# Issuer-Hiding for BBS Anonymous Credentials via Randomizable Keys

Andrea Flamini Karla Friedrichs Anja Lehmann

eprint.iacr.org/2026/369

# Issuer-Hiding for BBS Anonymous Credentials via Randomizable Keys

Andrea Flamini Karla Friedrichs Anja Lehmann

eprint.iacr.org/2026/369

eprint.iacr.org/2026/870

with also Jonathan Katz, Watson Ladd, Marek Sefranek

# Anonymous Credentials (ACs)

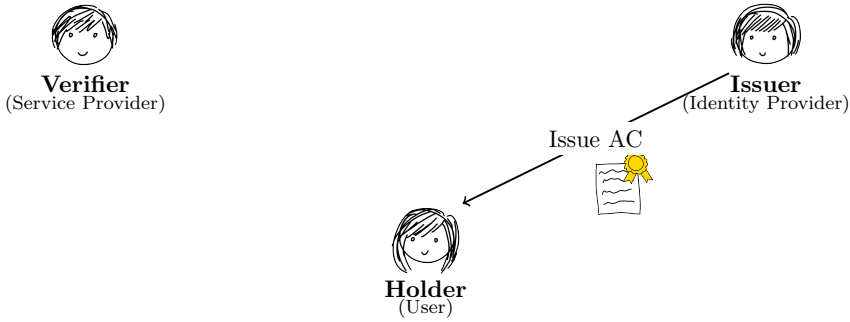
# Anonymous Credentials (ACs)

**Verifier**
(Service Provider)

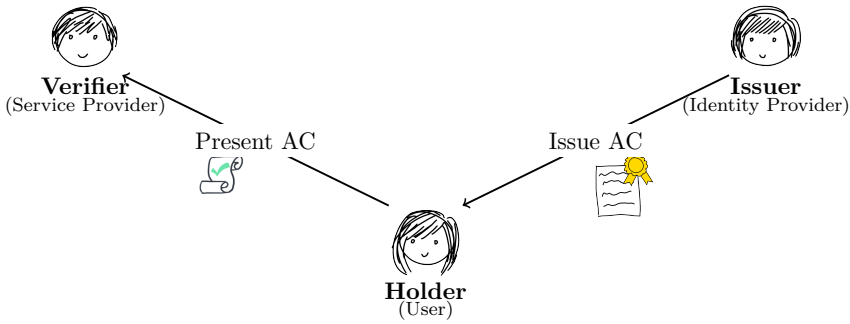
**Issuer**
(Identity Provider)

**Holder**
(User)

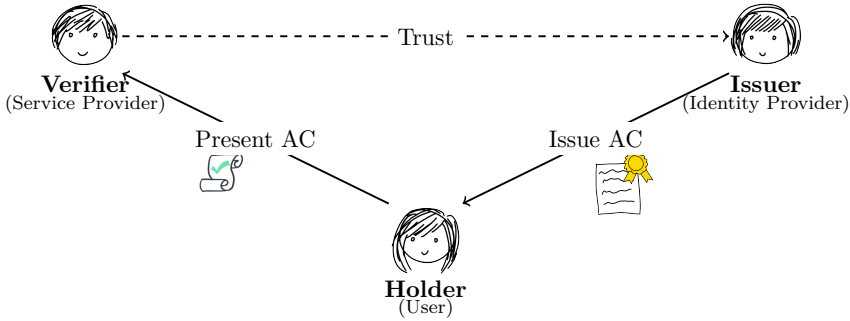
# Anonymous Credentials (ACs)



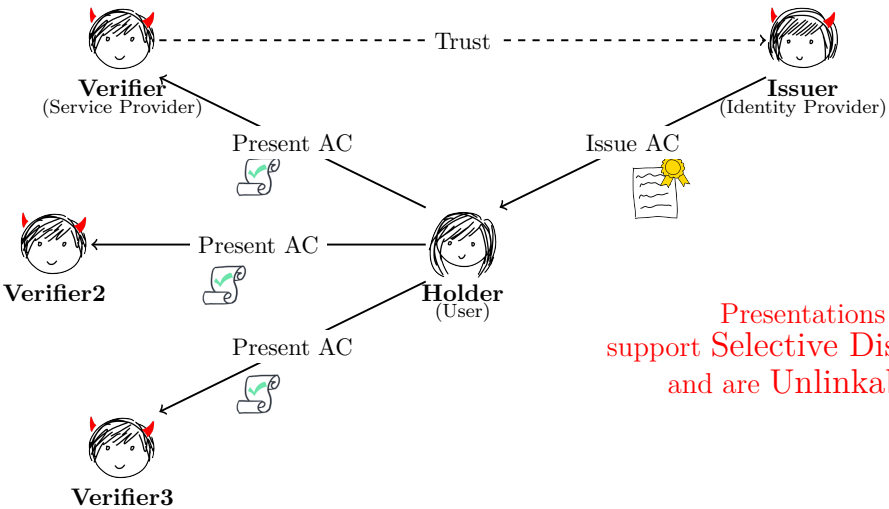
# Anonymous Credentials (ACs)



# Anonymous Credentials (ACs)



# Anonymous Credentials (ACs)




Presentations
support Selective Disclosure
and are Unlinkable!

# What do presentations actually leak?

AC presentations reveal (at least):

# What do presentations actually leak?

AC presentations reveal (at least):

*the disclosed attributes*
*(possibly none)*

# What do presentations actually leak?

AC presentations reveal (at least):

*the disclosed attributes*
*(possibly none)*

*the issuer*
*(mandatory)*

# What do presentations actually leak?

AC presentations reveal (at least):

*the disclosed attributes*
*(possibly none)*

*the issuer*
*(mandatory)*

Revealing the issuer is undesirable in some circumstances, e.g.:

