Multi-Verifier Keyed Verification Anonymous Credentials

Jan Bobolz[1], Emad Heydari Beni[2,3], Anja Lehmann[4], **Omid Mirzamohammadi[2]**, Cavit Özbay[4], Mahdi Sedaghat[2]

[1] University of Edinburgh, Edinburgh, UK
[2] COSIC, KU Leuven, Leuven, Belgium
[3] Nokia Bell Labs
[4] Hasso Plattner Institute, University of Potsdam, Germany

PrivCrypt, May 10, 2026

Outline

- Motivation
- Problem Statement
- Designing mKVAC
- Our mKVAC: Core Idea
- A Comparison with SAAC

- **Motivation**
- Problem Statement
- Designing mKVAC
- Our mKVAC: Core Idea
- A Comparison with SAAC

Why Anonymous Credentials (AC)?

User's attributes

- Date of birth 1993/02/17
- Nationality BEL
- University KUL

User

Service Provider

What the service provider needs to know

Is the university KUL?

Why Anonymous Credentials (AC)?

User's attributes

- Date of birth 1993/02/17
- Nationality BEL
- University KUL

User

Privacy-preserving Authentication

Not revealing unnecessary data

Service Provider

What the service provider needs to know

Is the university KUL?

Access

Why Anonymous Credentials (AC)?

User's attributes

- Date of birth 1993/02/17
- Nationality BEL
- University KUL

User

Privacy-preserving Authentication

Not revealing unnecessary data

Service Provider

What the service provider needs to know

Is the university KUL?

Access

Real-World Applications

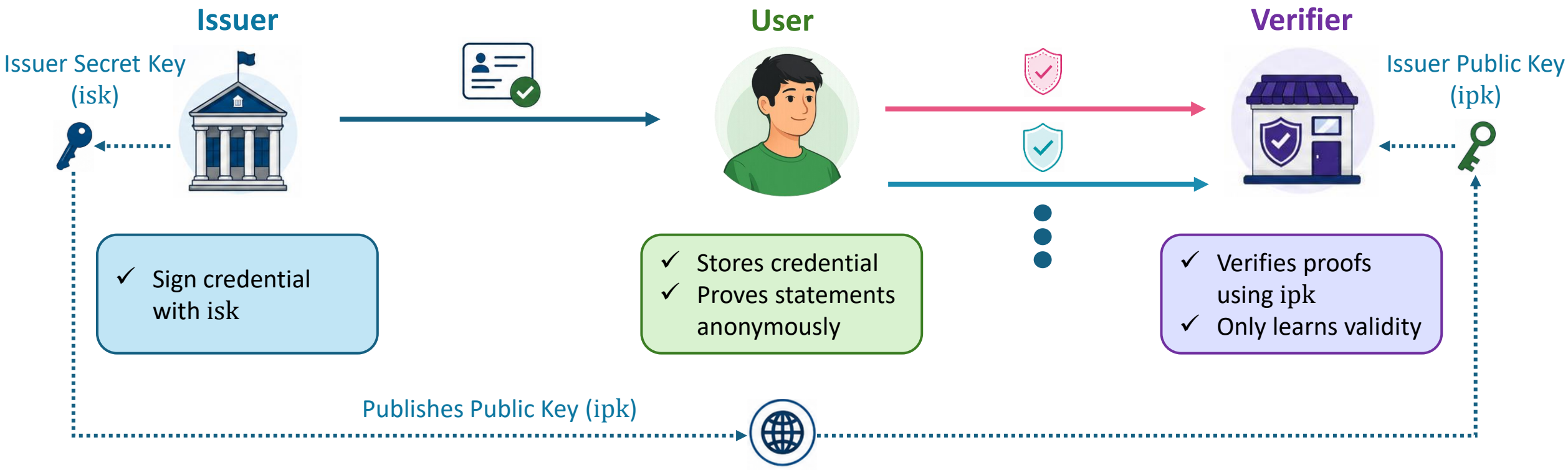
European Digital Identity Wallet

Apple Wallet

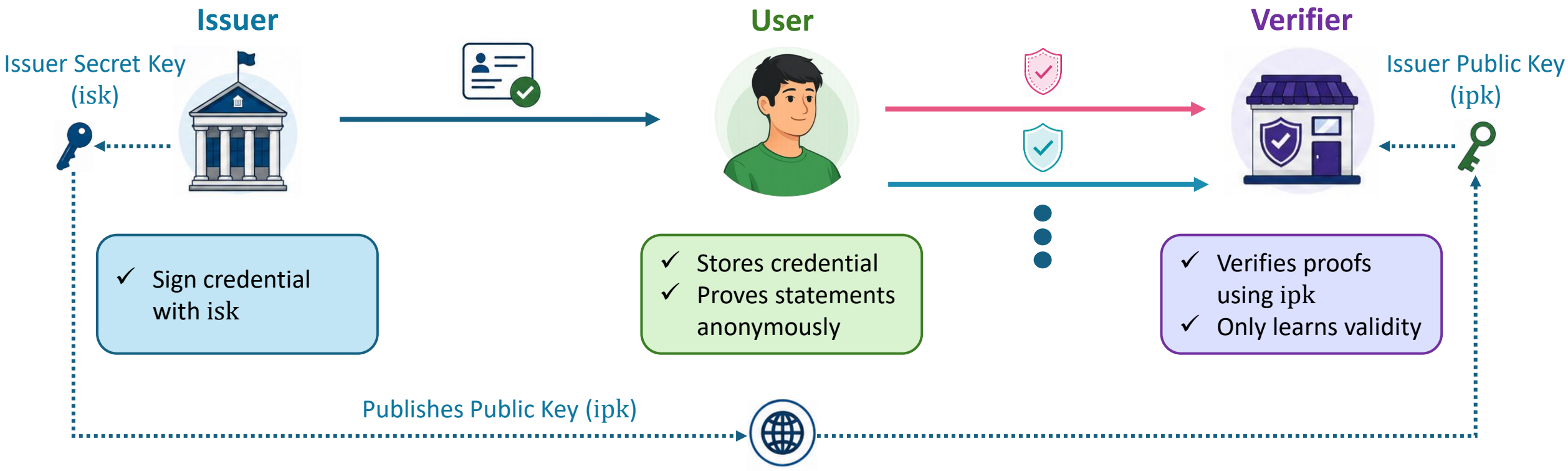
Google Wallet

- Motivation
- **Problem Statement**
- Designing mKVAC
- Our mKVAC: Core Idea
- A Comparison with SAAC

Publicly Verifiable Anonymous Credentials (PVAC)



