Improved Constructions of Anonymous Credentials From Structure-Preserving Signatures on Equivalence Classes

Aisling Connolly[1†] Pascal Lafourcade [‡]
Octavio Perez Kempner [§,¶]

[†]DFINITY

[‡]University Clermont Auvergne, LIMOS, France

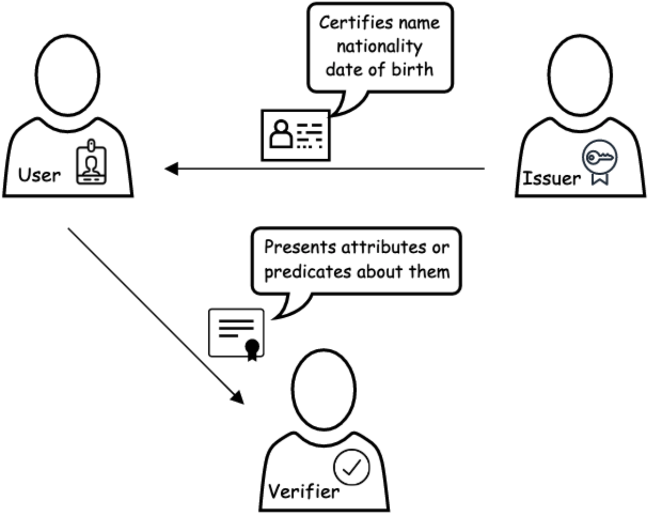
[§]DIENS, École normale supérieure, CNRS, PSL University, Paris, France

[¶]be-ys Research, France

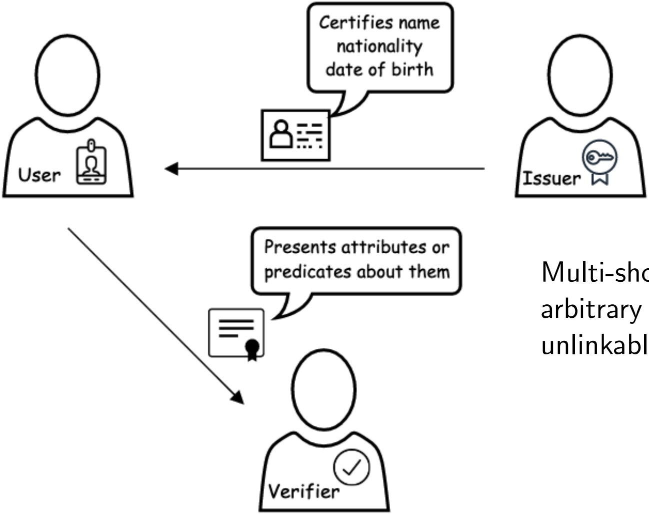
[1]Work done while the author was at Wordline Global.

Agenda

Attribute-based Credentials

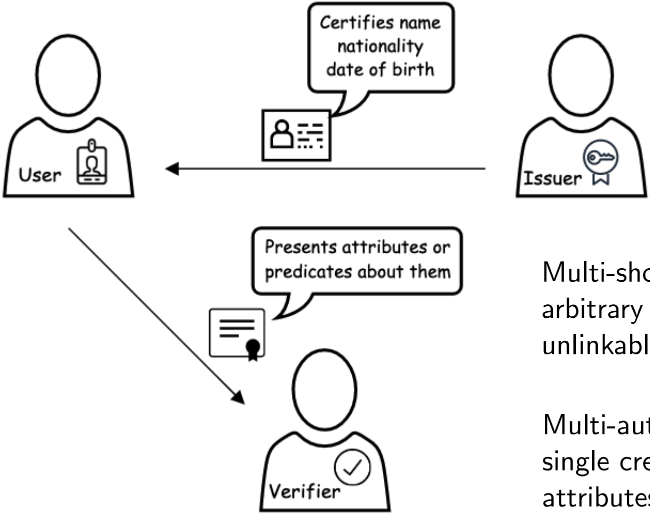


Attribute-based Credentials



Multi-show ABC's:
arbitrary number of
unlinkable showings

Attribute-based Credentials



Multi-show ABC's: arbitrary number of unlinkable showings

Multi-authority ABC's: single credential for attributes issued by multiple authorities

Attribute-based Credentials: Differences

Expressiveness

Expressiveness

Efficiency

Attribute-based Credentials: Differences

Expressiveness

Efficiency

Communication

Expressiveness

Efficiency

Communication

Security model

Attribute-based Credentials: Differences

Expressiveness

Efficiency

Communication

Security model

Revocation

- CL signatures [**?**]: Idemix [**?**] and [**?**]

- CL signatures [**?**]: Idemix [**?**] and [**?**]
- Aggregatable signatures: [**?**] and [**?**]

- CL signatures [**?**]: Idemix [**?**] and [**?**]
- Aggregatable signatures: [**?**] and [**?**]
- Sanitizable signatures: [**?**]

- CL signatures [?]: Idemix [?] and [?]
- Aggregatable signatures: [?] and [?]
- Sanitizable signatures: [?]
- Redactable signatures: [?] and [?]