Age verification credentials

# Issuer Hiding Anonymous Credentials

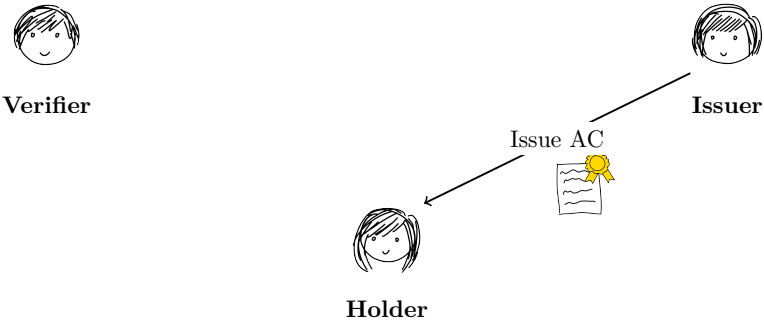
# Issuer Hiding Anonymous Credentials

**Verifier**

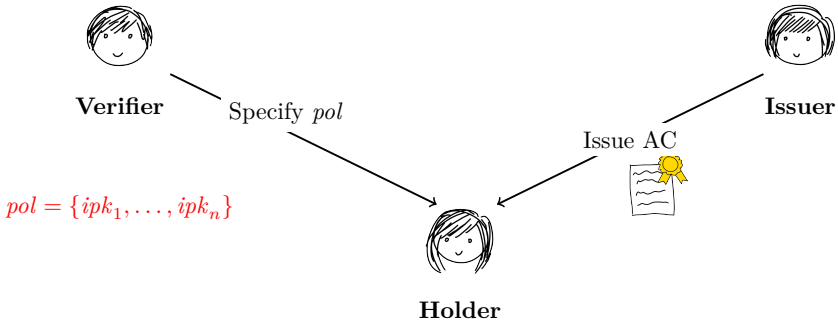
**Issuer**

**Holder**

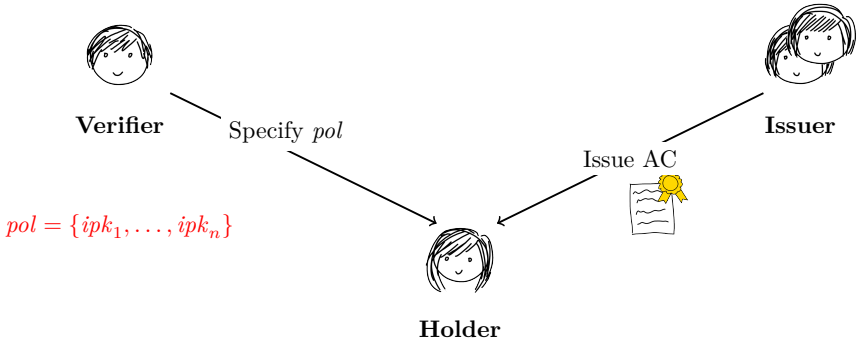
# Issuer Hiding Anonymous Credentials



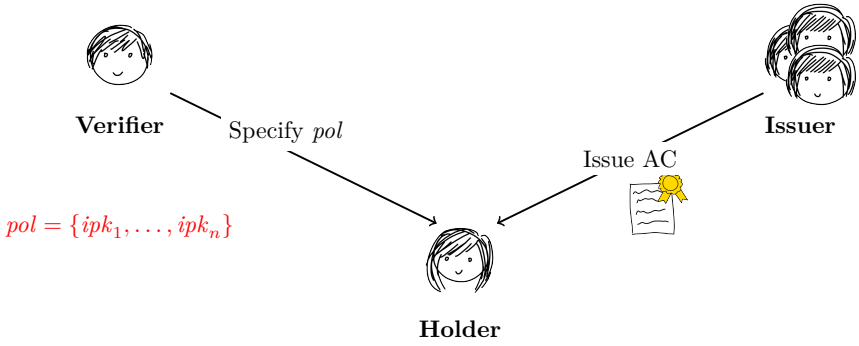
# Issuer Hiding Anonymous Credentials



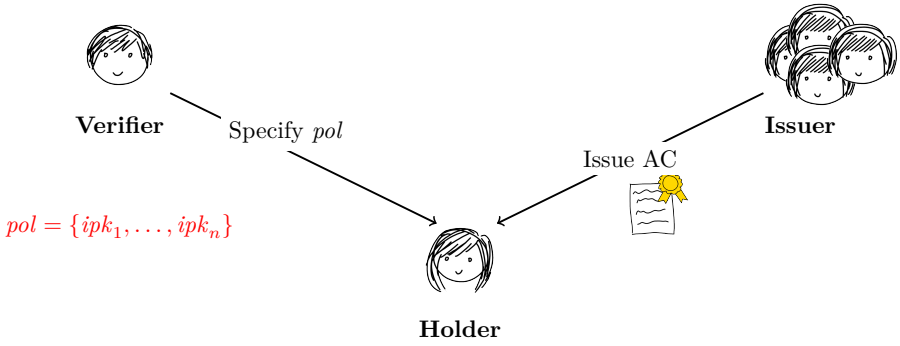
# Issuer Hiding Anonymous Credentials



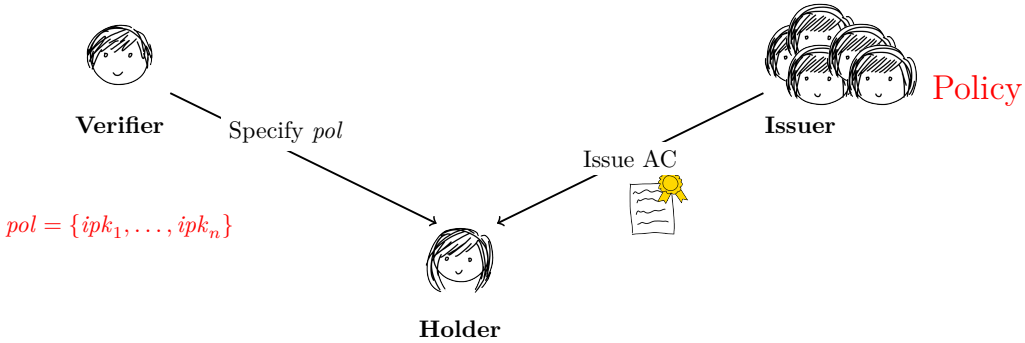
# Issuer Hiding Anonymous Credentials



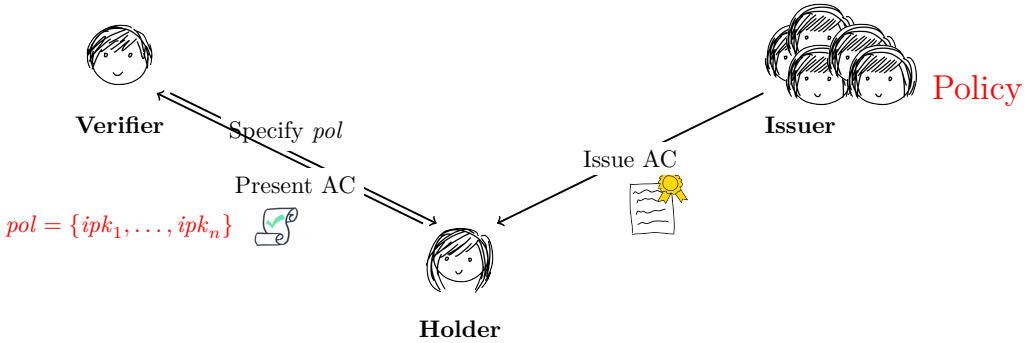
# Issuer Hiding Anonymous Credentials



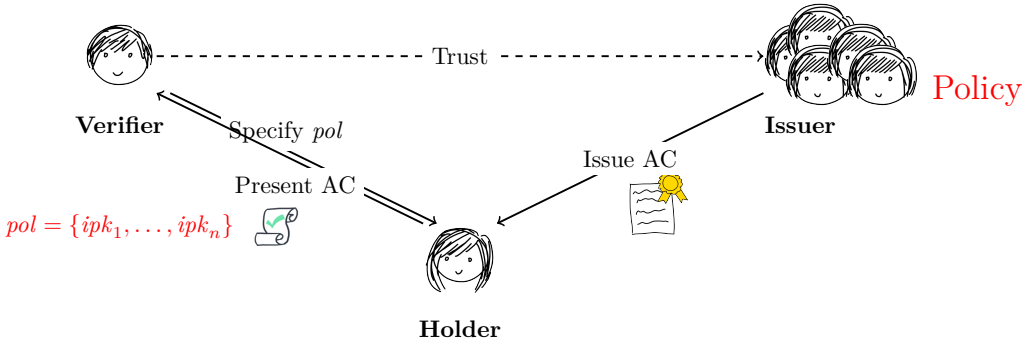
# Issuer Hiding Anonymous Credentials



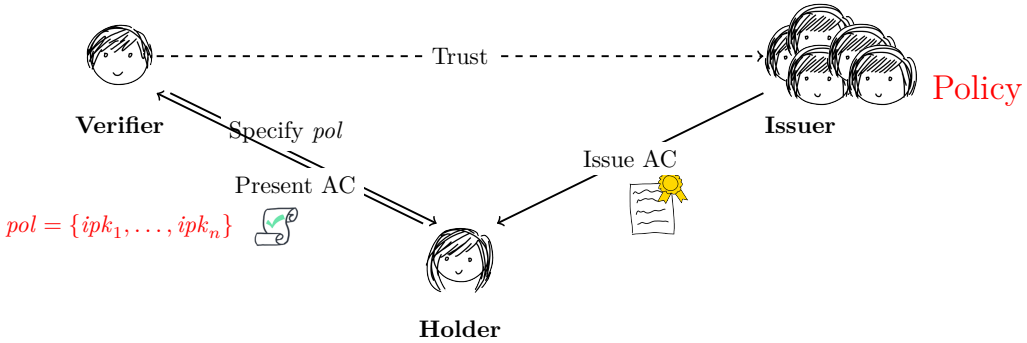
# Issuer Hiding Anonymous Credentials



# Issuer Hiding Anonymous Credentials

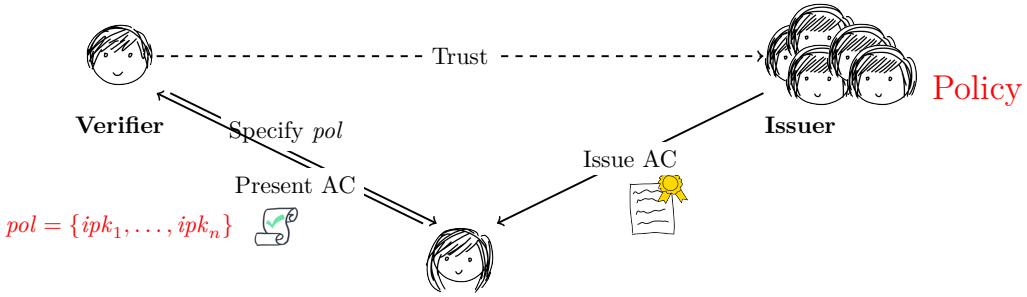


# Issuer Hiding Anonymous Credentials




Natural solution: using OR proofs [CDS94]

# Issuer Hiding Anonymous Credentials

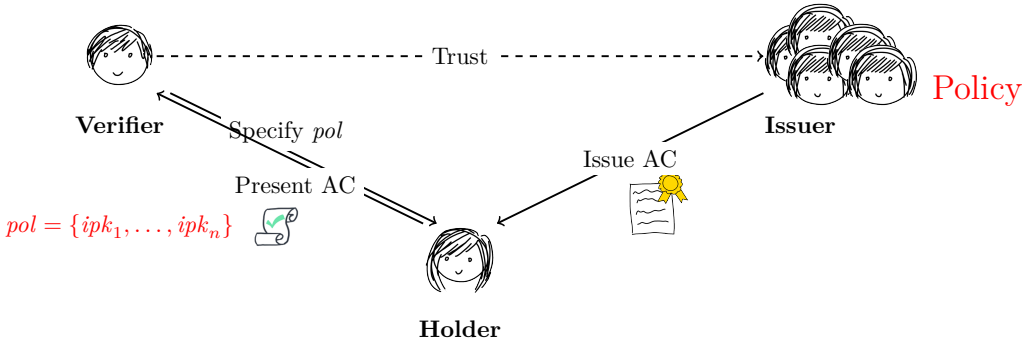



Not ideal! Presentation size linear in $|pol|$.

Natural solution: using OR proofs [CDS94]

$\mathsf{Vf}(cred, ipk_1) \vee \mathsf{Vf}(cred, ipk_2) \vee \cdots \vee \mathsf{Vf}(cred, ipk_n)$

# Issuer Hiding Anonymous Credentials




Not ideal! Presentation size linear in $|pol|$.

Natural solution: using OR proofs [CDS94]

$\mathsf{Vf}(cred, ipk_1) \vee \mathsf{Vf}(cred, ipk_2) \vee \cdots \vee \mathsf{Vf}(cred, ipk_n)$

All solutions with constant-size presentations adopt *policy public keys* $pk_{pol}$

# Existing Constructions

# Existing Constructions

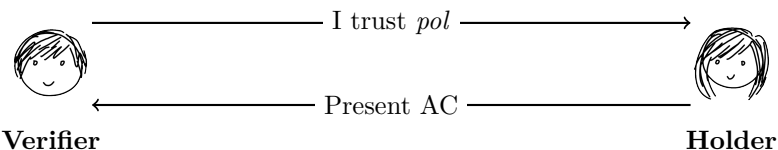
Without Policy key

(Type 0)

[CLPK22]

Naïve approach

$pol = (ipk_1, ipk_2, \ldots, ipk_n)$



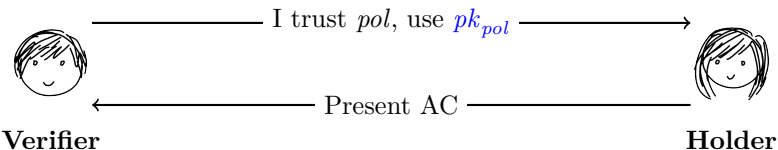
# Existing Constructions

Without Policy key
(Type 0)
[CLPK22]
Naïve approach

Universal Policy Keys
(Type 1)
[BEK+21, CDLPK22]
*constant size!*

$pk_{pol} \xleftarrow{\$} \mathsf{SetPolicy}(pol)$

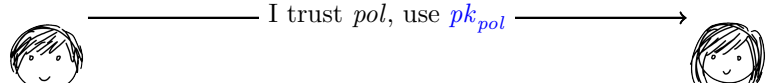
$pol = (ipk_1, ipk_2, \ldots, ipk_n)$



# Existing Constructions

| Without Policy key | Universal Policy Keys | Per-Verifier Policy Keys |
|---|---|---|
| (Type 0) | (Type 1) | (Type 2) |
| [CLPK22] | [BEK+21, CDLPK22] | [MBG+23, ST24, KS25] |
| Naïve approach | *constant size!* | *constant size!* |

