Orange
Innovation

BBS# and eIDAS 2.0

Making BBS Anonymous Credentials eIDAS 2.0 Compliant

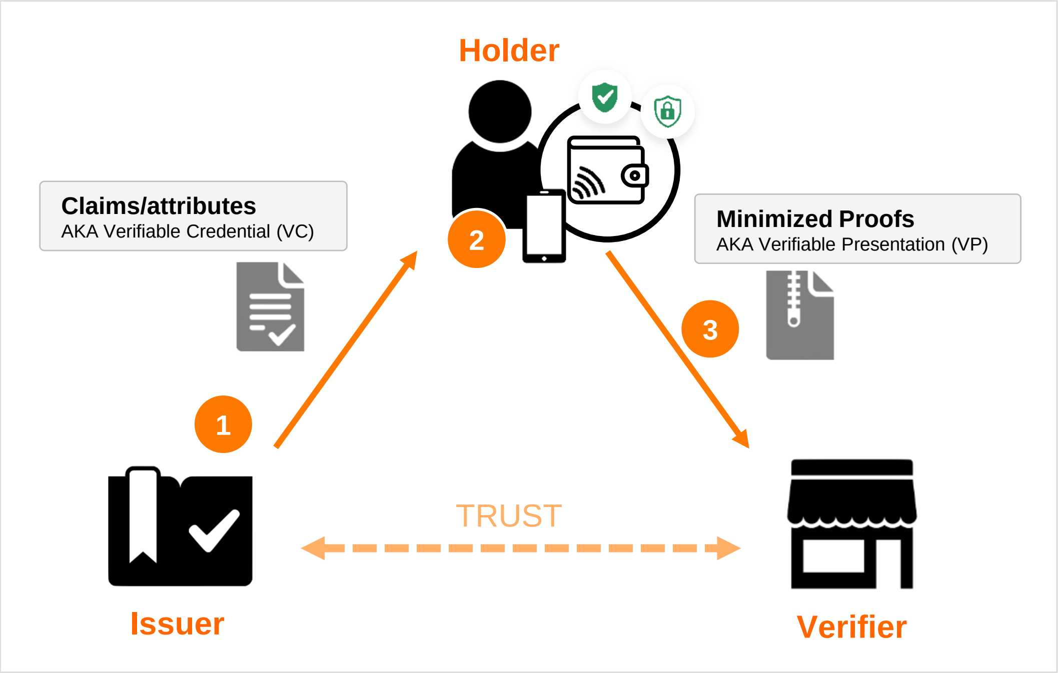
NIST WPEC 2024, September 26, 2024

Jacques Traoré
Antoine Dumanois
Orange Innovation

High-level eIDAS functional overview

eIDAS: *Electronic Identification, Authentication, and Trust Services* is an EU regulation that establishes a legal framework for secure and trustworthy electronic interactions and introducing the **European Digital Identity Wallet**, which allows citizens to securely verify their identity and access services across the EU.

ARF: *Architecture Reference Framework* is a standardized framework that provides guidelines and best practices for designing and managing architectures in compliance with eIDAS requirements. Its main goal is to ensure interoperability between different systems while maintaining consistency and alignment with business objectives.



Holder

Claims/attributes
AKA Verifiable Credential (VC)

Minimized Proofs
AKA Verifiable Presentation (VP)

TRUST

Issuer

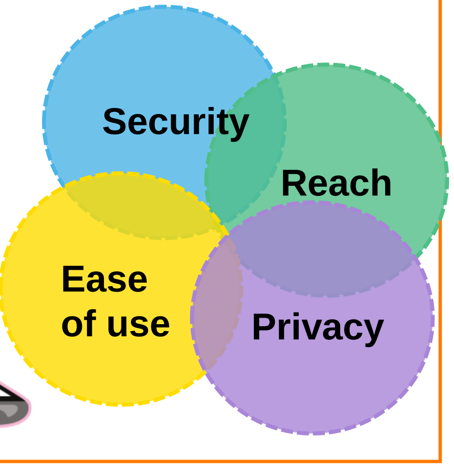
Verifier

1. The **Issuer** generates a VC containing attributes about the Holder and its public key, all signed with the Issuer private key
2. The **Holder** then stores these VCs and can use and combine them to generate proofs to be presented to Verifiers
3. Upon request, the Holder independently presents this proof to the **Verifier**, who can validate it using the Issuer's public key and the Holder's public key embedded in the VC.

Key features for the success of eIDAS 2.0 wallet

Conditions for the success of wallets

- For eIDAS to succeed, it must simultaneously possess all key characteristics: security, privacy, reach, and user experience.
- The civil society is expecting solutions that **meet its expectations**.



Nov 2026

Deployment of SSI wallets

States are required to provide SSI wallets to citizens by November 2026.

High security requirements

eIDAS wallets must meet stringent security standards, achieving a high certification level.

Compliance with state-of-the-art privacy

Importance of adhering to the latest privacy standards, ensuring full unlinkability, plausible deniability, and everlasting privacy.