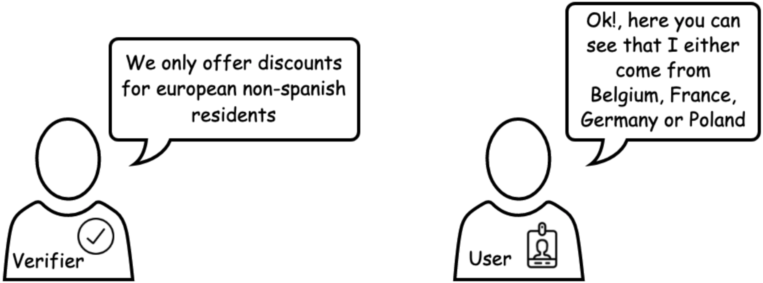
- CL signatures [?]: Idemix [?] and [?]
- Aggregatable signatures: [?] and [?]
- Sanitizable signatures: [?]
- Redactable signatures: [?] and [?]
- Structure-Preserving Signatures on Equivalence Classes (SPS-EQ): [?], [?] and [?]

- CL signatures [?]: Idemix [?] and [?]
- Aggregatable signatures: [?] and [?]
- Sanitizable signatures: [?]
- Redactable signatures: [?] and [?]
- Structure-Preserving Signatures on Equivalence Classes (SPS-EQ): [?], [?] and [?]
- All previous constructions leak the issuer's identity

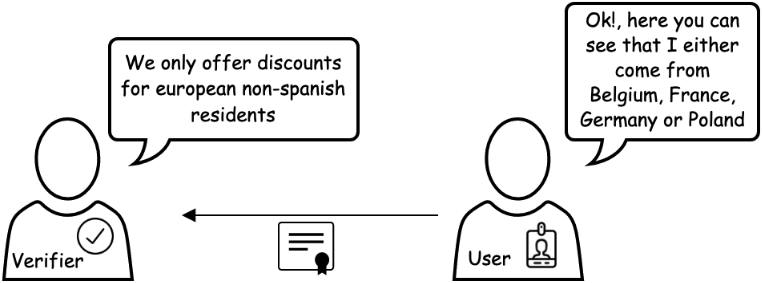
Motivation: self-sovereign identity across Europe


We only offer discounts for european non-spanish residents
Verifier
User

Motivation: self-sovereign identity across Europe



Motivation: self-sovereign identity across Europe



Agenda

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \rightarrow (\sigma', m')$

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \rightarrow (\sigma', m')$
- Message space can be partitioned into equivalence classes

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \to (\sigma', m')$
- Message space can be partitioned into equivalence classes
  - *e.g.,* $m \in \mathbb{G}^\ell \sim_{\mathcal{R}} m' \in \mathbb{G}^\ell \iff \mu \in \mathbb{Z}_p^*$ s.t. $m' = \mu m$

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \to (\sigma', m')$
- Message space can be partitioned into equivalence classes
  - *e.g.*, $m \in \mathbb{G}^\ell \sim_{\mathcal{R}} m' \in \mathbb{G}^\ell \iff \mu \in \mathbb{Z}_p^*$ s.t. $m' = \mu m$
- Unforgeability holds with respect to classes

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \to (\sigma', m')$
- Message space can be partitioned into equivalence classes
  - *e.g.*, $m \in \mathbb{G}^\ell \sim_\mathcal{R} m' \in \mathbb{G}^\ell \iff \mu \in \mathbb{Z}_p^*$ s.t. $m' = \mu m$
- Unforgeability holds with respect to classes
- Message-signature pairs in the same class are unlinkable

SPS-EQ: Intuition

- Controlled form of malleability: $(\sigma, m) \to (\sigma', m')$
- Message space can be partitioned into equivalence classes
  - *e.g.*, $m \in \mathbb{G}^\ell \sim_{\mathcal{R}} m' \in \mathbb{G}^\ell \iff \mu \in \mathbb{Z}_p^*$ s.t. $m' = \mu m$
- Unforgeability holds with respect to classes
- Message-signature pairs in the same class are unlinkable
- Recently extended to consider equivalence classes on the key space (*e.g.*, [**?**, **?**, **?**])

SPS-EQ: Syntax

SPS-EQ: Syntax

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$

SPS-EQ: Syntax

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
- $(m^*, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$

SPS-EQ: Syntax

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
- $(m^*, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$
- $b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk})$

SPS-EQ: Syntax

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
- $(m^*, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$
- $b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk})$
- $\mathsf{sk} \leftarrow \mathsf{ConvertSK}(\mathsf{sk}, \rho)$

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
- $(m^*, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$
- $b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk})$
- $\mathsf{sk} \leftarrow \mathsf{ConvertSK}(\mathsf{sk}, \rho)$
- $\mathsf{pk} \leftarrow \mathsf{ConvertPK}(\mathsf{pk}, \rho)$