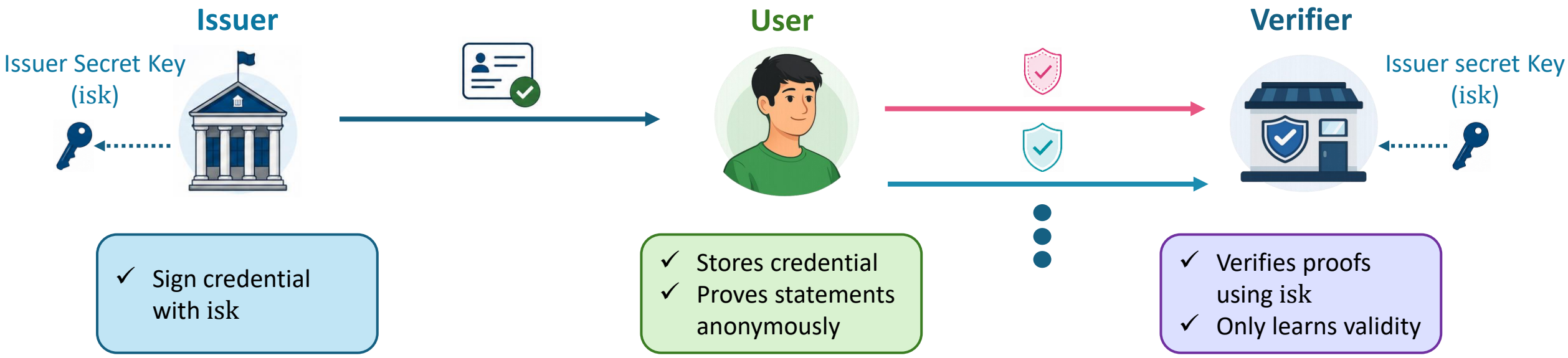
Publicly Verifiable Anonymous Credentials (PVAC)



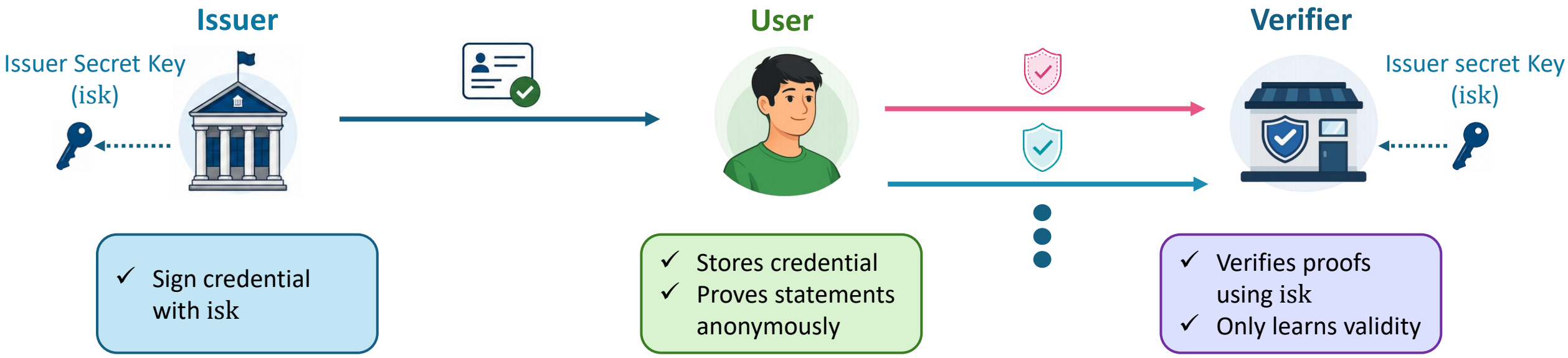
Most efficient constructions use pairings

- Pairing-friendly curves are not recognized by standardization bodies
- Not widely supported in hardware

Keyed-Verification Anonymous Credentials (KVAC)

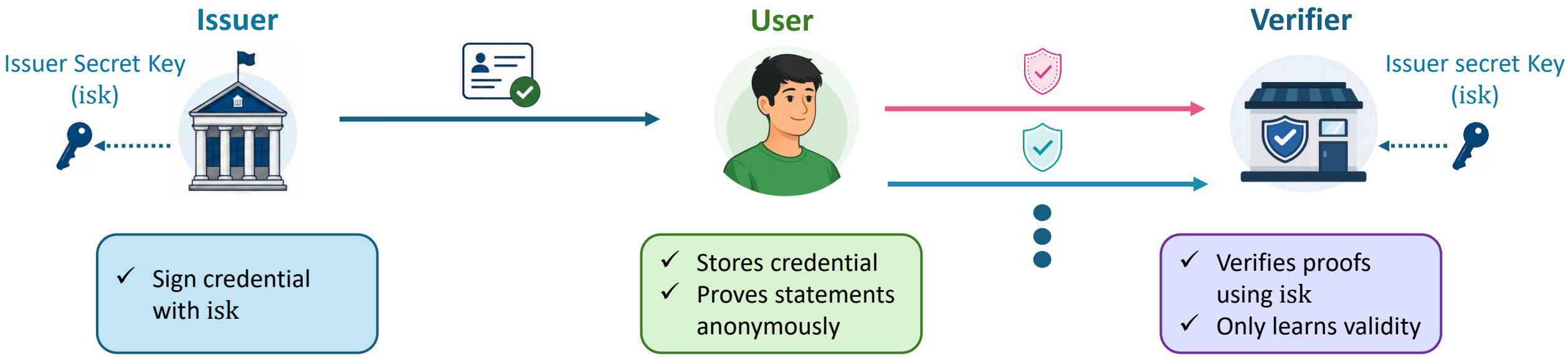


Keyed-Verification Anonymous Credentials (KVAC)



✚ No pairings required
✚ Efficient

Keyed-Verification Anonymous Credentials (KVAC)



+ No pairings required
+ Efficient

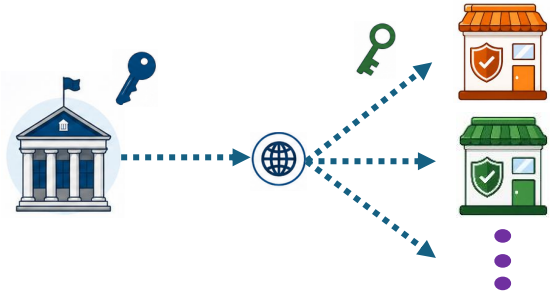
− Limited applicability
− High risk for secret key

Our Research Question

KVAC

- ✚ Pairing-free
- ▬ Applicablilty (one verifier)
- ✚ Multi-show unlinkable

PVAC



- ▬ Pairing-based
- ✚ Applicablilty (Multi-verifier)
- ✚ Multi-show unlinkable

Our Research Question

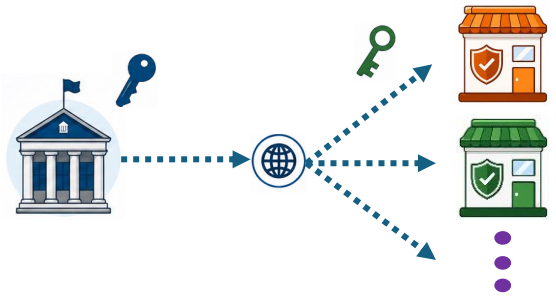
KVAC

- ✚ Pairing-free
- ➖ Applicablilty (one verifier)
- ✚ Multi-show unlinkable

The Gap

- ✚ Pairing-free
- ✚ Applicablilty (Multi-verifier)
- ✚ Multi-show unlinkable

PVAC



- ➖ Pairing-based
- ✚ Applicablilty (Multi-verifier)
- ✚ Multi-show unlinkable

Our Research Question

KVAC

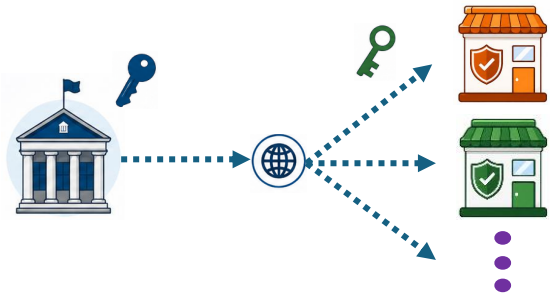
- ✚ Pairing-free
- ▬ Applicablilty (one verifier)
- ✚ Multi-show unlinkable

The Gap

- ✚ Pairing-free
- ✚ Applicablilty (Multi-verifier)
- ✚ Multi-show unlinkable

Can we design such construction?