Reach and User Experience

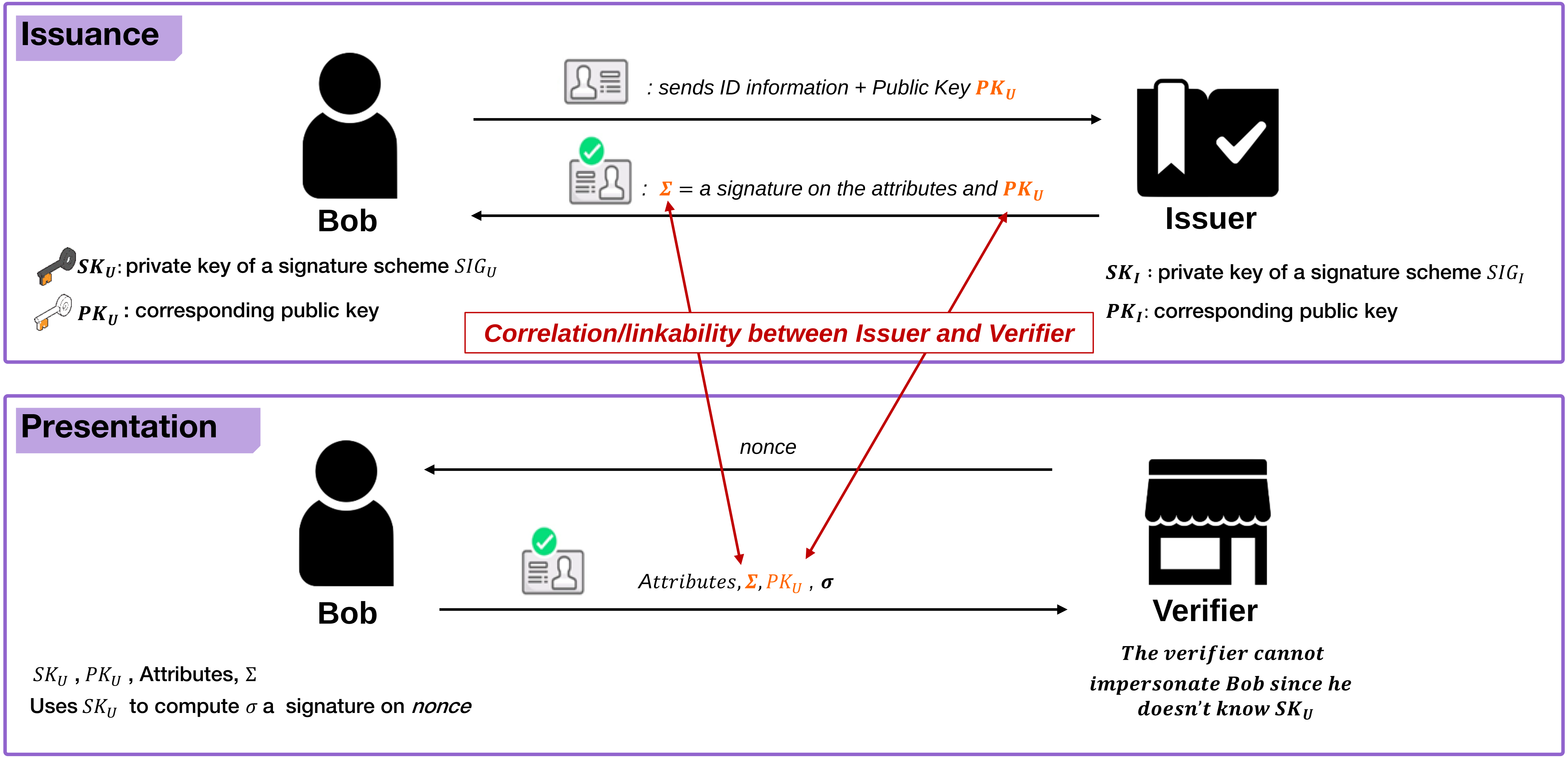
Wallets must ensure a broad reach and provide a seamless user experience with minimal transaction times.

Note: The European Commission has chosen **ISO mDL** (ISO/IEC 18013-5) as the primary protocol.

Challenge

The ARF lacks a solution that fully meets these requirements, particularly in delivering a state-of-the-art way of **combining** privacy and security.

The classical approach to Verifiable Credentials



Issuance

Bob

: *sends ID information + Public Key* PK_U

: Σ = *a signature on the attributes and* PK_U

Issuer

SK_U: private key of a signature scheme SIG_U

PK_U : corresponding public key

SK_I : private key of a signature scheme SIG_I

PK_I: corresponding public key

Correlation/linkability between Issuer and Verifier

Presentation

Bob

nonce

$Attributes, \Sigma, PK_U$, σ

Verifier

SK_U , PK_U , Attributes, Σ

Uses SK_U to compute σ a signature on *nonce*

The verifier cannot impersonate Bob since he doesn't know SK_U

Anonymous credentials: the Swiss knife for eIDAS 2.0?