- $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$
- $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell)$
- $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
- $(m^*, \sigma^*) \leftarrow \cancel{\mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})}\mathsf{ChgRep}(m, \sigma, \mu, \rho, \mathsf{pk})$
- $b \leftarrow \mathsf{Verify}(m, \sigma, \mathsf{pk})$
- $\mathsf{sk} \leftarrow \mathsf{ConvertSK}(\mathsf{sk}, \rho)$
- $\mathsf{pk} \leftarrow \mathsf{ConvertPK}(\mathsf{pk}, \rho)$

SPS-EQ: Properties

- Unforgeability: Given the set of queries Q that $\mathcal{A}$ issues to the signing oracle $\textsc{Sign}$, the following probability is negligible

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda), \\ (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell), \\ (m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\textsc{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) \end{array} : \begin{array}{l} [m^*]_{\mathcal{R}} \neq [m]_{\mathcal{R}} \ \forall\ m \in \mathsf{Q} \\ \wedge\ \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}) = 1 \end{array}\right]$$

SPS-EQ: Properties

- Perfect adaption of signatures (1)

- Perfect adaption of signatures (1)
    - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:

- Perfect adaption of signatures (1)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:

SPS-EQ: Properties

- Perfect adaption of signatures (1)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m,\ \sigma, \mathsf{pk}) = 1$, the following holds:
    $(\mu m, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mu m, \sigma^*, \mathsf{pk}) = 1$
- Perfect adaption of signatures (2)

SPS-EQ: Properties

- Perfect adaption of signatures (1)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m,\ \sigma, \mathsf{pk}) = 1$, the following holds:
    $(\mu m, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mu m, \sigma^*, \mathsf{pk}) = 1$
- Perfect adaption of signatures (2)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu, \rho)$ s.t. $\mathsf{pp} \stackrel{\$}{\leftarrow} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m,\ \sigma, \mathsf{pk}) = 1$, the following holds:

SPS-EQ: Properties

- Perfect adaption of signatures (1)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:
    $(\mu m, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mu m, \sigma^*, \mathsf{pk}) = 1$
- Perfect adaption of signatures (2)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu, \rho)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:

SPS-EQ: Properties

- Perfect adaption of signatures (1)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:
    $(\mu m, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \mathsf{pk})$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mu m, \sigma^*, \mathsf{pk}) = 1$
- Perfect adaption of signatures (2)
  - For all tuples $(\mathsf{pp}, \mathsf{pk}, m, \sigma, \mu, \rho)$ s.t. $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$ and $\mathsf{Verify}(m, \sigma, \mathsf{pk}) = 1$, the following holds:
    $(\mu m, \sigma^*) \leftarrow \mathsf{ChgRep}(m, \sigma, \mu, \rho, \mathsf{pk})$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mu m, \sigma^*, \mathsf{ConvertPK}(\mathsf{pk}, \rho)) = 1$

Agenda

The ABC framework from [?]

The ABC framework from [?]

- A credential is a **signature** on a (randomizable) accumulator representing a **set of attributes**

The ABC framework from [?]

- A credential is a **signature** on a (randomizable) accumulator representing a **set of attributes**
- A credential showing involves the joint randomization of a message-signature pair

The ABC framework from [?]

- A credential is a **signature** on a (randomizable) accumulator representing a **set of attributes**
- A credential showing involves the joint randomization of a message-signature pair
- The accumulator uses batch membership proofs to allow **constant-size** showings

The ABC framework from [?]

- A credential is a **signature** on a (randomizable) accumulator representing a **set of attributes**
- A credential showing involves the joint randomization of a message-signature pair
- The accumulator uses batch membership proofs to allow **constant-size** showings
- Security properties: **Unforgeability** & **Anonymity**

The ABC framework from [?]

- A credential is a **signature** on a (randomizable) accumulator representing a **set of attributes**
- A credential showing involves the joint randomization of a message-signature pair
- The accumulator uses batch membership proofs to allow **constant-size** showings
- Security properties: **Unforgeability** & **Anonymity**
- Main drawback: **expressiveness is limited**

- We focused on improving the following aspects:

- We focused on improving the following aspects:
  - expressiveness (extending the accumulator)

Towards improved constructions

- We focused on improving the following aspects:
  - expressiveness (extending the accumulator)
  - efficiency (leveraging user/verifier costs)

Towards improved constructions

- We focused on improving the following aspects:
  - expressiveness (extending the accumulator)
  - efficiency (leveraging user/verifier costs)
  - security model (~~GGM~~ Standard model + CRS)

Agenda

Overview of results

- **Obtained a SPS-EQ** acting on the message and key spaces

- **Obtained a SPS-EQ** acting on the message and key spaces
- Extended the accumulator from [?] to **support batch non-membership proofs**

- **Obtained a SPS-EQ** acting on the message and key spaces
- Extended the accumulator from [**?**] to **support batch non-membership proofs**
- Incorporated a proof of exponentiation to **outsource computational cost** from the verifier to the user