$pol = (ipk_1, ipk_2, \ldots, ipk_n)$




$sk_{pol}, pk_{pol} \xleftarrow{\$} \mathsf{SetPolicy}(pol)$

# Existing Constructions

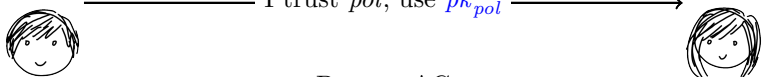
Without Policy key
(Type 0)
[CLPK22]
Naïve approach

Universal Policy Keys
(Type 1)
[BEK+21, CDLPK22]
*constant size!*

Per-Verifier Policy Keys
(Type 2)
[MBG+23, ST24, KS25]
*constant size!*

No outsourcing $pk_{pol}$ generation
Privatly verifiable
Verifier must handle $sk_{pol}$

$pol = (ipk_1, ipk_2, \ldots, ipk_n)$




**Verifier** **Holder**

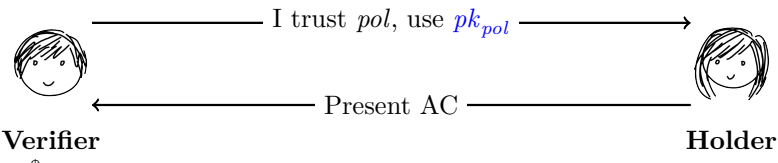
$sk_{pol}, pk_{pol} \xleftarrow{\$} \mathsf{SetPolicy}(pol)$

# Existing Constructions

| Without Policy key | Universal Policy Keys | Per-Verifier Policy Keys |
|---|---|---|
| (Type 0) | (Type 1) | (Type 2) |
| [CLPK22] | [BEK+21, CDLPK22] | [MBG+23, ST24, KS25] |
| Naïve approach | *constant size!* | *constant size!* |

No outsourcing $pk_{pol}$ generation
Privatly verifiable
Verifier must handle $sk_{pol}$

$pol = (ipk_1, ipk_2, \ldots, ipk_n)$



# New Anonymity Set

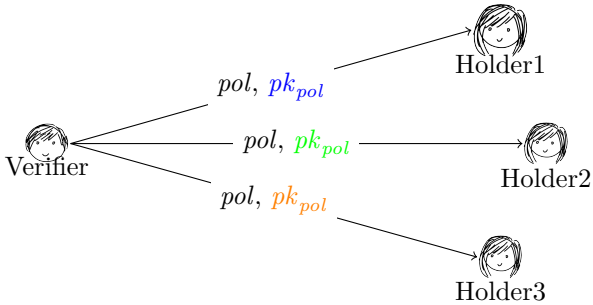
The anonymity set depends on $pol$ and $pk_{pol}$.

They can be used to track users if they reuse them!

# New Anonymity Set

The anonymity set depends on $pol$ and $pk_{pol}$.

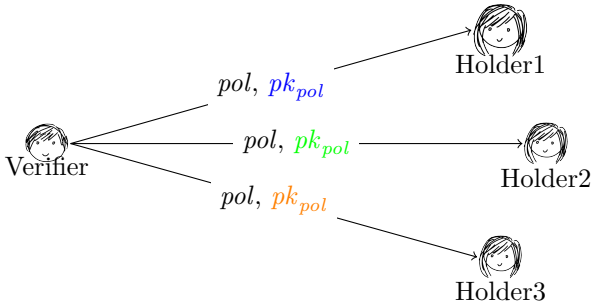
They can be used to track users if they reuse them!



# New Anonymity Set

The anonymity set depends on $pol$ and $pk_{pol}$.

They can be used to track users if they reuse them!




$pol, pk_{pol}$ are context information, and verifiers must send them to holders right before every presentation.

# Policy Registry

$pol$ and $pk_{pol}$ are linear in the size of the policy.

Let's use a Registry.

# Policy Registry

$pol$ and $pk_{pol}$ are linear in the size of the policy.
Let's use a Registry.

Type 0

$$\begin{aligned}
\mathsf{id}_1 &: \ pol_1 = (ipk_1, ipk_2, ipk_3) \\
\mathsf{id}_2 &: \ pol_2 = (ipk_1, ipk_2, ipk_4, ipk_5) \\
\mathsf{id}_3 &: \ pol_3 = (ipk_2, ipk_3, ipk_5, ipk_6, ipk_7) \\
&\vdots
\end{aligned}$$

Type 1

$$\begin{aligned}
\mathsf{id}_1 &: \ (pol_1, pk_{pol_1}) \\
\mathsf{id}_2 &: \ (pol_2, pk_{pol_2}) \\
\mathsf{id}_3 &: \ (pol_3, pk_{pol_3}) \\
&\vdots
\end{aligned}$$

# Policy Registry

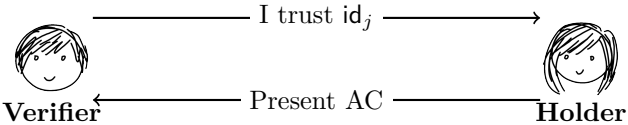
$pol$ and $pk_{pol}$ are linear in the size of the policy.
Let's use a Registry.

Type 0

$\mathsf{id}_1 : \ pol_1 = (ipk_1, ipk_2, ipk_3)$
$\mathsf{id}_2 : \ pol_2 = (ipk_1, ipk_2, ipk_4, ipk_5)$
$\mathsf{id}_3 : \ pol_3 = (ipk_2, ipk_3, ipk_5, ipk_6, ipk_7)$
$\vdots$

Type 1

$\mathsf{id}_1 : \ (pol_1, pk_{pol_1})$
$\mathsf{id}_2 : \ (pol_2, pk_{pol_2})$
$\mathsf{id}_3 : \ (pol_3, pk_{pol_3})$
$\vdots$



# Our *Efficient* Type 0 and Type 1 Solutions for BBS credentials

# Randomizing BBS Keys

It is possible to randomize a BBS public key $ipk$, and adapt a signature $\sigma$ over $\mathbf{a}$.

# Randomizing BBS Keys

It is possible to randomize a BBS public key $ipk$, and adapt a signature $\sigma$ over $\mathbf{a}$.

$$\mathsf{Randomize}(ipk, \alpha) \longrightarrow ipk^\star$$

$$\mathsf{Adapt}(\sigma, \alpha) \longrightarrow \sigma^\star$$

$$\mathsf{verify}(ipk^\star, \sigma^\star, \mathbf{a}) = 1$$

# Randomizing BBS Keys

It is possible to randomize a BBS public key $ipk$, and adapt a signature $\sigma$ over $\mathbf{a}$.

$$\mathsf{Randomize}(ipk, \alpha) \longrightarrow ipk^\star$$

$$\mathsf{Adapt}(\sigma, \alpha) \longrightarrow \sigma^\star$$

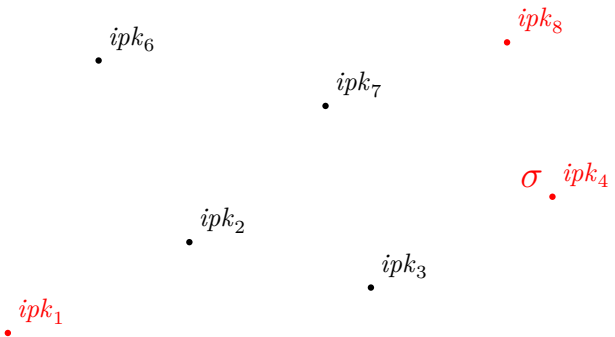
$$\mathsf{verify}(ipk^\star, \sigma^\star, \mathbf{a}) = 1$$

More specifically:

$$\mathsf{Randomize}(ipk, \alpha) \longrightarrow ipk^\alpha$$

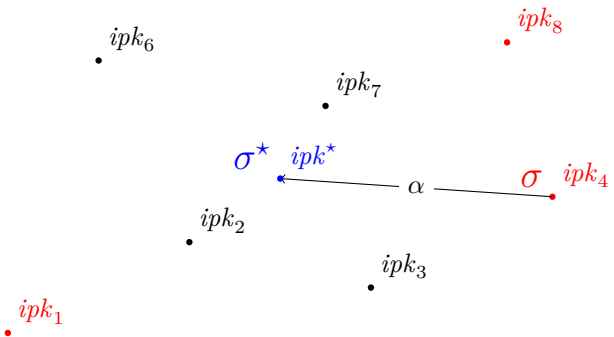
$$\mathsf{Adapt}((A, e), \alpha) \longrightarrow (A^{\frac{1}{\alpha}}, \alpha e)$$

# Type 0 Construction
(without policy public key)




- $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{Adapt}(ipk_4, \sigma, \alpha) \to (ipk^\star, \sigma^\star)$;
- Prove knowledge of a credential issued by $ipk^\star$
- Prove knowledge of the dlog of one of the $ipk_i \in pol$ w.r.t. $ipk^\star$.

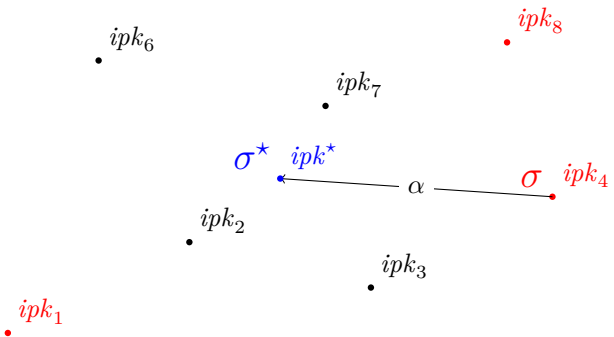
# Type 0 Construction
(without policy public key)




- $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{Adapt}(ipk_4, \sigma, \alpha) \to (ipk^\star, \sigma^\star)$;
- Prove knowledge of a credential issued by $ipk^\star$
- Prove knowledge of the dlog of one of the $ipk_i \in pol$ w.r.t. $ipk^\star$.

# Type 0 Construction

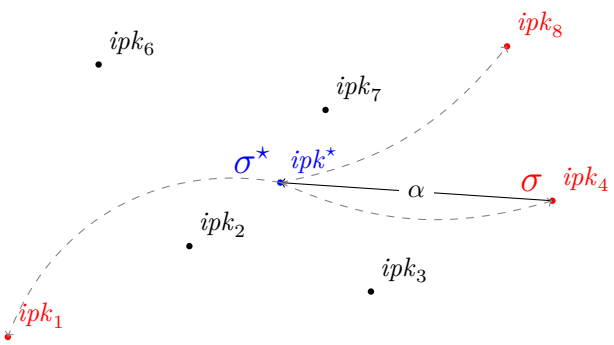
(without policy public key)




- $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{Adapt}(ipk_4, \sigma, \alpha) \to (ipk^\star, \sigma^\star)$;
- Prove knowledge of a credential issued by $ipk^\star$
- Prove knowledge of the dlog of one of the $ipk_i \in pol$ w.r.t. $ipk^\star$.