Definition

Anonymous credentials (AC) are digital credentials that are issued to holders allowing them to prove statements about their identity in a privacy preserving way. More specifically, the holders can present the same AC multiple times to verifiers while keeping the presentations unlinkable/anonymous.

- Make use of specific signatures schemes (so-called ZKP-enabled signature schemes)
- With such schemes, generating and verifying NIZK like the following one, can be done **very efficiently**

$$\mathrm{NIZK}[x = ipk, \omega = \{att, cred\}: \mathrm{Verify}(ipk, cred, att)]$$

where $\boldsymbol{x}$ is the issuer's public key, $\boldsymbol{cred}$ the credential issued on the attributes $\boldsymbol{att}$

While anonymous credentials offer strong privacy protection, the solution we choose must also comply with the strict security requirements set by European security agencies. Specifically, it is crucial that:

- **No pairing-based cryptography** is used, as it does not meet the security standards of certain regulatory bodies.
- The **holder's binding signature** is implemented according to a **SOG-IS compliant protocol**, ensuring that the solution aligns with European cybersecurity frameworks and standards.

The BBS line of protocols and the compliance challenge

> **BBS** is recognized as the **most mature** and **efficient protocol** for anonymous credentials in terms of computational speed and space requirements.

Compliance issues with mDL

- **Incompatibility:** Current BBS/BBS+ protocols are not compliant with mDL due to the following reasons:
    - **Trust model:** The mDL trust model mandates separate holder and issuer signatures.
    - **Data model**: BBS/BBS+ are not compatible with the MSO data structure.
    - **Use of pairings:** BBS relies on cryptographic pairings, which are not accepted by European security agencies.

Rejection by the European Commission

- A group of cryptographers proposed BBS/BBS+ for compliance, but it was rejected by the Commission due to these critical non-compliance issues (use of non-certified pairing friendly curves, non-SOG-IS compliant holder binding).

Conclusion

Current versions of BBS/BBS+ do not meet the necessary compliance standards and therefore are not suitable. BBS+ is great for privacy, but not up to the mark in other areas.

BBS#

A look under the hood

In red: specific to BBS
In green: specific to BBS# a.k.a MAC_{BBS}

BBS/BBS#: signatures

- Prime p
- Group $G_1 \neq G_2 \neq G_T$ of order p
- Pairing $e: G_1 \times G_2 \rightarrow G_T$
- Generators $g_1, h, h_1, h_2, \dots, h_L \in G_1$ and $g_2 \in G_2$

- Private key: $x \in Z_p$
- Public key: $PK_I = g_2^x$ or $PK_I = h^x$
- Sign messages $M = (m_1, m_2, \dots, m_L)$
  1. Sample $e \in Z_p$
  2. Compute A as
  3. Let $Com = g_1 h_1^{m_1} h_2^{m_2} \cdots h_L^{m_L}$
  4. Observe that $A^x = ComA^{-e}$
  5. Let $B = ComA^{-e} = A^x$
  6. Compute $\hat{\pi}$ =ZKP$\{(x): B = A^x \wedge PK_I = h^x\}$

→

$$A = \left(g_1 h_1^{m_1} h_2^{m_2} \cdots h_L^{m_L}\right)^{\frac{1}{x+e}}$$

Signature on $M \rightarrow (A, e)$ or $(\mathbf{A}, \mathbf{e}, \hat{\boldsymbol{\pi}})$

The DL equality proof $\hat{\pi}$ can be issued « anonymously » and « obliviously » on a randomized version (A^l, B^l) of the « signature » (A, B) (Orrù et al. Crypto 2024)

In red: specific to BBS
In green: specific to BBS# a.k.a MAC_{BBS}

BBS/BBS#: verification of a signature

- Prime p
- Group $G_1 \neq G_2 \neq G_T$ of order p
- Pairing $e: G_1 \times G_2 \rightarrow G_T$
- Generators $g_1, h, h_1, h_2, \ldots, h_L \in G_1$ and $g_2 \in G_2$

- Private key $x \in Z_p$, Public key $PK_I = g_2^x$ or $PK_I = h^x$
- Messages $M = (m_1, m_2, \ldots, m_L)$
- $Com = g_1 h_1^{m_1} h_2^{m_2} \cdots h_L^{m_L}$