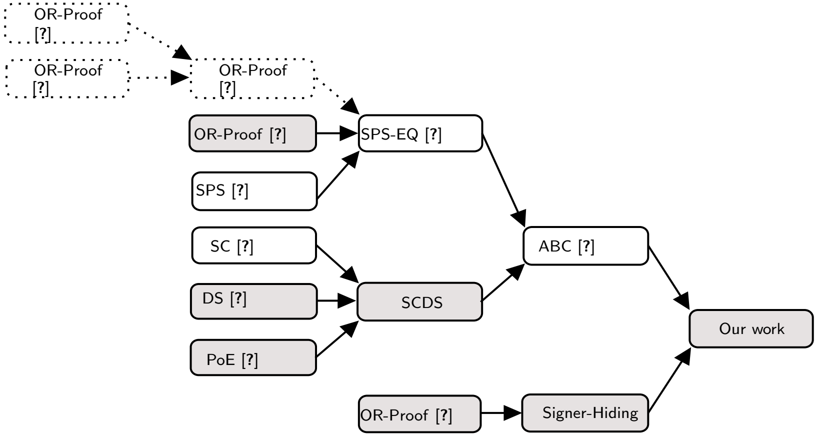
- **Obtained a SPS-EQ** acting on the message and key spaces
- Extended the accumulator from [?] to **support batch non-membership proofs**
- Incorporated a proof of exponentiation to **outsource computational cost** from the verifier to the user
- Proposed a 1-out-of-n NIZK OR-proof so that **users can hide the issuer identity** during a showing

Overview of results

- **Obtained a SPS-EQ** acting on the message and key spaces
- Extended the accumulator from [**?**] to **support batch non-membership proofs**
- Incorporated a proof of exponentiation to **outsource computational cost** from the verifier to the user
- Proposed a 1-out-of-n NIZK OR-proof so that **users can hide the issuer identity** during a showing
- Extended the security model from [**?**]

Overview of results

Overview of results



Agenda

A SPS-EQ from standard assumptions

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space
  - Key idea: use a malleable OR-Proof

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space
  - Key idea: use a malleable OR-Proof
- We extended [?] to support EQ's on the key space

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space
  - Key idea: use a malleable OR-Proof
- We extended [?] to support EQ's on the key space
- We replaced the OR-Proof from [?] using [?]

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space
  - Key idea: use a malleable OR-Proof
- We extended [?] to support EQ's on the key space
- We replaced the OR-Proof from [?] using [?]

A SPS-EQ from standard assumptions

- [?] presented a SPS based on an OR-Proof
- [?] extended [?] to obtain a SPS-EQ for the message space
  - Key idea: use a malleable OR-Proof
- We extended [?] to support EQ's on the key space
- We replaced the OR-Proof from [?] using [?]

Scheme	$\|\sigma\|$	$\|pk\|$	Sign	Verify	ChgRep	Assumptions
[?]	$8\|\mathbb{G}_1\| + 6\|\mathbb{G}_2\|$	$2\|\mathbb{G}_1\| + (9+\ell)\|\mathbb{G}_2\|$	28E	9P	N/A	SXDH
[?]	$8\|\mathbb{G}_1\| + 9\|\mathbb{G}_2\|$	$(2+\ell)\|\mathbb{G}_2\|$	29E	11P	19P+38E	SXDH
Our work	$9\|\mathbb{G}_1\| + 4\|\mathbb{G}_2\|$	$(2+\ell)\|\mathbb{G}_2\|$	10E	11P	19P+21E	extKerMDH, SXDH

A Set Commitment Scheme Supporting Disjoint Sets

- For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$ let
$$\mathsf{Ch}_{\mathcal{X}}(X) = \prod_{x\in\mathcal{X}}(X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$$

A Set Commitment Scheme Supporting Disjoint Sets

- For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$ let

$$\mathrm{Ch}_{\mathcal{X}}(X) = \prod_{x \in \mathcal{X}} (X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$$

- $\mathrm{Ch}_{\mathcal{X}}(s)P$ can be efficiently computed when given $(s^i P)_{i=0}^{|\mathcal{X}|}$

A Set Commitment Scheme Supporting Disjoint Sets

- For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$ let
$$\mathsf{Ch}_{\mathcal{X}}(X) = \prod_{x \in \mathcal{X}}(X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$$
- $\mathsf{Ch}_{\mathcal{X}}(s)P$ can be efficiently computed when given $(s^i P)_{i=0}^{|\mathcal{X}|}$
- To randomize $\mathsf{Ch}_{\mathcal{X}}(s)P$ multiply by $\rho \xleftarrow{\$} \mathbb{Z}_p^*$

A Set Commitment Scheme Supporting Disjoint Sets

- For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$ let
$$\mathsf{Ch}_{\mathcal{X}}(X) = \prod_{x \in \mathcal{X}}(X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$$
- $\mathsf{Ch}_{\mathcal{X}}(s)P$ can be efficiently computed when given $(s^i P)_{i=0}^{|\mathcal{X}|}$
- To randomize $\mathsf{Ch}_{\mathcal{X}}(s)P$ multiply by $\rho \xleftarrow{\$} \mathbb{Z}_p^*$
- Schwartz-Zippel: Let $q_1(x), q_2(x)$ be two d-degree polynomials from $\mathbb{Z}_p[X]$ with $q_1(x) \neq q_2(x)$, then for $s \xleftarrow{\$} \mathbb{Z}_p$,
$$\Pr[q_1(s) = q_2(s)] \text{ is at most } d/p$$

