04] Dan Boneh, Xavier Boyen, Hovav Shacham: Short Group Signatures. CRYPTO 2004: 41-55

[BBS24] **The BBS Signature Scheme, published** 31 January 2024 **: The BBS Signature Scheme (identity.foundation)**.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, CRYPTO 2020, Part II, volume 12171 of LNCS, pages 121–151. Springer, Heidelberg, August 2020.

[BLL+22] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, Mariana Raykova: On the (in)Security of ROS. J. Cryptol. 35(4): 25 (2022)

[BPW12] David Bernhard, Olivier Pereira, Bogdan Warinschi: How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. ASIACRYPT 2012: 626-643

[Bra97] Stefan Brands: Rapid Demonstration of Linear Relations Connected by Boolean Operators. EUROCRYPT 1997: 318-333

[CDL16] Jan Camenisch, Manu Drijvers, Anja Lehmann: Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. IACR Cryptol. ePrint Arch. 2016: 663 (2016)

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, EUROCRYPT 2006, volume 4004 of LNCS, pages 1–11. Springer, Heidelberg, May / June 2006.

[CP92] David Chaum, Torben P. Pedersen: Wallet Databases with Observers. CRYPTO 1992: 89-105

[CS97] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In CRYPTO 97, volume 1294, pages 410–424. Springer, 1997.

[FS86] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature schemes. In: Advances in Cryptology — Crypto'86. Volume 263 of Lecture Notes in Computer Science., Springer-Verlag (1986) 186–194

[GMR85] Shafi Goldwasser, Silvio Micali and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In 17th ACM STOC, pages 291–304. ACM Press, May 1985.

[ISO14888] ISO/IEC 14888-3:2018. Techniques de sécurité IT — Signatures numériques avec appendice. Partie 3: Mécanismes basés sur un logarithme discret.

[ISO18370] ISO/IEC 18370-2:2016. Technologie de l'information — Techniques de sécurité — Signatures numériques en aveugle. Partie 2: Mécanismes fondés sur le logarithme discret.

[JSI96] Markus Jakobsson, Kazue Sako, Russell Impagliazzo: Designated Verifier Proofs and Their Applications. EUROCRYPT 1996: 143-154

[JY09] David Jao and Kayo Yoshida. Boneh-Boyen signatures and the strong Diffie-Hellman problem. In Hovav Shacham and Brent Waters, editors, PAIRING 2009, volume 5671 of LNCS, pages 1–16. Springer, Heidelberg, August 2009.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, CRYPTO'91, volume 576 of LNCS, pages 129–140. Springer, Heidelberg, August 1992.

[TZ23] Stefano Tessaro, Chenzhi Zhu: Revisiting BBS Signatures. EUROCRYPT (5) 2023: 691-721