# Type 0 Construction

(without policy public key)




- $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{Adapt}(ipk_4, \sigma, \alpha) \to (ipk^\star, \sigma^\star)$;
- Prove knowledge of a credential issued by $ipk^\star$
- Prove knowledge of the dlog of one of the $ipk_i \in pol$ w.r.t. $ipk^\star$.

# Type 0 Construction Baseline

OR Proof over BBS presentations

$$\mathsf{Vf}(cred, ipk_1) \vee \mathsf{Vf}(cred, ipk_2) \vee \cdots \vee \mathsf{Vf}(cred, ipk_n)$$

# Type 0 Construction Baseline

OR Proof over BBS presentations

$$\mathsf{Vf}(cred, ipk_1) \vee \mathsf{Vf}(cred, ipk_2) \vee \cdots \vee \mathsf{Vf}(cred, ipk_n)$$

$$\underbrace{\overline{A}_1, \overline{B}_1, ch_1, \{z_{1,i}\}_{i \in \mathcal{U}}, e_1, r_1}_{\text{BBS pres. for } ipk_1}, \underbrace{\overline{A}_2, \overline{B}_2, ch_2, \{z_{2,i}\}_{i \in \mathcal{U}}, e_2, r_2}_{\text{BBS pres. for } ipk_2}, \ldots, \underbrace{\overline{A}_n, \overline{B}_n, ch_n, \{z_{n,i}\}_{i \in \mathcal{U}}, e_n, r_n}_{\text{BBS pres. for } ipk_n}$$

$$n \cdot \mathsf{size}(\mathbb{G}_1^2 \times \mathbb{Z}_p^{|\mathcal{U}|+3})$$

$$|pol| = n$$

# Type 0 Construction Baseline

OR Proof over BBS presentations

$$\mathsf{Vf}(cred, ipk_1) \vee \mathsf{Vf}(cred, ipk_2) \vee \cdots \vee \mathsf{Vf}(cred, ipk_n)$$

$$\underbrace{\overline{A}_1, \overline{B}_1, ch_1, \{z_{1,i}\}_{i\in\mathcal{U}}, e_1, r_1}_{\text{BBS pres. for } ipk_1}, \underbrace{\overline{A}_2, \overline{B}_2, ch_2, \{z_{2,i}\}_{i\in\mathcal{U}}, e_2, r_2}_{\text{BBS pres. for } ipk_2}, \ldots, \underbrace{\overline{A}_n, \overline{B}_n, ch_n, \{z_{n,i}\}_{i\in\mathcal{U}}, e_n, r_n}_{\text{BBS pres. for } ipk_n}$$

$$n \cdot \mathsf{size}(\mathbb{G}_1^2 \times \mathbb{Z}_p^{|\mathcal{U}|+3})$$

$$|pol| = n$$

We can reduce it to

$$\underbrace{ipk^\star}_{\text{Rand. } ipk}, \underbrace{\overline{A}, \overline{B}, \{z_i\}_{i\in\mathcal{U}}, e, r}_{\text{BBS pres.}}, \underbrace{ch_1, z_1}_{\mathsf{DLog}(ipk^\star, ipk_1)}, \ldots, \underbrace{ch_n, z_n}_{\mathsf{DLog}(ipk^\star, ipk_n)}$$

$$\mathsf{size}(\mathbb{G}_1^2 \times \mathbb{Z}_p^{|\mathcal{U}|+3}) + n \cdot \mathsf{size}(\mathbb{Z}_p^2)$$

# Performances

Running example: credential with 33 attributes, prover proves to be European ($|pol| = 27$)

Compare $\mathsf{IHBBS}_0$ with [CDS94, TZ23]($\mathsf{IHBBSOR}$).

| IHAC Scheme | Comput. OH | Commun. OH |
|---|---|---|
| Naïve | $(\|pol\| - 1)\mathsf{Sim}_{\mathsf{BBS}}$ | $(\|pol\| - 1)\pi_{\mathsf{BBS}}$ |
| running example | 46 ms | 27 KB |
| our $\mathsf{IHBBS}_0$ | $\mathsf{e}_{\mathbb{G}_1} + 2\mathsf{e}_{\mathbb{G}_2} + (\|pol\| - 1)\mathsf{Sim}_{\mathsf{DL}}$ | $(\|pol\|)\pi_{\mathsf{DL}}$ |
| running example | 27 ms | 1.7 KB |

# Type 1 IHAC for BBS

(With universal policy keys)

# Type 1 IHAC for BBS

(With universal policy keys)

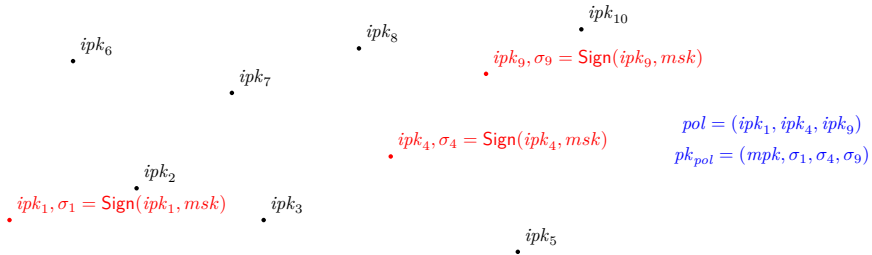
Public key randomization enables the framework from Bobolz et al. [BEK+21].

# Type 1 IHAC for BBS

(With universal policy keys)

Public key randomization enables the framework from Bobolz et al. [BEK+21].

A manager signs with $(mpk, msk)$ the public keys to include in the policy.

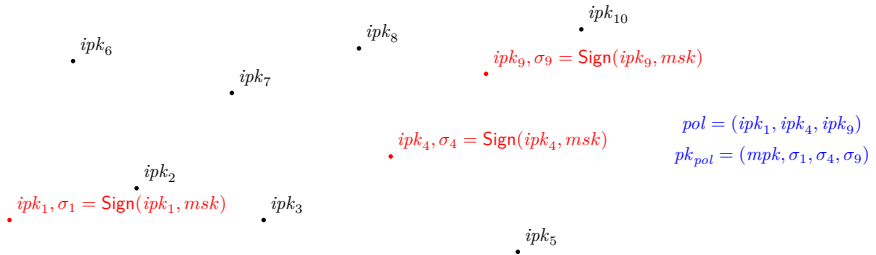


# Type 1 IHAC for BBS

(With universal policy keys)

Public key randomization enables the framework from Bobolz et al. [BEK+21].

A manager signs with $(mpk, msk)$ the public keys to include in the policy.



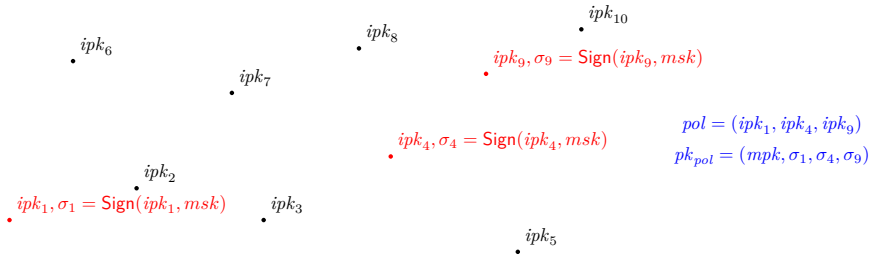

The prover computes $\mathsf{NIZKP}\{cred, ipk, \sigma_i | \mathsf{Vf}(\mathbf{a}, cred, ipk) = 1 \ \wedge \ \mathsf{Vf}(ipk, \sigma_i, mpk) = 1\}$

# Type 1 IHAC for BBS

(With universal policy keys)

Public key randomization enables the framework from Bobolz et al. [BEK+21].

A manager signs with $(mpk, msk)$ the public keys to include in the policy.




The prover computes $\mathsf{NIZKP}\{cred, ipk, \sigma_i | \mathsf{Vf}(\mathbf{a}, cred, ipk) = 1 \ \wedge \ \mathsf{Vf}(ipk, \sigma_i, mpk) = 1\}$

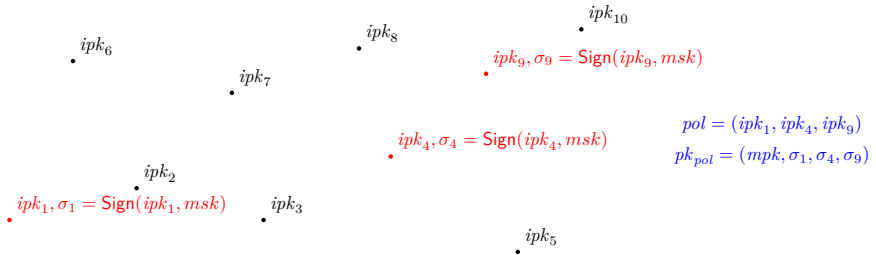
We can do it using *BBS key randomization!*

# Type 1 IHAC for BBS

(With universal policy keys)

Public key randomization enables the framework from Bobolz et al. [BEK+21].

A manager signs with $(mpk, msk)$ the public keys to include in the policy.



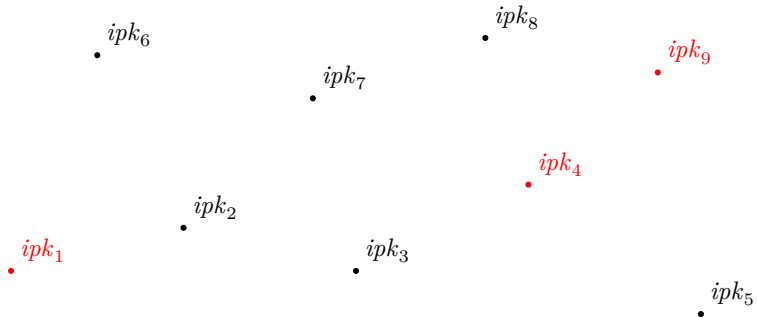

The prover computes $\mathsf{NIZKP}\{cred, ipk, \sigma_i | \mathsf{Vf}(\mathbf{a}, cred, ipk) = 1 \ \wedge \ \mathsf{Vf}(ipk, \sigma_i, mpk) = 1\}$

We can do it using *BBS key randomization!*

Do we really need a digital signature for this?