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership:

- Batch membership:

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership:

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{S}}(s)P_1$
- Batch non-membership:

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X} \setminus \mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases
- Proof of exponentiation: the prover computes $\mathsf{Ch}_{\mathcal{S}}(X)$ and sends a proof of correctness

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases
- Proof of exponentiation: the prover computes $\mathsf{Ch}_{\mathcal{S}}(X)$ and sends a proof of correctness
  - Let $Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$ and $h(X)$ and β s.t. $\mathsf{Ch}_{\mathcal{X}}(X){=}(X+\alpha)\cdot h(X)+\beta$

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases
- Proof of exponentiation: the prover computes $\mathsf{Ch}_{\mathcal{S}}(X)$ and sends a proof of correctness
  - Let $Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$ and $h(X)$ and β s.t. $\mathsf{Ch}_{\mathcal{X}}(X){=}(X+\alpha) \cdot h(X) + \beta$
  - Let $\pi_Q \leftarrow h(s)P_2$

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases
- Proof of exponentiation: the prover computes $\mathsf{Ch}_{\mathcal{S}}(X)$ and sends a proof of correctness
  - Let $Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$ and $h(X)$ and β s.t. $\mathsf{Ch}_{\mathcal{X}}(X) = (X+\alpha) \cdot h(X) + \beta$
  - Let $\pi_Q \leftarrow h(s)P_2$
  - Send (π_Q, Q)

A Set Commitment Scheme Supporting Disjoint Sets

- Batch membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \subseteq \mathcal{X}$
  - Witness: $\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
- Batch non-membership: $\mathsf{Ch}_{\mathcal{S}}(X)$ with $\mathcal{S} \cap \mathcal{X} = \emptyset$
  - Witness: use the EEA to compute $q_1(X)$ and $q_2(X)$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) \cdot q_1(X) + \mathsf{Ch}_{\mathcal{S}}(X) \cdot q_2(X) = 1$
- $\mathsf{Ch}_{\mathcal{S}}(X)$ has to be computed by the verifier in both cases
- Proof of exponentiation: the prover computes $\mathsf{Ch}_{\mathcal{S}}(X)$ and sends a proof of correctness
  - Let $Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$ and $h(X)$ and β s.t. $\mathsf{Ch}_{\mathcal{X}}(X){=}(X+\alpha)\cdot h(X)+\beta$
  - Let $\pi_Q \leftarrow h(s)P_2$
  - Send (π_Q, Q)
  - Use $\beta \leftarrow \mathsf{Ch}_{\mathcal{S}}(X) (\mathrm{mod}\ (X+\alpha))$

Signer-Hiding

- Main idea:

- Main idea:
  - Randomize the credential and issuer's public-key consistently

Signer-Hiding

- Main idea:
  - Randomize the credential and issuer's public-key consistently
  - Present them to the verifier alongside a proof of correct randomization of issuer's public-key

Signer-Hiding

- Main idea:
  - Randomize the credential and issuer's public-key consistently
  - Present them to the verifier alongside a proof of correct randomization of issuer's public-key
- The 1-out-of-n OR-proof is a fully adaptive NIZK argument

- Main idea:
  - Randomize the credential and issuer's public-key consistently
  - Present them to the verifier alongside a proof of correct randomization of issuer's public-key
- The 1-out-of-n OR-proof is a fully adaptive NIZK argument
- Users can select arbitrary long sets of public keys to compute a proof with linear cost

Signer-hiding: Formalization

An ABC system supports signer-hiding if for all $\lambda > 0$, all $q > 0$, all $n > 0$, all $t > 0$, all $\mathcal{X}$ with $0 < |\mathcal{X}| \leq t$, all $\emptyset \neq \mathcal{S} \subset \mathcal{X}$ and $\emptyset \neq \mathcal{D} \not\subseteq \mathcal{X}$ with $0 < |\mathcal{D}| \leq t$, and p.p.t adversaries $\mathcal{A}$, the following holds

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ \forall\ i \in [n] : (\mathsf{osk}_i, \mathsf{opk}_i) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UsrKGen}(\mathsf{pp}); j \xleftarrow{\$} [n]; \\ (\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}_j, \mathcal{X}), \\ \qquad\qquad\qquad \mathsf{Issue}(\mathsf{upk}, \mathsf{osk}_j, \mathcal{X})); \\ j^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Show}}}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \mathsf{opk}_i)_{i \in [n]}) \end{array} \quad : \quad j^* = j \right] \leq \frac{1}{n}$$