PVAC

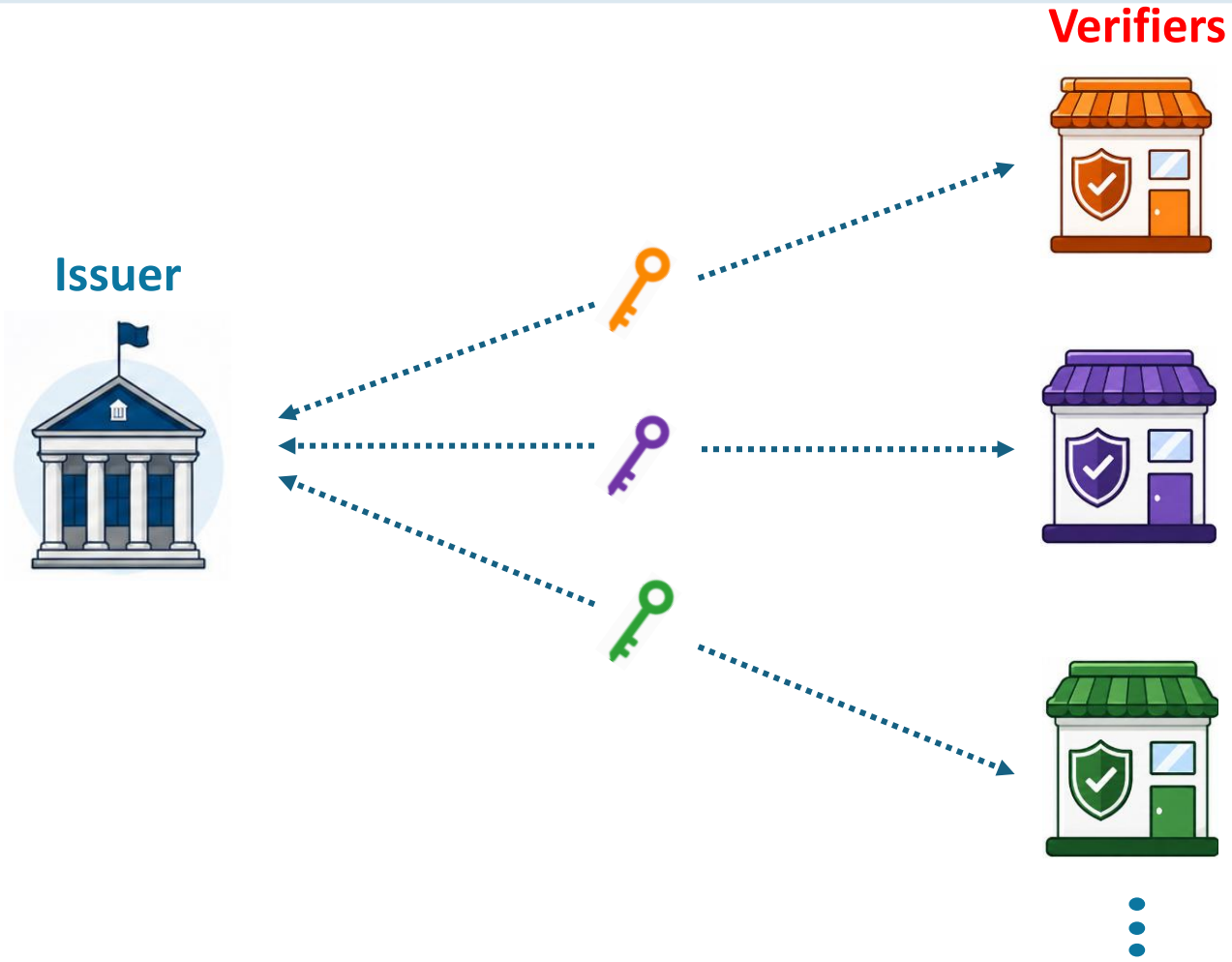


- ▬ Pairing-based
- ✚ Applicablilty (Multi-verifier)
- ✚ Multi-show unlinkable

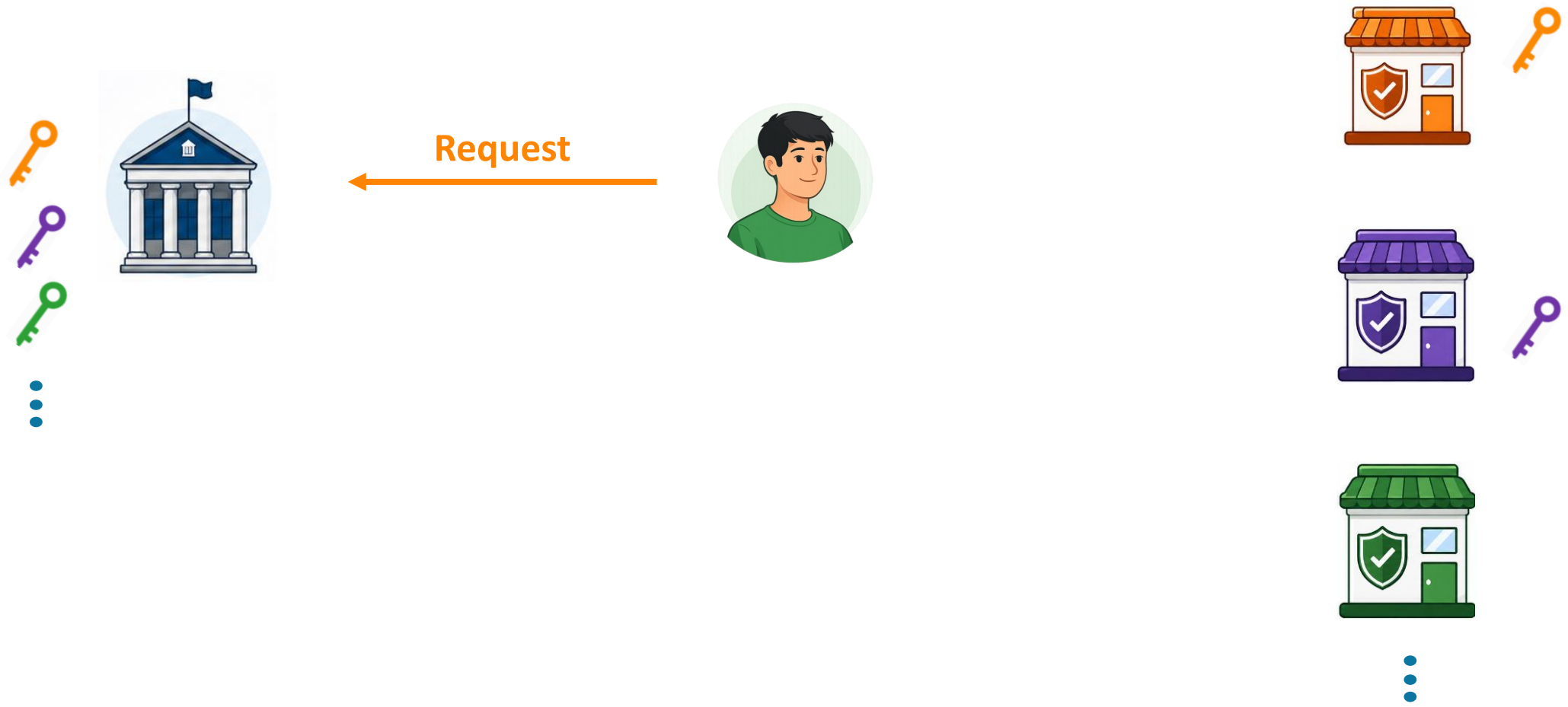
- Motivation
- Problem Statement
- **Designing mKVAC**
- Our mKVAC: Core Idea
- A Comparison with SAAC