Signature on M : $(A = Com^{\frac{1}{x+e}}, e)$

Signature on M : $(A = Com^{\frac{1}{x+e}}, e, \hat{\pi})$

Check whether $e(A, g_2^e PK_I) = e(Com, g_2)$

Verify the ZKP proof $\hat{\pi}$ with PK_I

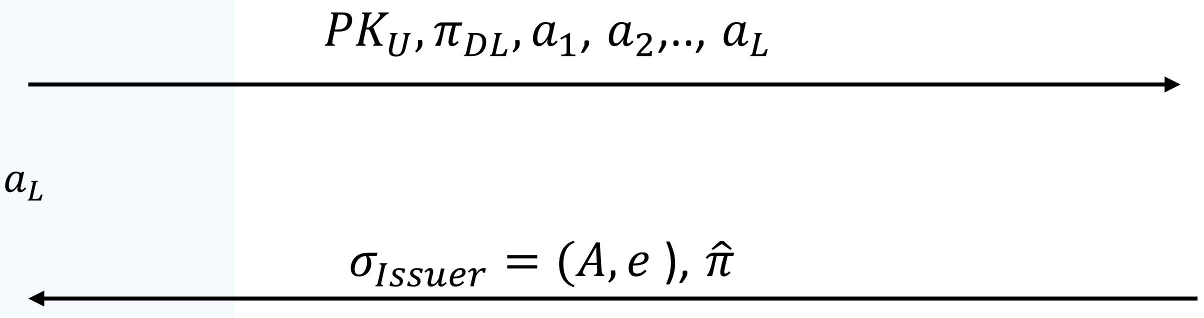
▶ No pairing computations are required on the Verifier's side with MAC_{BBS}

BBS#: blind issuance of the holder binding private key

Public parameters: $g, g_1, h, h_1, h_2,.., h_L$, L+2 generators of G_1 (a cyclic group of order p)

Issuer's private key: $x \in [1, p]$ of BBS#

Issuer's public key: $PK_I = h^x$



$PK_U, \pi_{DL}, a_1, a_2,.., a_L$

$\sigma_{Issuer} = (A, e), \hat{\pi}$

User

Attributes: $a_1, a_2,.., a_L$

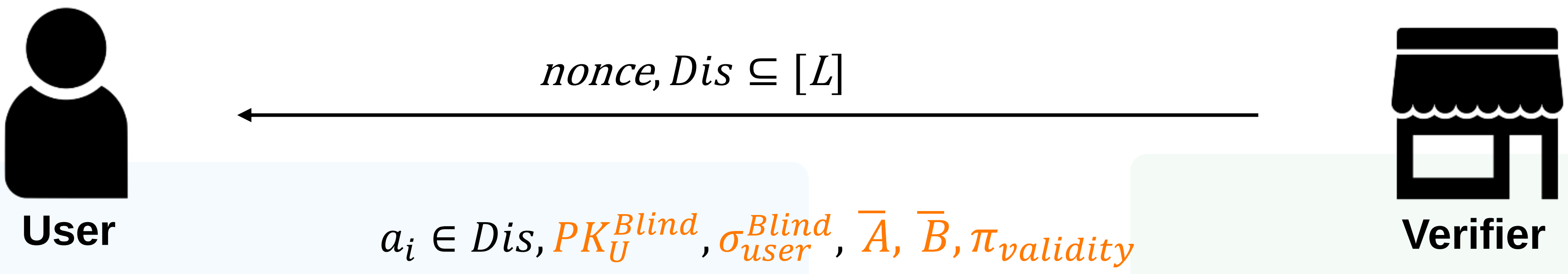
1. Computes $PK_U = g^s$
2. Computes a ZKP π_{DL} of s
3. Verifies $\hat{\pi}$

User's public key = PK_U

Issuer

1. Computes $Com = g_1 \, PK_U h_1^{a_1} h_2^{a_2} \dots h_L^{a_L}$
2. Computes $A = Com^{1/(x+e)}$
3. Computes $\hat{\pi}$ (see previous slide) a proof of validity of $\sigma_{Issuer} = (A, e)$ on the a_i's

BBS#: Verifiable Presentation



$nonce, Dis \subseteq [L]$

User

$a_i \in Dis, PK_U^{Blind}, \sigma_{user}^{Blind}, \bar{A}, \bar{B}, \pi_{validity}$

Verifier

$PK_U = g^s, \sigma_I = (A, e)$

Attributes: $a_1, a_2,.., a_L$

1. **Chooses** $r, l \in [1, p]$
2. **Randomizes** the public key PK_U: $PK_U^{Blind} = g^{s\times r}$
3. **Computes** an ECDSA signature σ_{user}^{Blind} on $nonce$ and PK_U^{Blind} using $SK_{Blind} = s \times r \bmod p$
4. **Randomizes** the VC's signature: ($\bar{A} = A^l, \bar{B} = (ComA^{-e})^l = \bar{A}^x$)
5. **Computes a ZKP $\pi_{validity}$** proving that the user knows a value r and a MAC_{BBS} signature on $(PK_U^{Blind})^{r^{-1}}$ and the a_i's in Dis

$PK_I = h^x$

1. Checks that σ_{user}^{Blind} is valid on *nonce* and PK_U^{Blind} using PK_U^{Blind}
2. **Checks that $\bar{B} = \bar{A}^x$ (Issue 2)**
3. Checks that the ZKP $\pi_{validity}$ is valid using $\bar{A}$, $\bar{B}$ and PK_I

▶ **Issue 1**: current WSCD cannot "randomize" their own public and private keys because they have not been programmed to perform such operations

Secure splitting with ECDSA:

Joint computation of PK_U^{Blind} and σ_{user}^{Blind}

WSCD (HSM)