Agenda

Proposed ABC construction

ABC.Obtain(pp, usk, opk, $\mathcal{X}$)		ABC.Issue(pp, upk, osk, $\mathcal{X}$)
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$; $a \leftarrow r_1 P_1$		
$c \leftarrow \mathsf{Commit}(\mathsf{ck}, a, r_2)$	$\xrightarrow{c}$	
$z \leftarrow r_1 + e \cdot \mathsf{usk}$	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathbb{Z}_p^*$
$(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{scds_{pp}}, \mathcal{X}; \mathsf{usk})$	$C, R,$	
$r_3 \xleftarrow{\$} \mathbb{Z}_p^*$; $R \leftarrow r_3 C$	$\xrightarrow{z, a, r_2}$	**if** $(zP_1 \neq a + e \cdot \mathsf{upk} \ \vee \ c \neq \mathsf{Commit}(\mathsf{ck}, a, r_2))$
		return $\perp$
		if $(e(C, P_2) \neq e(\mathsf{upk}, \mathsf{Ch}_{\mathcal{X}}(s) P_2)$
		$\wedge \ \forall \ x \in \mathcal{X} : xP_1 \neq \mathsf{ek}_1^0)$ **return** $\perp$
	$\xleftarrow{(\sigma, \tau)}$	$(\sigma, \tau) \leftarrow \mathsf{SPS\text{-}EQ.Sign}(\mathsf{sps_{pp}}, (C, R, P_1), \mathsf{osk})$
check $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps_{pp}}(C, R, P_1), (\sigma, \tau), \mathsf{opk})$		
return $\mathsf{cred} = (C, (\sigma, \tau), r_3, O)$		

Proposed ABC construction

Proposed ABC construction

ABC.Show(pp, usk, $(\mathsf{opk}_i)_{i\in[n]}$, opk, $\mathcal{S}$, $\mathcal{D}$, cred)		ABC.Verify(pp, $(\mathsf{opk}_i)_{i\in[n]}$, $\mathcal{S}$, $\mathcal{D}$)
parse cred $= (C, (\sigma, \tau), r, O)$; $\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$ **if** $O = (1, (o_1, o_2))$ **then** $O' = (1, (\mu \cdot o_1, o_2))$ **else** $O' = \mu \cdot O$		
$\sigma' \xleftarrow{\$}$ SPS-EQ.ChgRep(sps$_{\mathsf{pp}}$, $(C, rC, P_1), \sigma, \tau, \mu, \rho$, opk) $(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$ cred$' \leftarrow (C_1, C_2, C_3, \sigma')$; opk$' \leftarrow$ ConvertPK(opk, ρ) $\Pi \leftarrow$ SH.PPrv($(\mathsf{opk}_i)_{i\in[n]}$, opk$'$, ρ) wit $\leftarrow$ SCDS.OpenSS(scds$_{\mathsf{pp}}$, μC, $\mathcal{S}$, O') $\underline{\mathsf{wit}} \leftarrow$ SCDS.OpenDS(scds$_{\mathsf{pp}}$, μC, $\mathcal{D}$, O')		
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$ $c_1 \leftarrow$ Commit(ck, a_1, r_2); $c_2 \leftarrow$ Commit(ck, a_2, r_4)		
$\Sigma_1 = (\mathsf{cred}', \Pi, \mathsf{opk}', \mathsf{wit}, \underline{\mathsf{wit}}, c_1, c_2)$	$\xrightarrow{\Sigma_1}$	**parse** $\Sigma_1 = (\mathsf{cred}', \Pi, \mathsf{opk}', \mathsf{wit}, \underline{\mathsf{wit}}, c_1, c_2)$
$\pi_1 \leftarrow$ SCDS.PoE(ek, $\mathcal{S}$, $\tilde{e}$)	$\xleftarrow{e, \tilde{e}}$	$e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
$\pi_2 \leftarrow$ SCDS.PoE(ek, $\mathcal{D}$, $\tilde{e}$) $z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$		**parse** cred$' = (C_1, C_2, C_3, \sigma)$
$\Sigma_2 = (z_i, a_i, r_i, \pi_i)_{i\in\{1,2\}}$	$\xrightarrow{\Sigma_2}$	**parse** $\Sigma_2 = (z_i, a_i, r_i, \pi_i)_{i\in\{1,2\}}$
		check $z_1 C_1 = a_1 + eC_2$; $z_2 P_1 = a_2 + eC_3$ $c_1 =$ Commit(ck, a_1, r_2); $c_2 =$ Commit(ck, a_2, r_4) SH.PVer(crs, $(\mathsf{opk}_i)_{i\in[n]}$, opk$'$, Π_1) SPS-EQ.Verify(sps$_{\mathsf{pp}}$, cred$'$, opk) SCDS.VerifySS(scds$_{\mathsf{pp}}$, C_1, $\mathcal{S}$, wit; π_1, $\tilde{e}$) SCDS.VerifyDS(scds$_{\mathsf{pp}}$, C_1, $\mathcal{S}$, $\underline{\mathsf{wit}}$; π_2, $\tilde{e}$)

Agenda

Security properties

Theorem (Unforgeability)

If the q-co-DL assumption holds, the ZKPoK's have perfect ZK, SCDS *is sound, and* SPS-EQ *is* EUF-CMA *secure, then the ABC is unforgeable.*

Security properties

Theorem (Anonymity)

If the DDH assumption holds, the ZKPoK's have perfect ZK, and the SPS-EQ *perfectly adapts signatures, then the ABC is anonymous.*

Security properties

Theorem (Signer-hiding)

If the underlying signature scheme is a SPS-EQ which perfectly adapts signatures (under malicious keys in the honest parameter model), then the ABC supports signer-hiding.