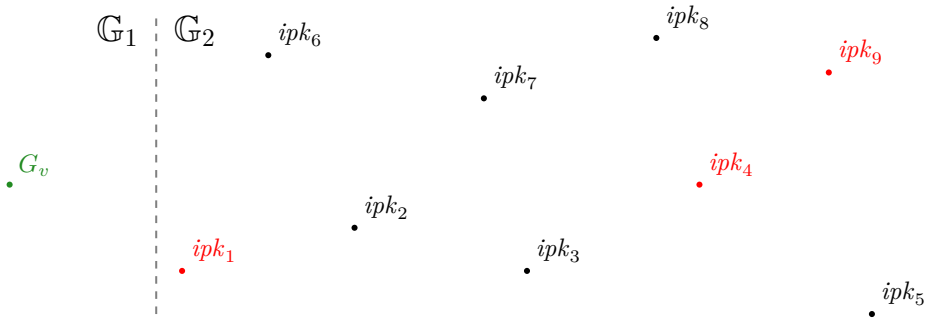
# Type 1 IHAC for BBS

## Policy Generation



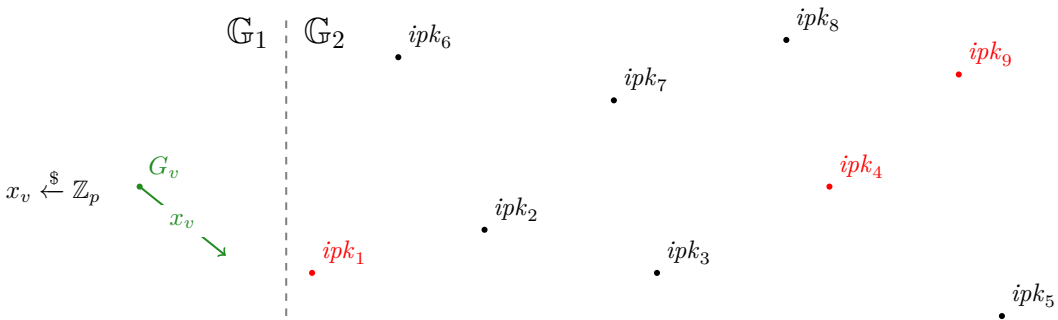
# Type 1 IHAC for BBS

## Policy Generation



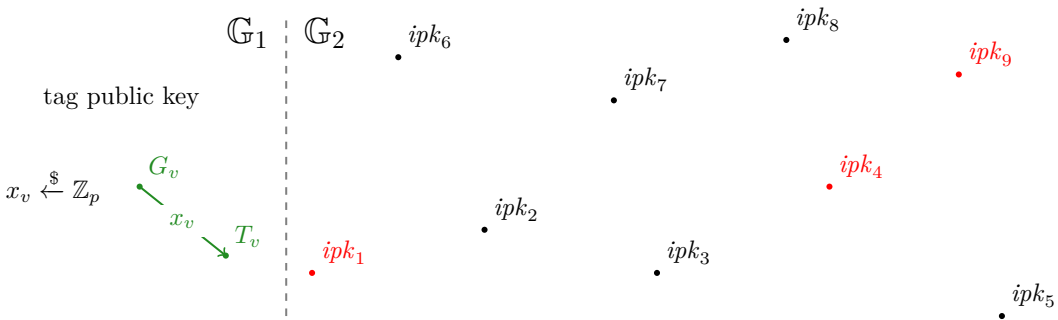
# Type 1 IHAC for BBS

## Policy Generation



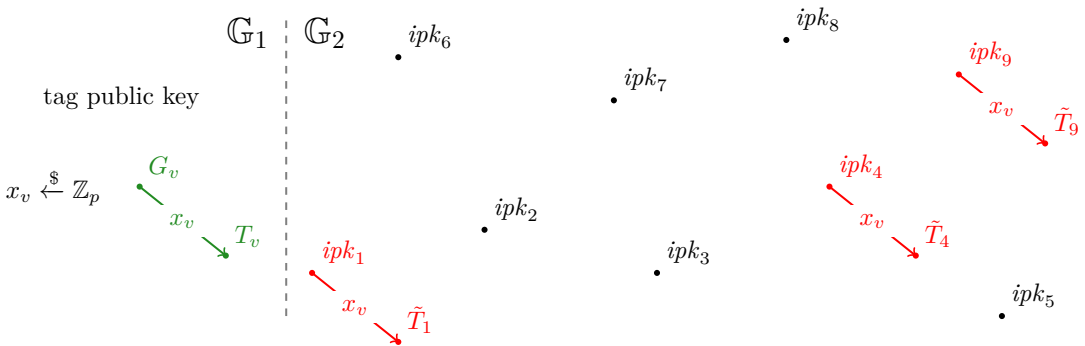
# Type 1 IHAC for BBS

## Policy Generation



# Type 1 IHAC for BBS

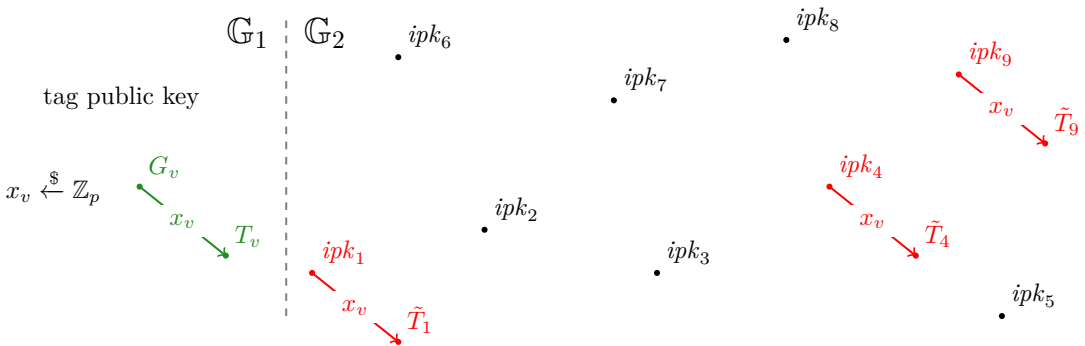
## Policy Generation




The policy key for $pol = \{ipk_1, ipk_4, ipk_9\}$ is $pk_{pol} = ((G_v, T_v), \underbrace{(\tilde{T}_1, \tilde{T}_4, \tilde{T}_9)}_{\text{the analogue of signatures in Bobolz et al.!}})$.

# Type 1 IHAC for BBS

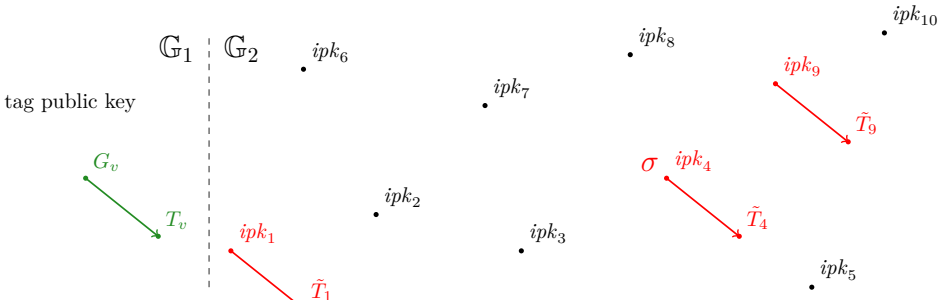
## Policy Generation




The policy key for $pol = \{ipk_1, ipk_4, ipk_9\}$ is $pk_{pol} = ((G_v, T_v), \underbrace{(\tilde{T}_1, \tilde{T}_4, \tilde{T}_9)}_{\text{the analogue of signatures in Bobolz et al.!}})$.

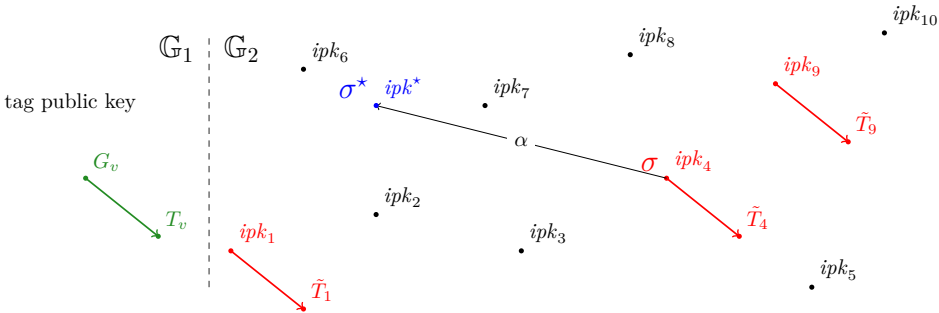
# Type 1 IHAC for BBS

## Presentation



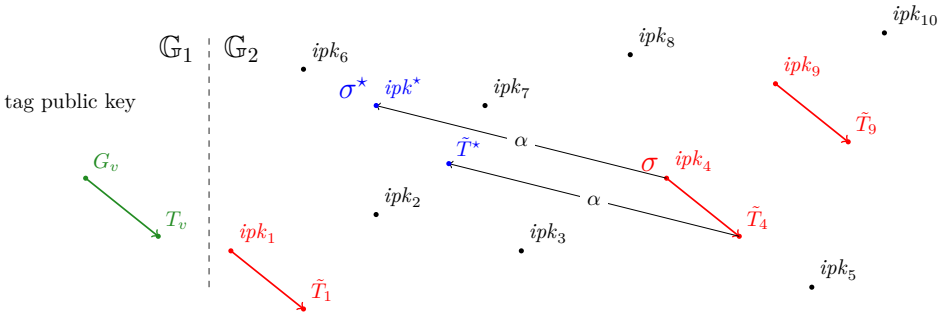
# Type 1 IHAC for BBS

## Presentation



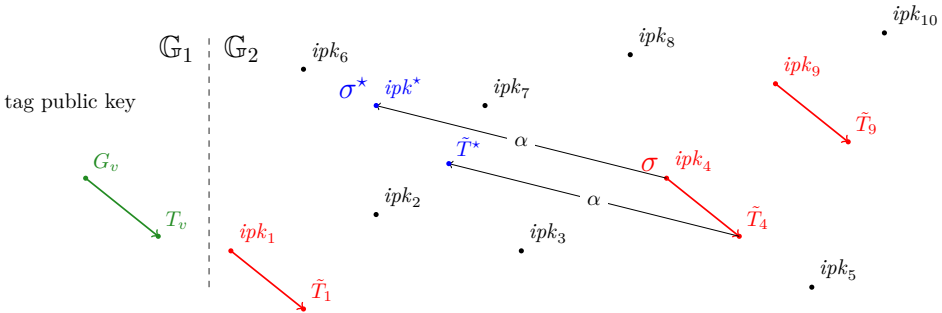
# Type 1 IHAC for BBS

## Presentation



# Type 1 IHAC for BBS

## Presentation




$$pres \leftarrow ipk^\star, \tilde{T}^\star, \pi_{\mathsf{BBS}}$$

# Performance

Running example: credential with 33 attributes, prover proves to be European ($|pol| = 27$)