$PK_U = g^s$

1. **Chooses** $a \in [1, p]$
2. **Computes** $T = g^a = (i, j)$
3. **Computes** $x = i \ (mod\ p)$
4. **Computes** $\rho = a^{-1} \times (M + s \times x)\ mod\ p$
5. **Let** $\sigma_{user} = (x, \rho)$

σ_{user} is an ECDSA signature on M that can be checked using PK_U

$M = r^{-1} \times \mathcal{H}(nonce, PK_U^{Blind})$

$\sigma_{user} = (x, \rho)$

Wallet

$nonce, Dis \subseteq [L]$

Verifier

1. **Chooses** $r \in [1, p]$
2. **Computes** $PK_U^{Blind} = PK^r = g^{s \times r}$
3. **Computes** $\rho_{Blind} = \rho \times r \ (mod\ p)$
4. **Let** $\sigma_{user}^{Blind} = (x, \rho_{Blind})$

σ_{user}^{Blind} is an ECDSA signature on *nonce* and PK_U^{Blind} that can be checked using PK_U^{Blind}

We make use of raw ECDSA (i.e., the signing queries are on $\mathcal{H}(m)$ instead of plain m) which is supported by a majority of WSCD ECDSA signature scheme is unforgeable, in the **elliptic curve GGM**, even if the adversaries makes raw signing queries [Groth and Shoup, EC'22]

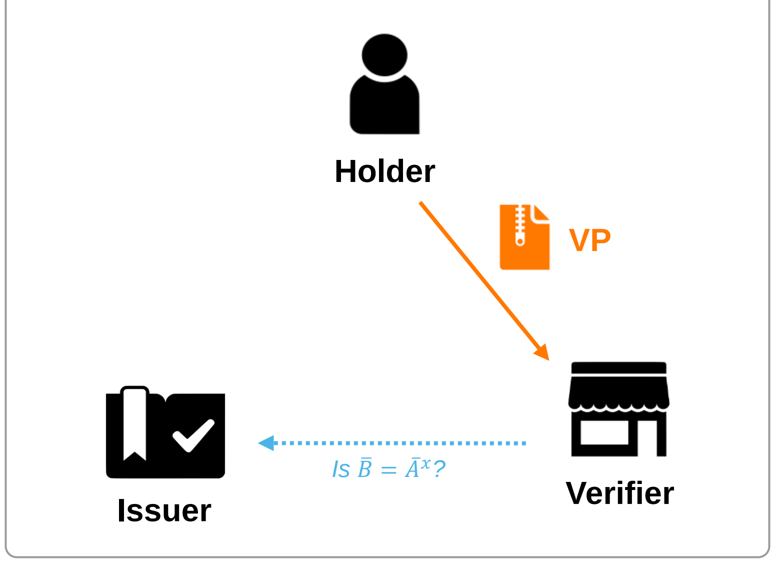
Verification: how to check that $\overline{B} = \overline{A}^x$?

With classic **BBS** protocol styles, pairings are used.

With **BBS#,** pairings could be used, but there is also a pairing-free option.

- To be **pairing-free**, assistance from the issuer is required. There are several ways to achieve this:

Option #1



Holder

VP

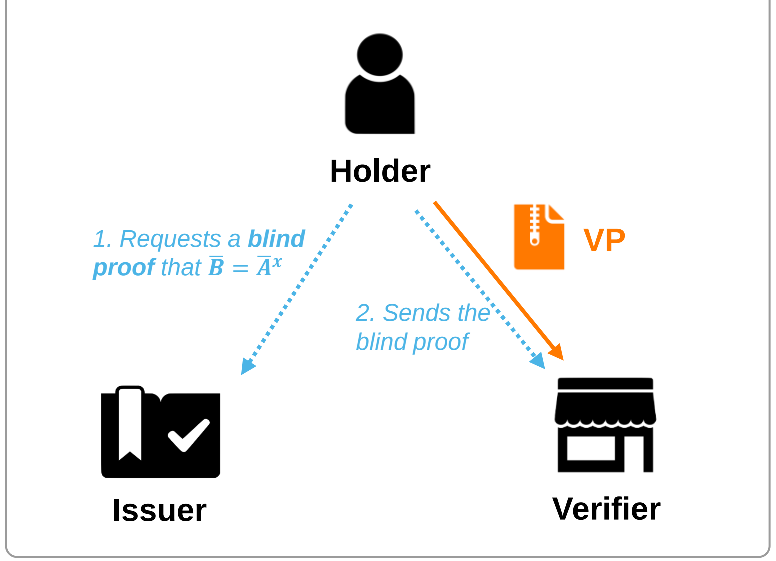
Issuer

Verifier

Is $\overline{B} = \overline{A}^x$?

The Verifier requests assistance from the Issuer **each time**.