Agenda

Conclusions and Future Work

- Our results explore multiple paths to extend the ABC framework from [**?**]

- Our results explore multiple paths to extend the ABC framework from [**?**]
- We obtained a more flexible framework leveraging different trade-offs

- Our results explore multiple paths to extend the ABC framework from [**?**]
- We obtained a more flexible framework leveraging different trade-offs
- The proposed signer-hiding notion enables more use cases

Conclusions and Future Work

- Our results explore multiple paths to extend the ABC framework from [**?**]
- We obtained a more flexible framework leveraging different trade-offs
- The proposed signer-hiding notion enables more use cases
- Exploring the use of aggregatable signatures with SPS-EQ in the multi-authority setting could enable even more use cases

- Our results explore multiple paths to extend the ABC framework from [**?**]
- We obtained a more flexible framework leveraging different trade-offs
- The proposed signer-hiding notion enables more use cases
- Exploring the use of aggregatable signatures with SPS-EQ in the multi-authority setting could enable even more use cases
- Devising other ways to define equivalence classes could lead to new and more efficient constructions

Thank you for your time!

Scheme	[?]	[?]	[?]	[?] & [?]	[?]	[?]	[?]	Ours
	Issuing n-attr. credential							
Comm.	$O(n)$	$O(n)$	$O(n)$	$\mathbf{O(1)}$	$O(n)$	$\mathbf{O(1)}$	$O(n)$	$\mathbf{O(1)}$
User	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Issuer	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	Showing k-of-n attributes (selective disclosure)							
$\lvert\mathsf{ek}\rvert$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$
Comm.	$O(n)$	$O(1)$	$O(k)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
User	$O(n)$	$O(n)$	$O(k)$	$O(n-k)$	$O(n-k)$	$O(n-k)$	$\mathbf{O(1)}$	$O(\max\{n-k,k\})$
Verifier	$O(n)$	$O(n)$	$O(k)$	$O(k)$	$O(k)$	$O(k)$	$O(n)$	$\mathbf{O(1)}$

Table: Asymptotic complexities of ABC systems where n is the number of attributes in the credential and k the number of disclosed ones during a showing.

ABC	[?]	[?]	[?]	[?]	Ours
			Parameters size (n-attributes)		
ek	$(\frac{n^2+n+2}{2})_{\mathbb{G}_1} + n_{\mathbb{G}_2}$	$(2n+2)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1} + (n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1} + (n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1} + (n+1)_{\mathbb{G}_2}$
Cred	$2_{\mathbb{G}_2}$	$4_{\mathbb{G}_1}$	$1_{\mathbb{G}_1} + 6_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1} + 1_{\mathbb{G}_2} + 2_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1} + 6_{\mathbb{G}_2} + 3_{\mathbb{Z}_p}$
			Bandwidth		
Issue	$4_{\mathbb{G}_2} + 2_{\mathbb{Z}_p}$	$n_{\mathbb{G}_1}$	$3_{\mathbb{G}_1} + (n+3)_{\mathbb{Z}_p}$	$12_{\mathbb{G}_1} + 1_{\mathbb{G}_2} + 8_{\mathbb{Z}_p}$	$14_{\mathbb{G}_1} + 11_{\mathbb{G}_2} + 7_{\mathbb{Z}_p}$
Show	$2_{\mathbb{G}_1} + 2_{\mathbb{G}_2} + 1_{\mathbb{G}_T} + 2_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1} + 1_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1} + 5_{\mathbb{Z}_p}$	$10_{\mathbb{G}_1} + 1_{\mathbb{G}_2} + 8_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1} + 14_{\mathbb{G}_2} + 4_{\mathbb{Z}_p}$
			k-of-n attributes (AND)		
Usr	$(2(n\text{-}k)+2)_{\mathbb{G}_1}, 2_{\mathbb{G}_2}$, **1**	$6_{\mathbb{G}_1}$	$(6+n\text{-}k)_{\mathbb{G}_1}$	$(11+n\text{-}k)_{\mathbb{G}_1}, 1_{\mathbb{G}_2}$, **8**	$(20+n\text{-}k)_{\mathbb{G}_1}$, $(k\text{-}1)_{\mathbb{G}_2}$, **19**
Ver	$(k+1)_{\mathbb{G}_1}, 1_{\mathbb{G}_T}$, **5**	$4_{\mathbb{G}_1}, 2n_{\mathbb{G}_2}$, **3**	$5_{\mathbb{G}_1}, (k+1)_{\mathbb{G}_2}$, **3**	$4_{\mathbb{G}_1}, (k+1)_{\mathbb{G}_2}$, **10**	$10_{\mathbb{G}_1}$, **16**
			k-of-n attributes (NAND)		
Usr	N/A	N/A	$(6+n)_{\mathbb{G}_1}$	N/A	$(31+n)_{\mathbb{G}_1}$, $(9+2k)_{\mathbb{G}_2}$, **19**
Ver	N/A	N/A	$(2k+5)_{\mathbb{G}_1}$, $(k+3)_{\mathbb{G}_2}$, **3**	N/A	$10_{\mathbb{G}_1}$, **17**