Naïve Approach

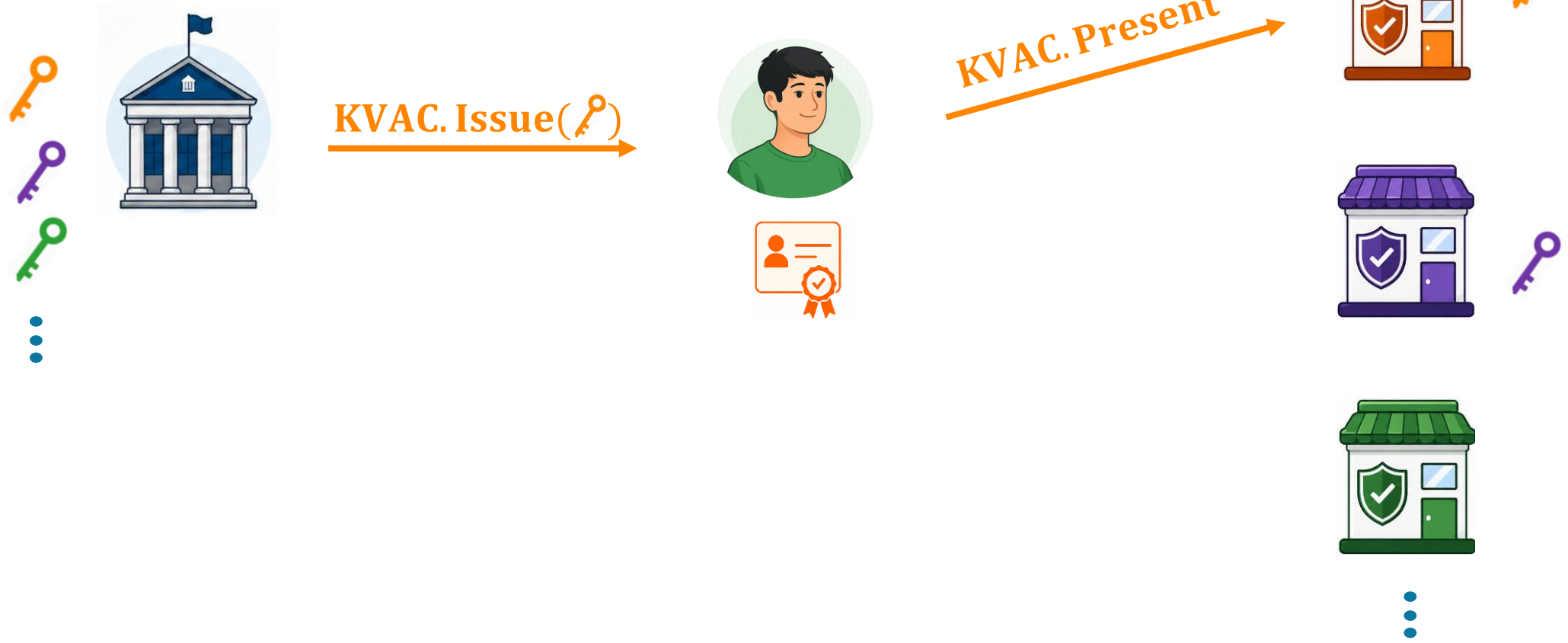
Each verifier shares a secret key with the issuer