Compare our $\mathsf{IHBBS}_0$ with BBS + Bobolz et. al.

| IHAC Scheme | Model | Comput. OH | Commun. OH | $pk_{pol}$ size |
|---|---|---|---|---|
| BBS + Bobolz et al. | GGM | $5\mathsf{e}_{\mathbb{G}_1} + 4\mathsf{e}_{\mathbb{G}_2} + 2\mathsf{p}$ | $\mathbb{G}_1 + 3\mathbb{G}_2 + 3\mathbb{Z}_p$ | $\|pol\|(\mathbb{G}_1 + 2\mathbb{G}_2)$ |
| running example | | 8.7 ms | 0.43 KB | 6.5 KB |
| our $\mathsf{IHBBS}_1$ | AGM | $\mathsf{e}_{\mathbb{G}_1} + 2\mathsf{e}_{\mathbb{G}_2}$ | $2\mathbb{G}_2$ | $\|pol\|\mathbb{G}_2 + 2\mathbb{G}_1$ |
| running example | | 2.2 ms | 0.19 KB | 2.6 KB |

# Thank you for your attention!

And to Eysa Lee for the Alice-and-Bobs drawings
https://github.com/eysalee/alice-and-bobs/tree/main

Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin.
Issuer-hiding attribute-based credentials.
In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 158–178. Springer, Cham, December 2021.

Aisling Connolly, Jérôme Deschamps, Pascal Lafourcade, and Octavio Perez-Kempner.
Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric.
In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 249–271. Springer, Cham, December 2022.

Ronald Cramer, Ivan Damgård, and Berry Schoenmakers.
Proofs of partial knowledge and simplified design of witness hiding protocols.
In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Berlin, Heidelberg, August 1994.

Aisling Connolly, Pascal Lafourcade, and Octavio Perez-Kempner.
Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes.
In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 409–438. Springer, Cham, March 2022.

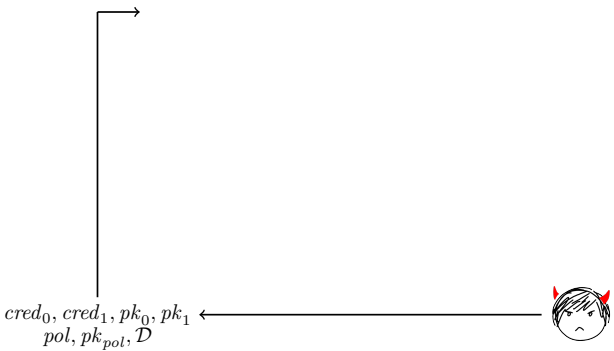
Jonathan Katz and Marek Sefranek.
Issuer hiding for bbs-based anonymous credentials.
*Cryptology ePrint Archive*, 2025.

Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig.
Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials.
In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 30–44. ACM Press, November 2023.

Olivier Sanders and Jacques Traoré.
Compact issuer-hiding authentication, application to anonymous credential.
*PoPETs*, 2024(3):645–658, July 2024.

Stefano Tessaro and Chenzhi Zhu.
Revisiting BBS signatures.
In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721. Springer, Cham, April 2023.