Table: Efficiency of ABCs considering issuing and showing interactions (the number of pairings is marked in bold).

$\underline{\mathsf{SPS\text{-}EQ.ParGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
return $(\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$

$\underline{\mathsf{SPS\text{-}EQ.KGen}(\mathsf{pp}, 1^\lambda)}$:
$\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$; $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times 2}$
$[\mathbf{B}]_2 \leftarrow [\mathbf{K}_0]_2[\mathbf{A}]_2$; $[\mathbf{C}]_2 \leftarrow [\mathbf{K}]_2[\mathbf{A}]_2$
$\mathsf{sk} \leftarrow (\mathbf{K}_0, \mathbf{K})$; $\mathsf{pk} \leftarrow ([\mathbf{B}]_2, [\mathbf{C}]_2)$
return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{SPS\text{-}EQ.Sign}(\mathsf{pp}, \mathsf{sk}, [\mathbf{m}]_1)}$:
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1$; $[\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
$\Omega \leftarrow \mathsf{PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
parse $\Omega = (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
$\mathbf{u}_1 \leftarrow \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top [\mathbf{m}]_1$; $\mathbf{u}_2 \leftarrow \mathbf{K}_0^\top [\mathbf{w}]_1$
$\sigma \leftarrow ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
$\tau \leftarrow ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$
return (σ, τ)

$\underline{\mathsf{SPS\text{-}EQ.TParGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$
return $(\mathsf{pp}, \mathsf{td})$

$\underline{\mathsf{SPS\text{-}EQ.Verify}(\mathsf{pp}, [\mathbf{m}]_1, (\sigma, \tau), \mathsf{pk})}$:
parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \perp\}$
check $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) =$
$\quad e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$
if $\tau \neq \perp$ **check**
$\quad \mathsf{PRVer}(\mathsf{crs}, [\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
$\quad e([\mathbf{u}_2]_1^\top, [\mathbf{A}_2]) = e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$

$\underline{\mathsf{SPS\text{-}EQ.ConvertPK}(\mathsf{pk}, \rho)}$:
parse $\mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2)$
return $(\rho[\mathbf{B}]_2, \rho[\mathbf{C}]_2)$

$\underline{\mathsf{SPS\text{-}EQ.ConvertSK}(\mathsf{sk}, \rho)}$:
parse $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K})$; **return** $(\rho\mathbf{K}_0, \rho\mathbf{K})$

$\underline{\text{SPS-EQ.ChgRep}(\text{pp}, [\mathbf{m}]_1, \sigma, \tau, \mu, \text{pk})}$:

parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$

parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \perp\}$

$\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$

check $\text{PVer}(\text{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$

check $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$

check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq$
$e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$

$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$

$[\mathbf{u}_1']_1 \leftarrow \mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1$

$[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$

for all $i \in \{0, 1\}$

$\quad [z_i']_2 \leftarrow \alpha[z_i]_2$

$\quad [\mathbf{a}_i']_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$

$\quad [d_i']_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$

$\Omega' \leftarrow (([\mathbf{a}_i']_1, [d_i']_2, [z_i']_2)_{i\in\{0,1\}}, \alpha Z_1)$

$\sigma' \leftarrow ([\mathbf{u}_1']_1, [\mathbf{t}']_1, \Omega')$

return $(\mu[\mathbf{m}]_1, \sigma')$

$\underline{\text{SPS-EQ.ConvertSig}(\text{crs}, [\mathbf{m}]_1, \sigma, \tau, \mu, \boldsymbol{\rho}, \text{pk})}$:

parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$

parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \perp\}$

$\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$

check $\text{PVer}(\text{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$

check $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$

check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq$
$e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$

$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$

$[\mathbf{u}_1']_1 := \boldsymbol{\rho}(\mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1)$

$[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$

for all $i \in \{0, 1\}$

$\quad [z_i']_2 \leftarrow \alpha[z_i]_2$

$\quad [\mathbf{a}_i']_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$

$\quad [d_i']_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$

$\Omega' \leftarrow (([\mathbf{a}_i']_1, [d_i']_2, [z_i']_2)_{i\in\{0,1\}}, \alpha Z_1)$

$\sigma' \leftarrow ([\mathbf{u}_1']_1, [\mathbf{t}']_1, \Omega')$

return $(\mu[\mathbf{m}]_1, \sigma')$

References I