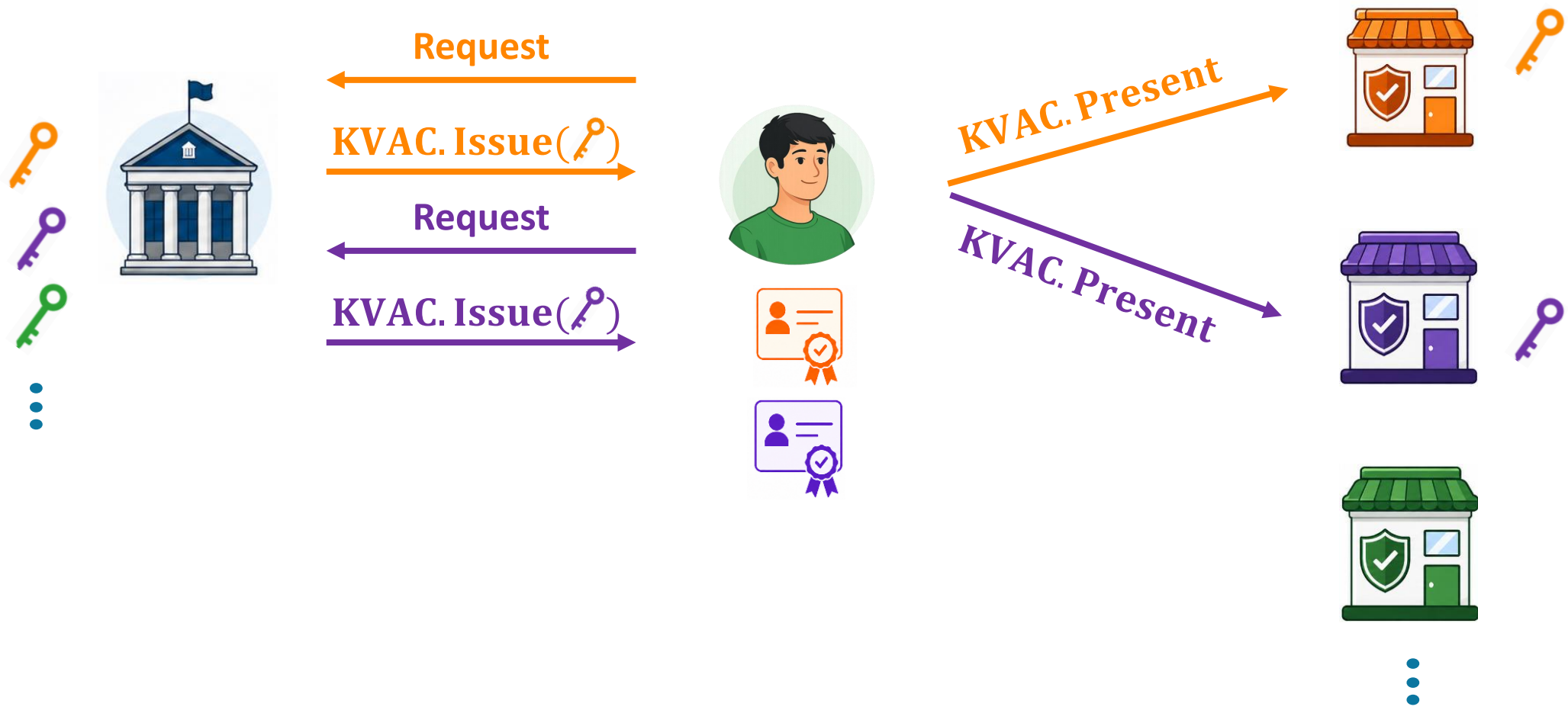
Naïve Approach



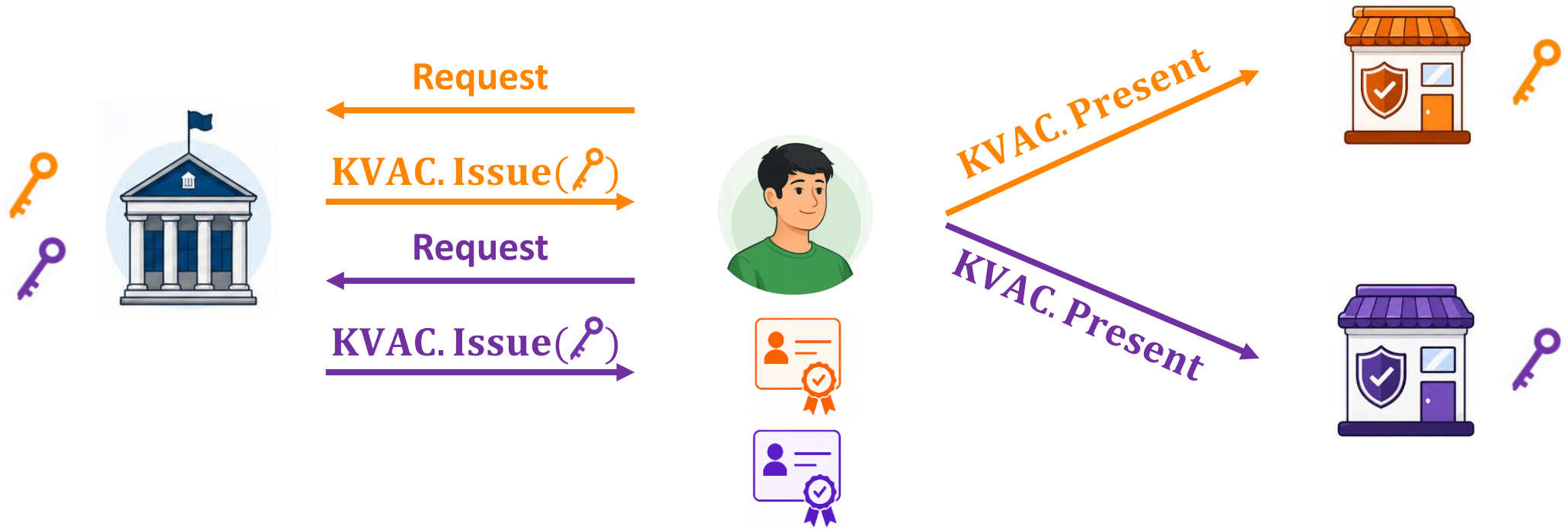
Naïve Approach



Naïve Approach



Naïve Approach



Privacy problem: The issuer learns the services the users want to access

Design Goals for mKVAC

Let each entity has their own public key and private key.

Issuer

$(\mathsf{isk}, \mathsf{ipk})$

Verifiers

$(\mathsf{vsk}, \mathsf{vpk})$

$(\mathsf{vsk}, \mathsf{vpk})$

$(\mathsf{vsk}, \mathsf{vpk})$

Design Goals for mKVAC

(isk, ipk)

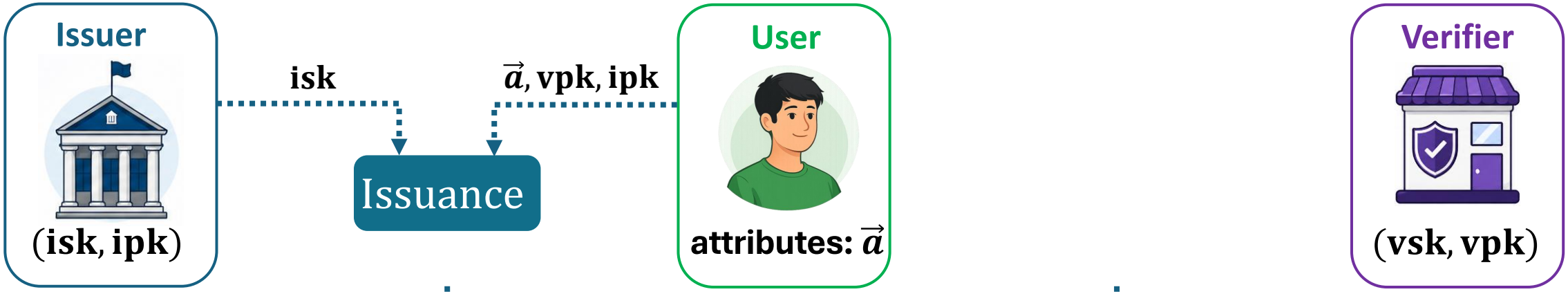
I choose

(vsk, vpk)

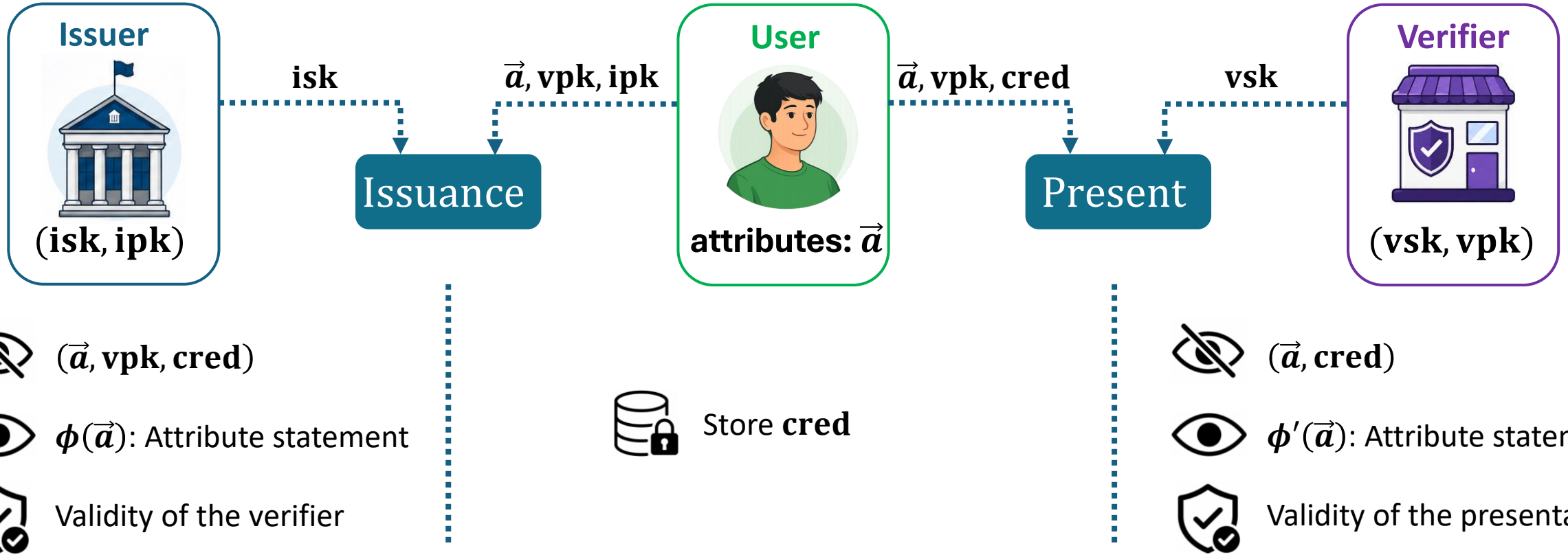
(vsk, vpk)