# Unlinkability Experiment

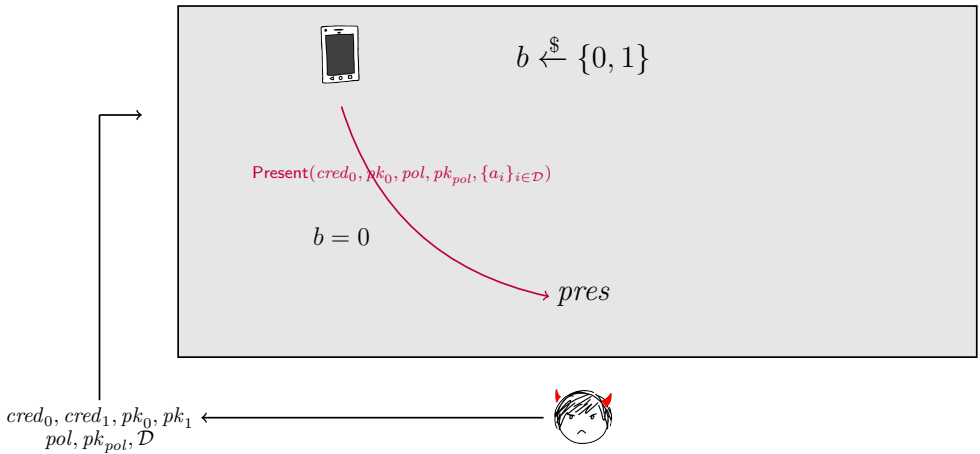


# Unlinkability Experiment

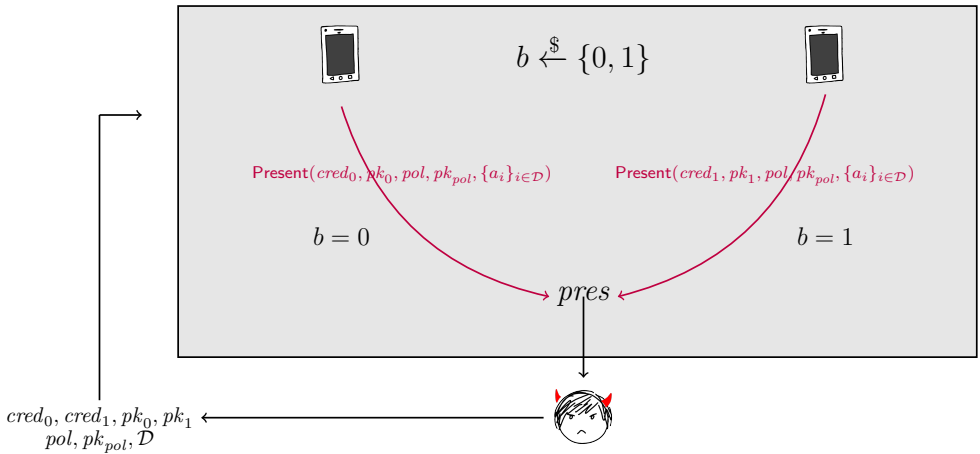
$b \xleftarrow{\$} \{0, 1\}$

$cred_0, cred_1, pk_0, pk_1$
$pol, pk_{pol}, D$

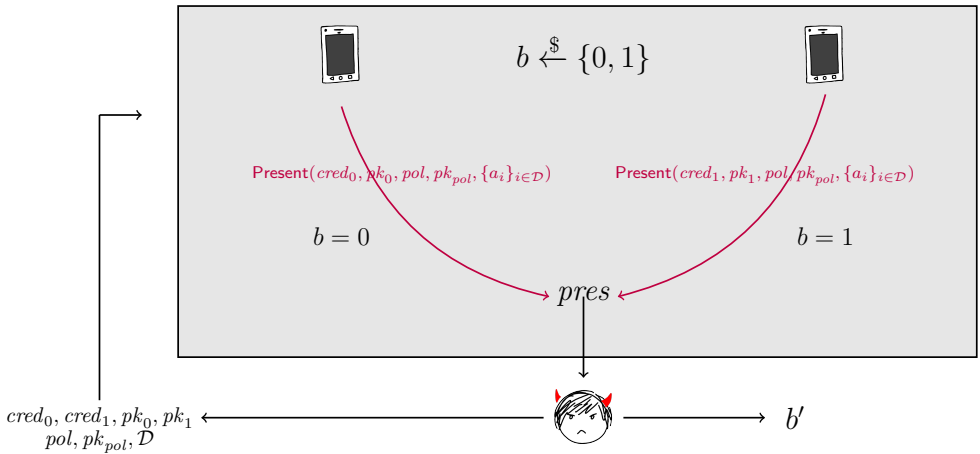
# Unlinkability Experiment



# Unlinkability Experiment



# Unlinkability Experiment



# Unforgeability Experiment

**Adversary**

**Challenger**

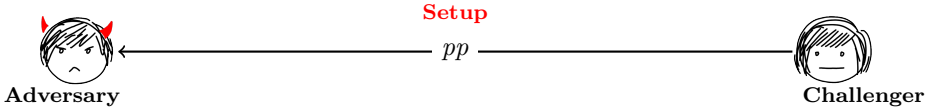
# Unforgeability Experiment

**Setup**

**Adversary**

**Challenger**

# Unforgeability Experiment



# Unforgeability Experiment

**Setup**

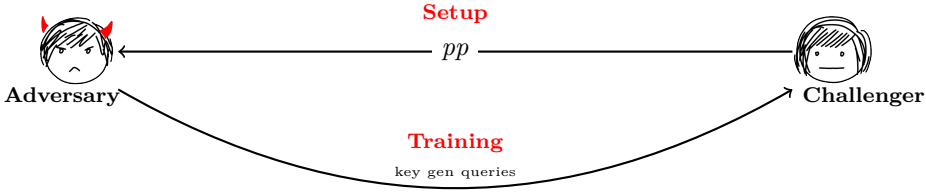
**Adversary**

$pp$

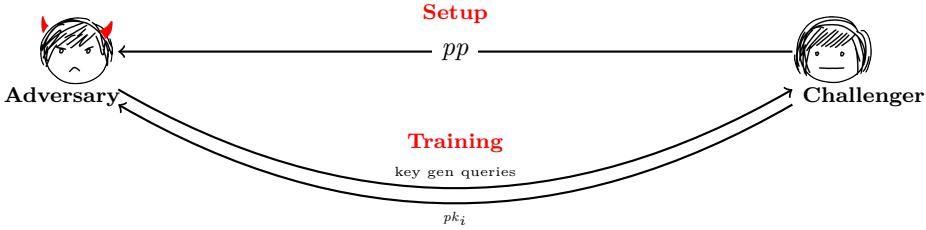
**Challenger**

**Training**

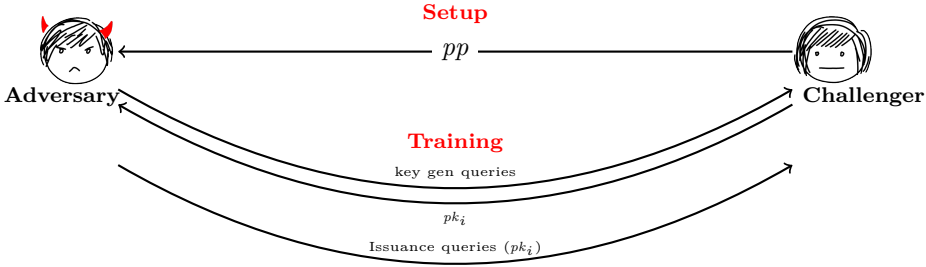
# Unforgeability Experiment



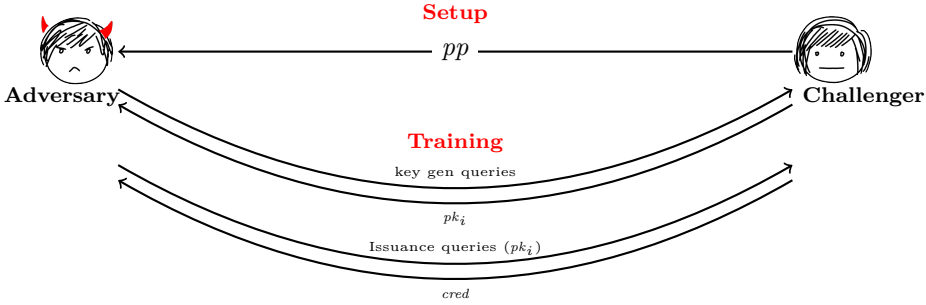
# Unforgeability Experiment



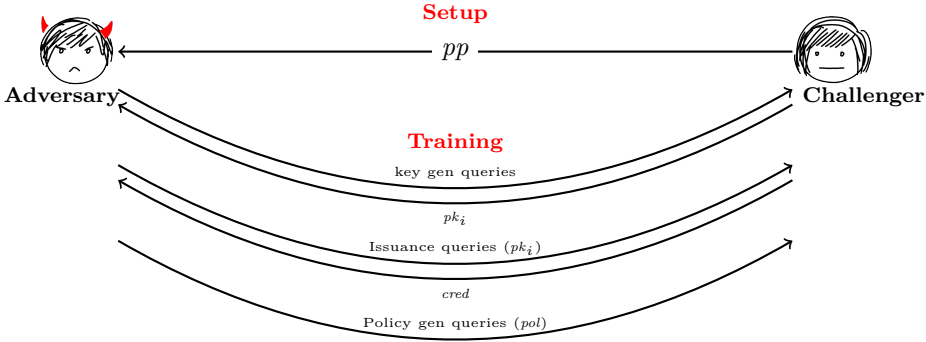
# Unforgeability Experiment



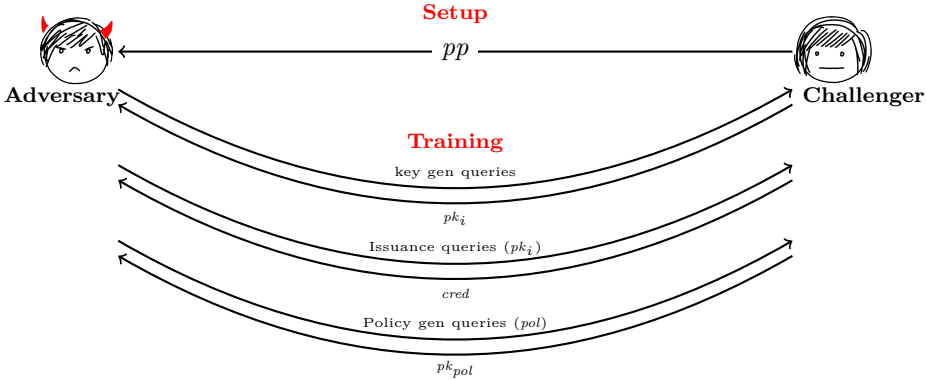
# Unforgeability Experiment



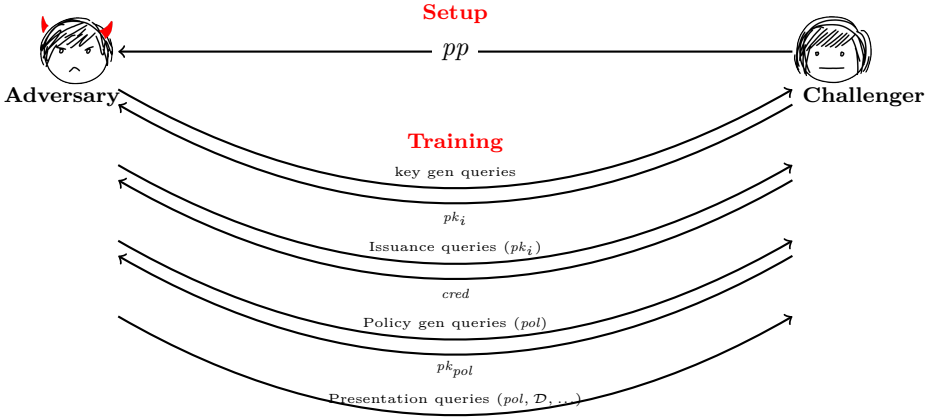
# Unforgeability Experiment



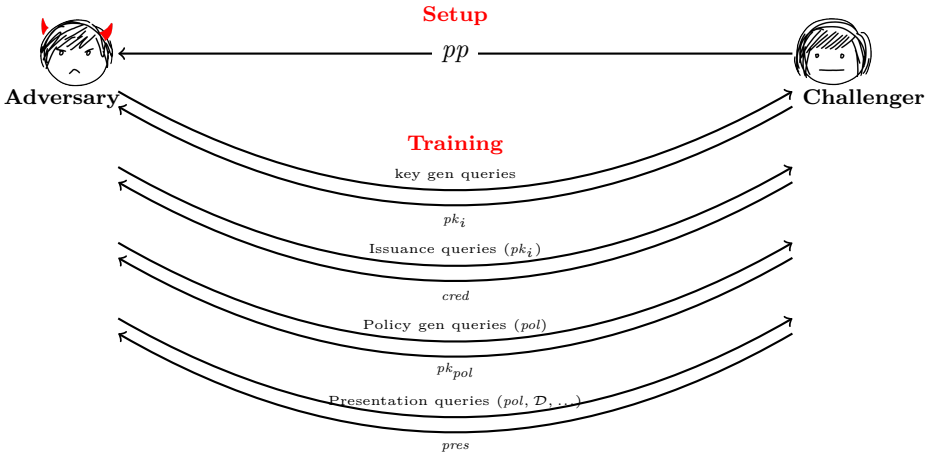
# Unforgeability Experiment



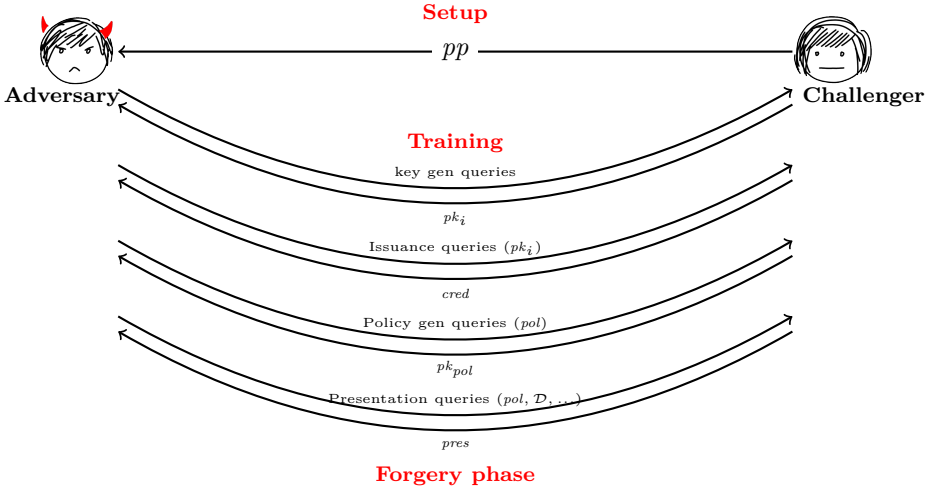
# Unforgeability Experiment



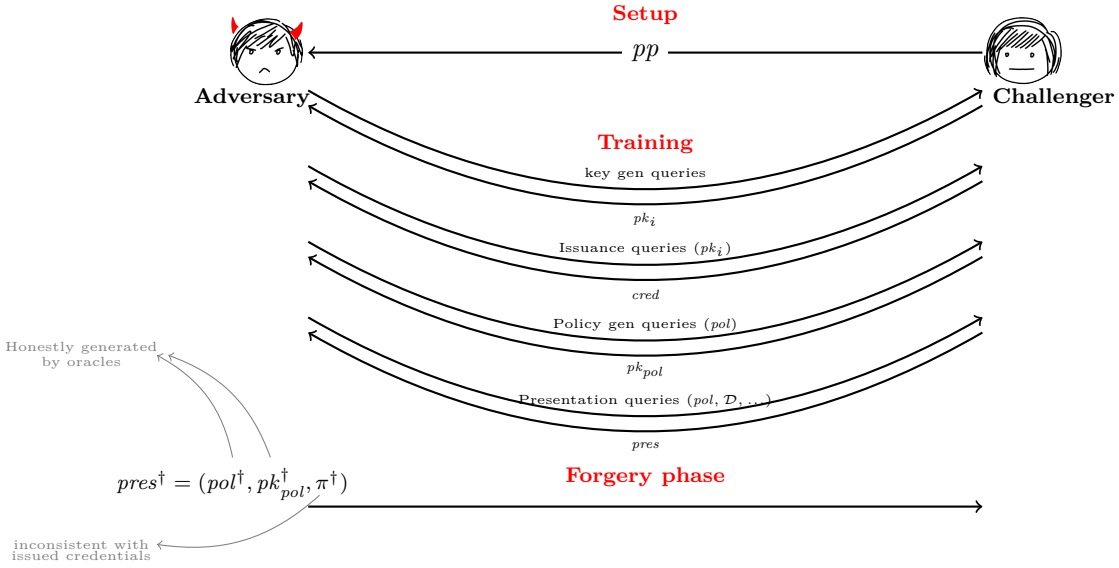
# Unforgeability Experiment



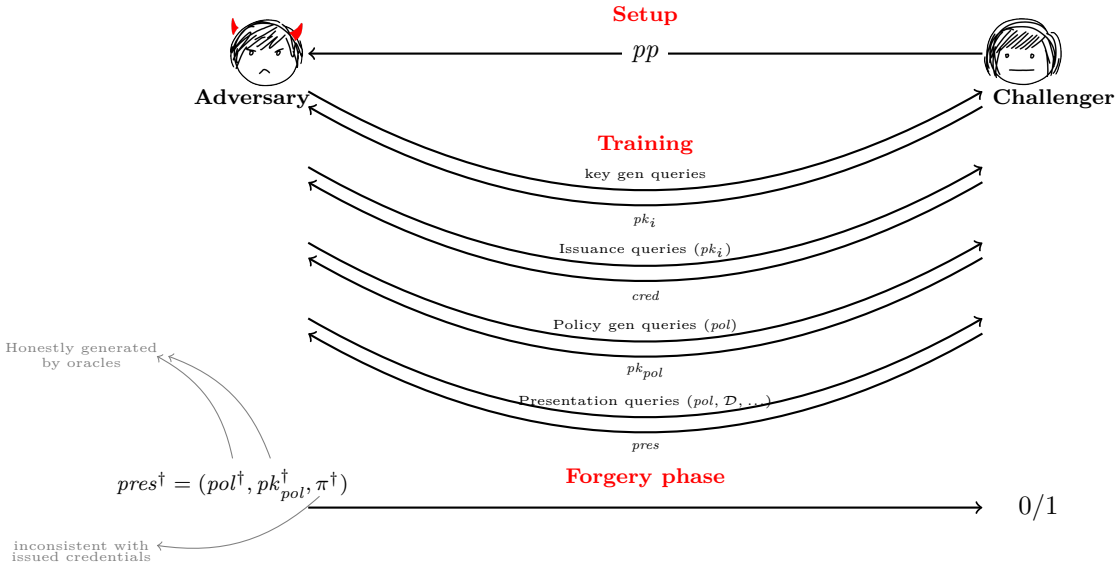
# Unforgeability Experiment



# Unforgeability Experiment

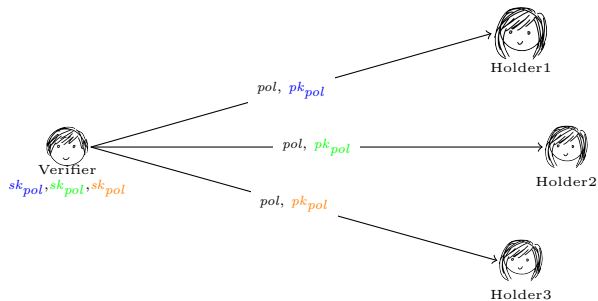


# Unforgeability Experiment



# Freshness of Policies

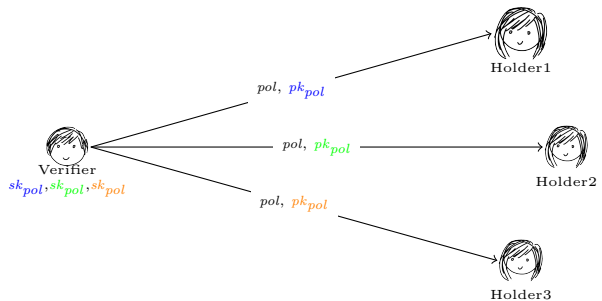
Policies and policy public keys should not be re-used by users to guarantee unlinkability



# Freshness of Policies

Policies and policy public keys should not be re-used by users to guarantee unlinkability

The verifier must send $pol, pk_{pol}$
to the holder before every presentation.

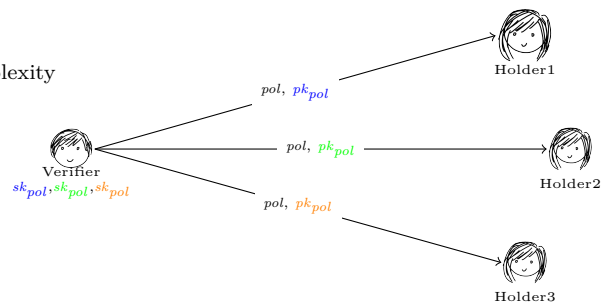


# Freshness of Policies

Policies and policy public keys should not be re-used by users to guarantee unlinkability

The verifier must send $pol, pk_{pol}$ to the holder before every presentation.

This makes the communication complexity linear in the size of the policy.

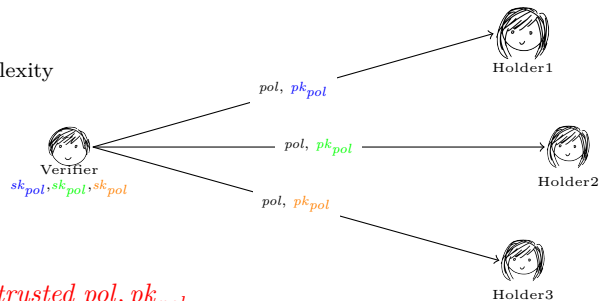


# Freshness of Policies

Policies and policy public keys should not be re-used by users to guarantee unlinkability

The verifier must send $pol, pk_{pol}$ to the holder before every presentation.