Option #2



Holder

*1. Requests a **blind proof** that $\overline{\boldsymbol{B}} = \overline{\boldsymbol{A}}^x$*

2. Sends the blind proof

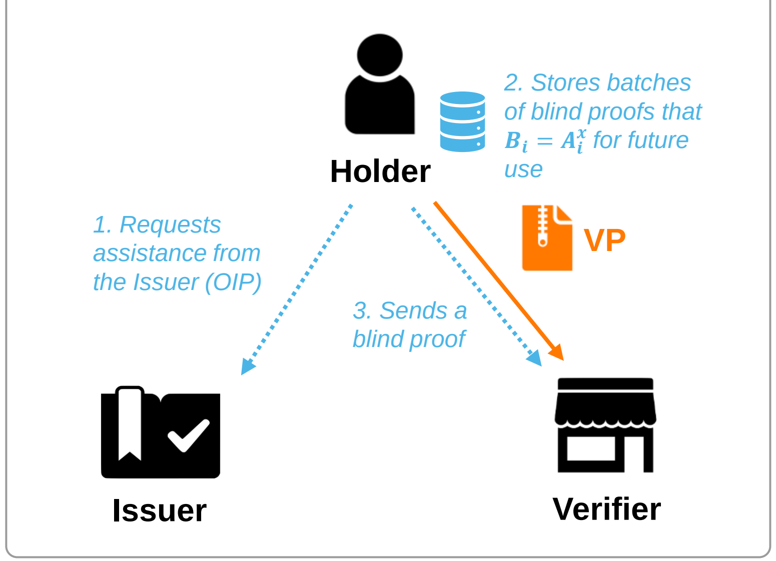
VP

Issuer

Verifier

The Holder requests **anonymously** an **Oblivious Issuance Proof** (Crypto 2024) that $\overline{\boldsymbol{B}} = \overline{\boldsymbol{A}}^x$ from the Issuer each time.

Option #3



Holder

2. Stores batches of blind proofs that $\boldsymbol{B_i} = \boldsymbol{A_i^x}$ for future use

1. Requests assistance from the Issuer (OIP)

3. Sends a blind proof

VP

Issuer

Verifier

The Holder generates in advance several pairs $(A_i = A^{l_i}, B_i = B^{l_i})$ and requests from the Issuer, in advance, blind proofs (OIP) that $\boldsymbol{B_i} = \boldsymbol{A_i^x}$ and stores these blind proofs for future use.

Option 3 is closer to what already exists in mDL.

Unlinkability applies to all three options

Attribute-based credential protocols in practice (I)

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. Argo et al., ACM CCS 2024
3. For $U = 10$

Attribute-based credential protocols in practice (II)

	Time Efficiency *			
	Credential Issuance[1]	**Presentation WSCD part**	**Presentation Wallet part**	**Presentation Verification**
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]	**400 ms**	**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

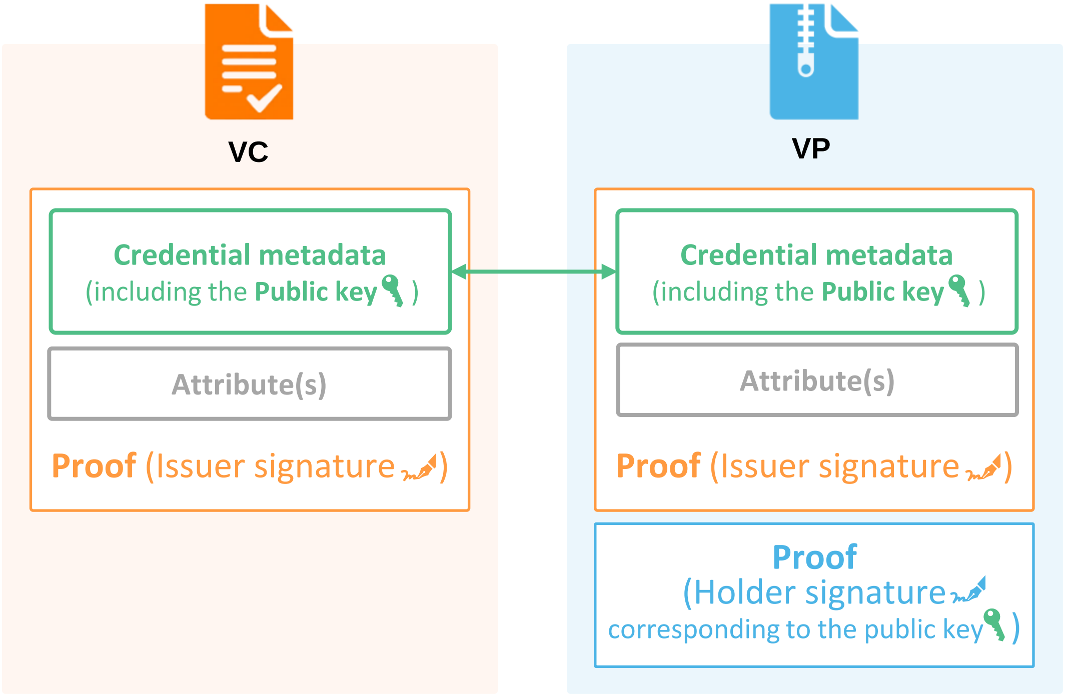
1. N =10
2. with ECDSA used on both the Holder and Issuer's side
3. Benchmarked on an Intel Core i7 12800H CPU running at 4.6 GHz
4. [AGJ+24]
5. Current WSCD do not support the computations involved in Argo et al., ACM CCS 2024