(vsk, vpk)

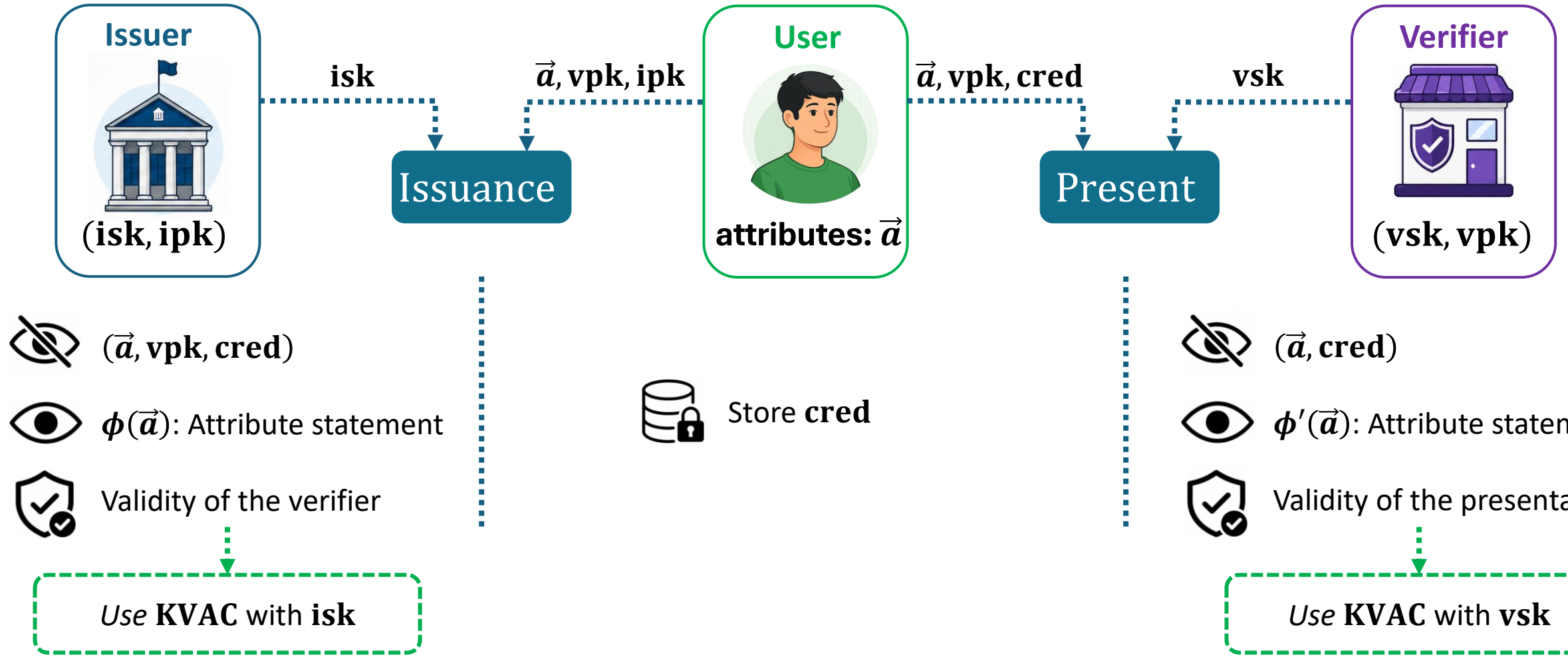
Design Goals for mKVAC



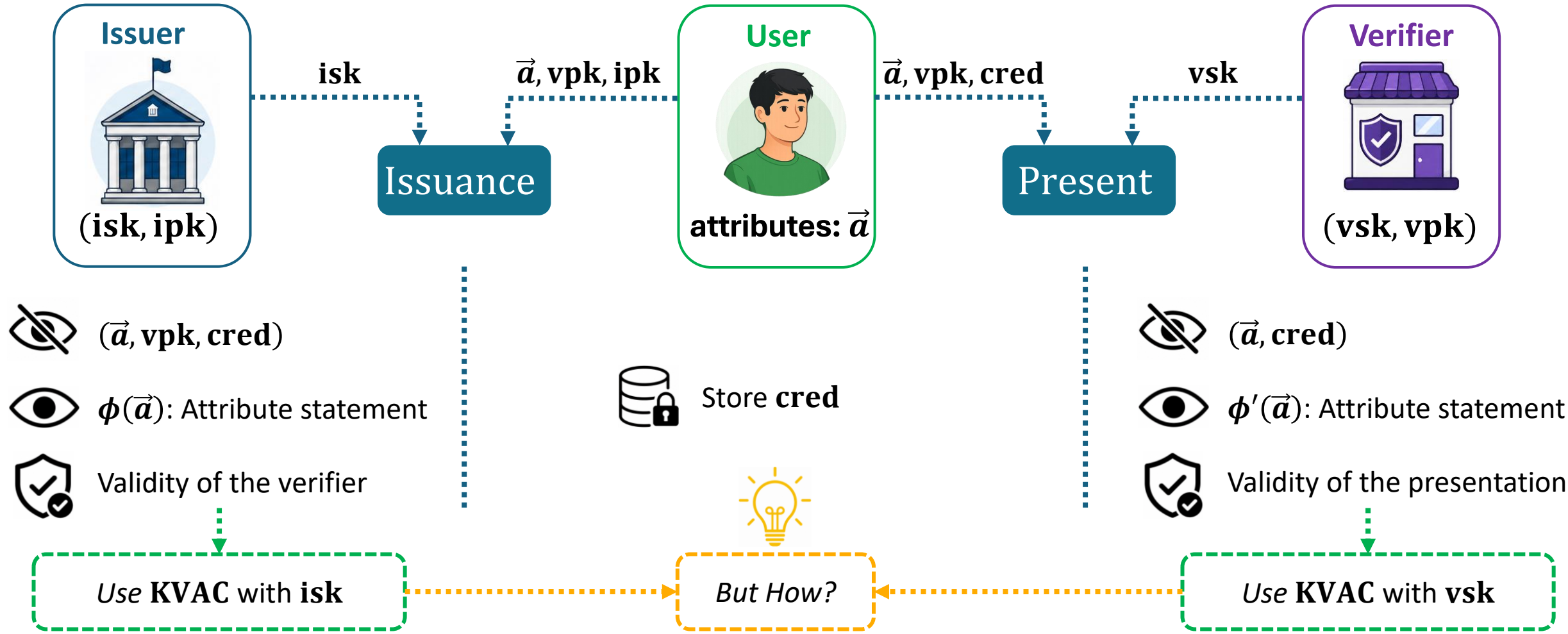
Design Goals for mKVAC



Design Goals for mKVAC



Design Goals for mKVAC



- Motivation
- Problem Statement
- Designing mKVAC
- **Our mKVAC: Core Idea**
- A Comparison with SAAC

KVAC for the Attributes

CMZ-KVAC: Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha, “Algebraic MACs and Keyed-Verification Anonymous Credentials,” CCS 2014.