This makes the communication complexity linear in the size of the policy.



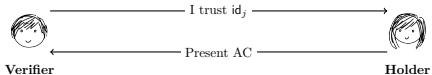

Solution:

Repository with *relevant* and *trusted* $pol, pk_{pol}$

# Policy Repository

Type 0

$\mathsf{id}_1 : \ pol_1$
$\mathsf{id}_2 : \ pol_2$
$\mathsf{id}_3 : \ pol_3$
$\mathsf{id}_4 : \ pol_4$
$\mathsf{id}_5 : \ pol_5$
$\vdots$



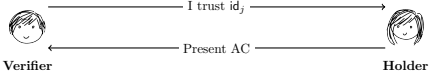
# Policy Repository

Type 0

$\mathsf{id}_1 : pol_1$
$\mathsf{id}_2 : pol_2$
$\mathsf{id}_3 : pol_3$
$\mathsf{id}_4 : pol_4$
$\mathsf{id}_5 : pol_5$
$\vdots$

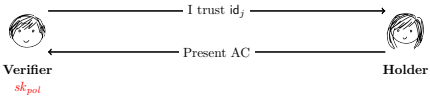
Type 1

$\mathsf{id}_1 : pol_1, pk_{pol_1}$
$\mathsf{id}_2 : pol_2, pk_{pol_2}$
$\mathsf{id}_3 : pol_3, pk_{pol_3}$
$\mathsf{id}_4 : pol_4, pk_{pol_4}$
$\mathsf{id}_5 : pol_5, pk_{pol_5}$
$\vdots$

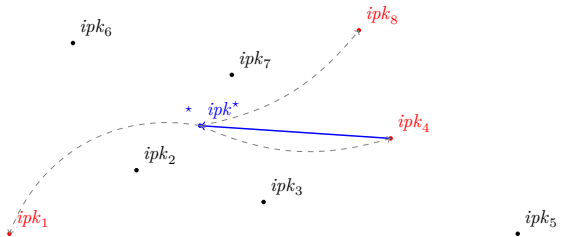


# Policy Repository

| Type 0 | Type 1 | Type 2 |
|---|---|---|
| $\mathsf{id}_1 : pol_1$ | $\mathsf{id}_1 : pol_1, pk_{pol_1}$ | $\mathsf{id}_1 : V_1, pol_1, pk_{pol_{1,1}}$ |
| $\mathsf{id}_2 : pol_2$ | $\mathsf{id}_2 : pol_2, pk_{pol_2}$ | $\mathsf{id}_2 : V_1, pol_2, pk_{pol_{1,2}}$ |
| $\mathsf{id}_3 : pol_3$ | $\mathsf{id}_3 : pol_3, pk_{pol_3}$ | $\vdots$ |
| $\mathsf{id}_4 : pol_4$ | $\mathsf{id}_4 : pol_4, pk_{pol_4}$ | $\mathsf{id}_{82} : V_2, pol_1, pk_{pol_{2,1}}$ |
| $\mathsf{id}_5 : pol_5$ | $\mathsf{id}_5 : pol_5, pk_{pol_5}$ | $\mathsf{id}_{83} : V_2, pol_2, pk_{pol_{2,2}}$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| | | $\mathsf{id}_{1000} : V_{30}, pol_1, pk_{pol_{30,1}}$ |
| | | $\mathsf{id}_{1001} : V_{30}, pol_2, pk_{pol_{30,2}}$ |
| | | $\vdots$ |



# Security Analysis (Hint)

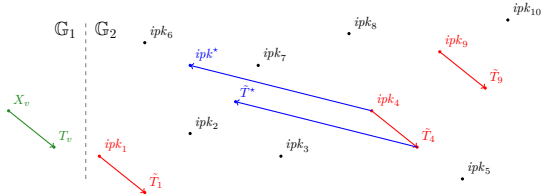



The presentation algorithm is a NIZKP derived from a $\Sigma$-protocol for the relation.

$$\mathcal{R} = \{(\underbrace{(\mathbf{a}, cred, ipk)}_{\text{witness}}, \underbrace{(pol, \mathcal{D})}_{\text{statement}}) : \mathsf{Vf}(cred, \mathbf{a}, ipk),\ ipk \in pol,\ \mathcal{D} \subseteq \mathbf{a}\}$$

# Security Analysis (Hint)

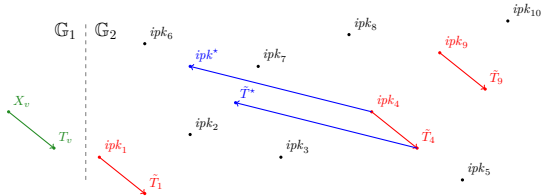
## Unlinkability




The distribution of $ipk^\star$ is uniform in the public key space, $\tilde{T}^\star$ is uniquely determined, and the adapted credential is kept hidden.

# Security Analysis (Hint)

## Unforgeability




Forging a tag is easy. What is hard is forging a tag for a specific public key outside of the policy, or for which the adversary knows the secret key.