Take away

- eIDAS wallets must meet stringent security standards, ensuring robust protection against all threats
- Current efficient Anonymous Credentials protocols such as CL, PS or BBS/BBS+, do not meet these requirements: they either make use of pairings or pairing-friendly curves and/or are not supported by current certified secure elements.
- BBS# is a variant of BBS which:
  - can be used with ‘’classic’’ (non pairing-friendly) elliptic curves, and thus with current WSCD (iOS/Secure Enclave, Android-/HBK+Strongbox, TPMs, PKCS11 based HSMs).
  - supports all privacy features of the BBS family of protocols (full unlinkability, everlasting privacy), and more, e.g.: plausible deniability (both for issuance and presentation).
- BBS# is provably secure; it inherits the security of BBS (Eurocrypt 2023), of Oblivious Issuance Proofs (Crypto 2024) and the security of ECDSA with multiplicative key randomization (ACM CCS 2019).
- BBS# allows Selective Disclosure in both offline and online modes.
- BBS# is compatible with ISO mDL.

Thank you

Structure of a VC and privacy challenges



VC

Credential metadata
(including the **Public key** 🔑)

Attribute(s)

Proof (Issuer signature ✍)

VP

Credential metadata
(including the **Public key** 🔑)

Attribute(s)

Proof (Issuer signature ✍)

Proof
(Holder signature ✍
corresponding to the public key 🔑)

Verifiable Credential (VC) Structure

- **VC Issuance**
  The VC is issued by the issuer and signed using the **issuer's private key**.
- **Verification Process**
  The corresponding public key 🔑 is used by the verifier to confirm the authenticity of the VC.

Holder's Presentation (VP)

- The VC is then used by the holder to create a VP, which is sent to the verifier.
- The VP includes both:
  - **Issuer's signature** (to verify data) ✍
  - **Holder's signature** ✍ (holder binding signature) linked to the holder's private key, ensuring the authenticity of the presentation

Privacy challenge: Correlation between Verifier and Issuer

Even though pairwise VCs can be created to avoid **correlation between different verifiers**, it is **not possible to avoid correlation between verifier and issuer**. At least, 2 elements can be used for tracking between issuers and verifiers (very often there are even more metadata that can be used for tracking).

- The **same issuer signature** is used on both the VC and VP.
- The **same public key** is used on both the VC and VP.

Conclusion

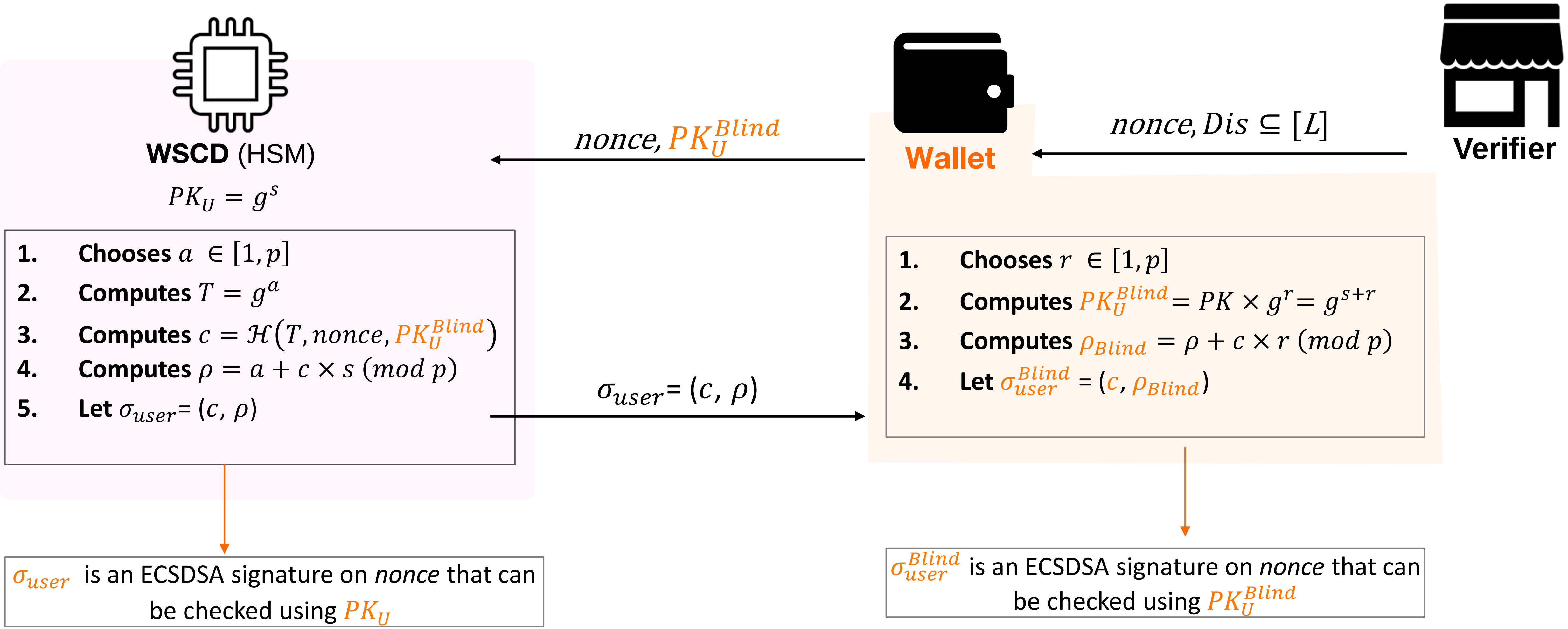
- The classic model ensures security through digital signatures, but the tracking of metadata can create significant privacy concerns related to the tracking across verifiers and between verifiers and issuers.
- Current solutions like pairwise verifier-issuer VCs don't solve the issuer-verifier correlation issue and create scalability and management overhead.