User

Verifier

Attributes: $\vec{a} \coloneqq (a_1, \ldots, a_n)$

Verifier secret key: $\mathbf{vsk} \coloneqq (x_0, x_1, \ldots, x_n)$

$$\mathbf{cred}_{\vec{a}} \coloneqq \left(uG, \left(x_0 + \sum_{i=1}^{n} x_i a_i\right) uG\right)$$

How to receive $\mathbf{cred}_{\vec{a}}$ from the issuer without revealing the verifier?

KVAC for the Attributes

- ✓ Given $(\boldsymbol{x_i}\mathbf{G})_{\boldsymbol{i=0}}^{\boldsymbol{n}}$, one can compute $\mathbf{cred}_{\vec{\boldsymbol{a}}} \coloneqq (\boldsymbol{u}\mathbf{G}, (\boldsymbol{x_0} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i a_i})\boldsymbol{u}\mathbf{G})$ for any $\vec{\boldsymbol{a}}$

KVAC for the Attributes

- ✓ Given $(\boldsymbol{x_i}\mathbf{G})_{\boldsymbol{i=0}}^{\boldsymbol{n}}$, one can compute $\mathbf{cred}_{\vec{\boldsymbol{a}}} \coloneqq (\boldsymbol{u}\mathbf{G}, (\boldsymbol{x_0} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i a_i})\boldsymbol{u}\mathbf{G})$ for any $\vec{\boldsymbol{a}}$
- ✓ If $\boldsymbol{x_0}\mathbf{G}$ is hidden, computing $\mathbf{cred}_{\vec{\boldsymbol{a}}}$ is no longer possible

KVAC for the Attributes

- ✓ Given $(\boldsymbol{x_i}\mathbf{G})_{\boldsymbol{i=0}}^{\boldsymbol{n}}$, one can compute $\mathbf{cred}_{\vec{\boldsymbol{a}}} \coloneqq (\boldsymbol{u}\mathbf{G}, (\boldsymbol{x_0} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i a_i})\boldsymbol{u}\mathbf{G})$ for any $\vec{\boldsymbol{a}}$
- ✓ If $\boldsymbol{x_0}\mathbf{G}$ is hidden, computing $\mathbf{cred}_{\vec{\boldsymbol{a}}}$ is no longer possible
- ✓ For each verifier, publish: $\{\boldsymbol{x_0}\mathbf{G} + \boldsymbol{v}\mathbf{E}, \boldsymbol{v}\mathbf{G}, (\boldsymbol{x_i}\mathbf{G})_{\boldsymbol{i=1}}^{\boldsymbol{n}}\}$, where $(\boldsymbol{x_0}\mathbf{G} + \boldsymbol{v}\mathbf{E}, \boldsymbol{v}\mathbf{G})$ is an Elgamal encryption of $\boldsymbol{x_0}\mathbf{G}$ under public key $\mathbf{E} = \boldsymbol{e}\mathbf{G}$, and the issuer holds the secret key $\boldsymbol{e}$.

KVAC for the Attributes

isk includes $\boldsymbol{e}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \dots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \dots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \dots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$

⋮

KVAC for the Attributes

isk includes e

$$\mathbf{vpk} := \{X'_0, X_1, \ldots, X_n\} := \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$$

$$\mathbf{vpk} := \{X'_0, X_1, \ldots, X_n\} := \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$$

$$\mathbf{vpk} := \{X'_0, X_1, \ldots, X_n\} := \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}$$

All encrypted under the same public key

KVAC for the Attributes

Issuer

User

Compute $\mathbf{cred}'_{\vec{a}} := \left(u\mathbf{G}, u(X'_0 + \sum_{i=1}^{n} a_i X_i)\right)$

$C_{\vec{a}}$: Commit on $\mathbf{cred}'_{\vec{a}}$

$C_{\vec{a}}$

Compute $\mathbf{bcred}_{\vec{a}}$: Decrypt $C_{\vec{a}}$ using e

(blindly signing)

$\mathbf{bcred}_{\vec{a}}$

Compute $\mathbf{cred}_{\vec{a}}$: Unblind $\mathbf{bcred}_{\vec{a}}$

KVAC for the Attributes

Issuer

User

Compute $\mathbf{cred}'_{\vec{a}} := \left(u\mathbf{G}, u(X'_0 + \sum_{i=1}^{n} a_i X_i)\right)$

$C_{\vec{a}}$: Commit on $\mathbf{cred}'_{\vec{a}}$

User can cheat here by not using correct $\mathbf{vpk}$

$C_{\vec{a}}$

Compute $\mathbf{bcred}_{\vec{a}}$: Decrypt $C_{\vec{a}}$ using e

(blindly signing)

$\mathbf{bcred}_{\vec{a}}$

Compute $\mathbf{cred}_{\vec{a}}$: Unblind $\mathbf{bcred}_{\vec{a}}$

KVAC for the verifiers' public keys

We designed a KVAC over group elements based on BBS signatures: Dan Boneh, Xavier Boyen, and Hovav Shacham, "Short group signatures," CRYPTO 2004

$\mathbf{isk}$ includes $\boldsymbol{e}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \ldots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}, \mathbf{cred}_{\mathbf{vpk}}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \ldots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}, \mathbf{cred}_{\mathbf{vpk}}$

$\mathbf{vpk} \coloneqq \{X_0', X_1, \ldots, X_n\} \coloneqq \{x_0\mathrm{G} + v\mathrm{E}, v\mathrm{G}, (x_i\mathrm{G})_{i=1}^{n}\}, \mathbf{cred}_{\mathbf{vpk}}$

⋮

KVAC for the verifiers' public keys

Issuer

User

Compute $\mathbf{cred}'_{\vec{a}} := \left(u\mathbf{G}, u(X'_0 + \sum_{i=1}^{n} a_i X_i)\right)$

$C_{\vec{a}}$: Commit on $\mathbf{cred}'_{\vec{a}}$

$C_{\vec{a}}$, $\mathbf{KVAC.Present}_{\mathbf{vpk}}$, π ←

Check the validity of $\mathbf{KVAC.Present}_{\mathbf{vpk}}$, π

Compute $\mathbf{bcred}_{\vec{a}}$: Decrypt $C_{\vec{a}}$ using e

(blindly signing)

$\mathbf{bcred}_{\vec{a}}$ →

Compute $\mathbf{cred}_{\vec{a}}$: Unblind $\mathbf{bcred}_{\vec{a}}$

Resulting mKVAC

We designed an anonymous credential scheme which is:

- ✓ Unforgeable in Generic Group Model
- ✓ Statistically anonymous
- ✓ Multi-show unlinkable
- ✓ Pairing-free
- ✓ Suitable for multi-verifiers setting

- Motivation
- Problem Statement
- Designing mKVAC
- Our mKVAC: Core Idea
- **A Comparison with SAAC**

Server Aided Anonymous Credentials (SAAC)

Rutchathon Chairattana, Franklin Harding, Anna Lysyanskaya, and Stefano Tessaro, Server-Aided Anonymous Credentials, CRYPTO 2025

Issuer

isk

User

Verifier

ipk

Helper

isk

Server Aided Anonymous Credentials (SAAC)

Issuer

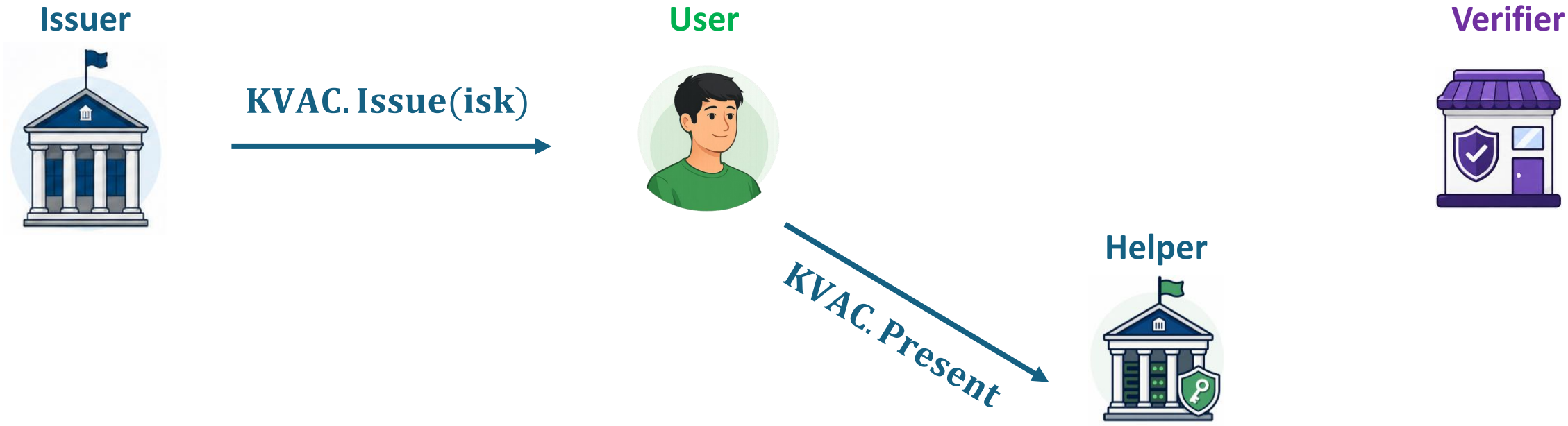
$\mathrm{KVAC.Issue}(\mathrm{isk})$

User

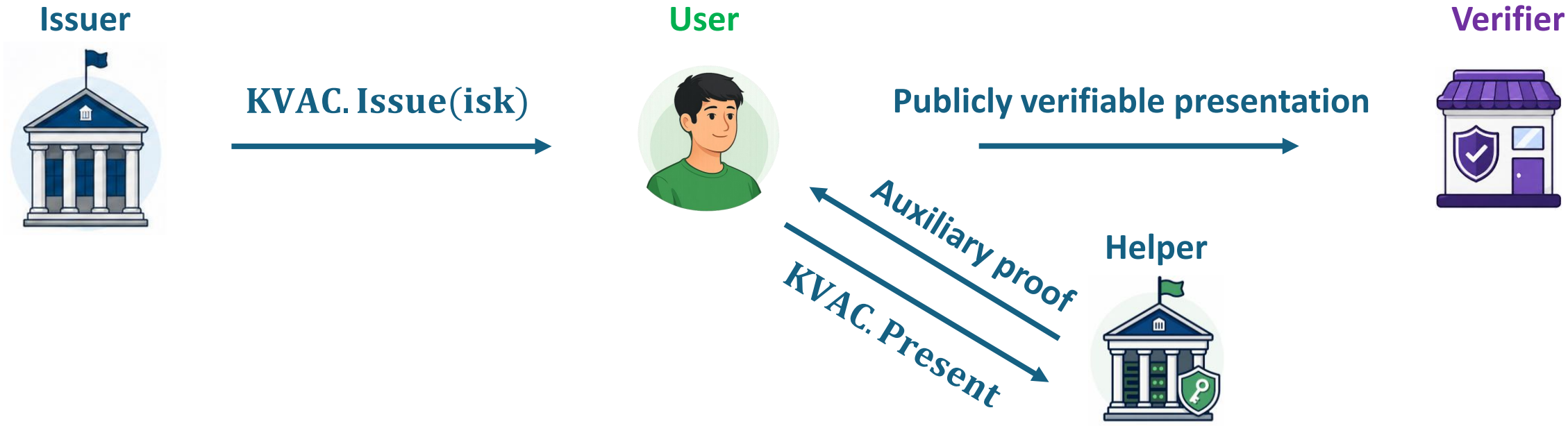
Verifier

Helper

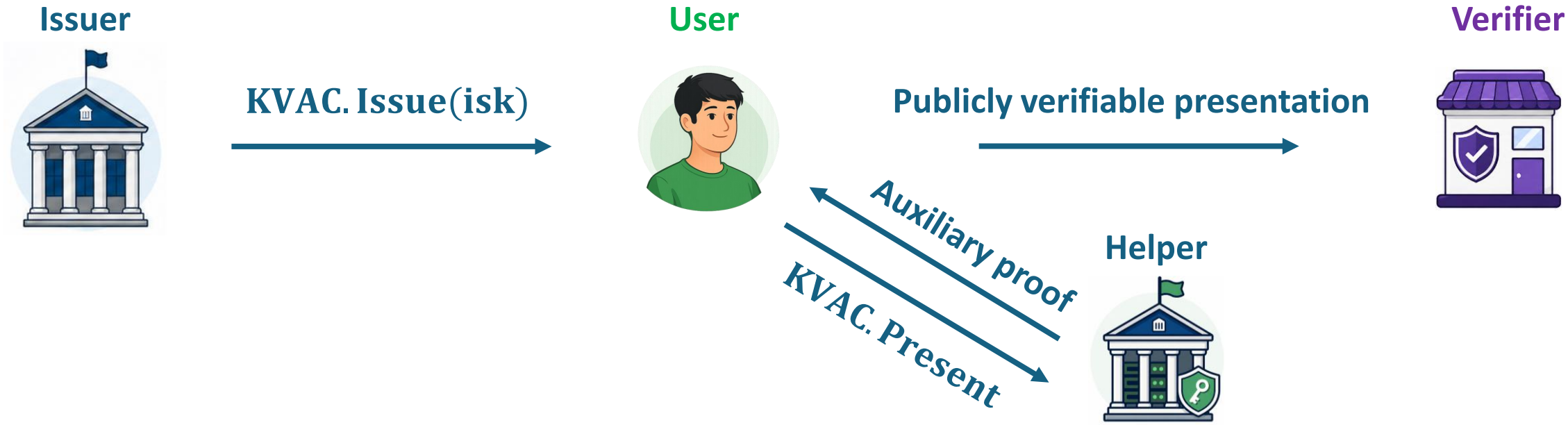
Server Aided Anonymous Credentials (SAAC)



Server Aided Anonymous Credentials (SAAC)



SAAC vs. mKVAC



SAAC	mKVAC
✓ Pairing-free	✓ Pairing-free
✓ Publicly verifiable AC	✓ Verifier-specific AC
✓ One-show unlinkable	✓ Multi-show unlinkable

Thanks for your attention!

Full paper