MSO-mDL VS. BBS+ VS. BBS#

	MSO	BBS+	BBS#
Privacy	Partial unlinkability (Verifier/Verifier) with batch VC issuance	Unlinkability	Unlinkability
	No Everlasting privacy	Everlasting privacy	Everlasting privacy
		Blind signature	Blind signature
	Data minimisation - SD	Data minimisation - SD	Data minimisation - SD
	Non-revocation - status lists	Non-revocation - accumulators	Non-revocation - accumulators
	Date of expiration in clear	Non expiration accumulator	Non expiration accumulator
		Pseudonyms	Pseudonyms
			Plausible deniability - VC / issuance - VP / presentation
			Issuer unlinkability
Implementability		Most efficient anonymous credentials protocol	Even more efficient
		Splitting	Efficient splitting (ECSchnorr or plainECDSA)
	Uses classic elliptic curve	Pairing	Must not use pairing Can use classic elliptic curve
	One holder binding per VC	One holder binding per VC	Linking all VCs with a single holder binding (and single authent.)
	Implementable on HSMs and SE	Implementable on customized HSMs	Implementable today on vanilla HSMs or even SEs (plainECDSA)
	Signatures in COSE format	Specific format for holder binding	Compatible with the usual issuer signature + holder signature
	Compatible with ISO/IEC 8013-5	Compatible with JSON-LD	JSON-LD + mDL & SD-JWT compatibility
	One WSCD secret key per VC (to avoid verifier tracking)	Use of a single WSCD secret key for all issuers, verifiers, VCs, VPs etc.	Use of a single WSCD secret key for all issuers, verifiers, VCs, VPs etc.
	Classic issuer signature	BBS+ signature (new implementation required)	BBS# signature (new implementation required)
	Classic verifier libraries	BBS+ signature (new libraries required)	BBS+ signature (new libraries required)
Certifiability	Use classic elliptic curve	Pairing	Must not use pairing Can use classic elliptic curve
	Holder binding using SOG-IS protocols	Holder binding	Holder binding with ECSchnorr or plainECDSA signature (in SOG-IS)
		Possible deanonimisation (loss of everlasting privacy)	Possible deanonimisation (loss of everlasting privacy)
	Sensitive to non authenticating WSCD takeover	Sensitive to non authenticating WSCD takeover	Immunity to WSCD takeover when leveraging non-authenticating WSCA
	Usual protocols for issuers and verifiers	New certification required? For issuers? For verifiers?	New certification required? For issuers? For verifiers?

Splitting with EC-Schnorr :

Joint computation of PK_U^{Blind} and σ_{user}^{Blind}



WSCD (HSM)

$PK_U = g^s$

1. **Chooses** $a \in [1, p]$
2. **Computes** $T = g^a$
3. **Computes** $c = \mathcal{H}(T, nonce, PK_U^{Blind})$
4. **Computes** $\rho = a + c \times s \ (mod\ p)$
5. **Let** $\sigma_{user} = (c,\ \rho)$

$nonce, PK_U^{Blind}$

$\sigma_{user} = (c,\ \rho)$

Wallet

$nonce, Dis \subseteq [L]$

Verifier

1. **Chooses** $r \in [1, p]$
2. **Computes** $PK_U^{Blind} = PK \times g^r = g^{s+r}$
3. **Computes** $\rho_{Blind} = \rho + c \times r \ (mod\ p)$
4. **Let** $\sigma_{user}^{Blind} = (c,\ \rho_{Blind})$

σ_{user} is an ECSDSA signature on *nonce* that can be checked using PK_U

σ_{user}^{Blind} is an ECSDSA signature on *nonce* that can be checked using PK_U^{Blind}

ABC protocols in a pre and post quantum world

	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

Before Q-Day:

BBS# is better suited in terms of security (relies on a well-known hardness assumption) and privacy (everlasting privacy) compared to the other alternatives

After Q-Day:

SD-JWT and mDL (but using PQ signature schemes (assuming they are WSCD ready) on both the holder and issuer's side) seems preferable as PQ-ABC alternatives won't probably scale after Q-Day (i.e. won't be WSCD ready), however we will lose privacy.