Private Delegation of (Non-)Membership Proof Updates in Cryptographic Accumulators:

Applications to Revocable Anonymous Credentials

NIST Workshop on Privacy-Enhancing Cryptography (WPEC) 2026
2026 October 26-28 (online)

István András Seres
Princeton University

ePrint

GitHub

Joint work with



Bence Soóki-Tóth
Aarhus University



Botond Glasz
Eötvös Loránd University



Alireza Kavousi
University of Cambridge

EUDI Wallets and anonymous credentials

The wallet

The **European Digital Identity (EUDI) Wallet** is intended to let people hold and present digitally issued attestations, affecting $\approx$ 450M users by late 2026.

With **anonymous credentials**, a holder can selectively disclose attributes and prove statements in zero knowledge, without disclosing a stable identifier.

Background: Flamini et al., *Comparing Privacy-Preserving Revocation for the EUDI Wallet*, ePrint 2026/1824, Secs. 1–3; Camenisch–Lysyanskaya (CRYPTO 2002).

EUDI Wallets and anonymous credentials

The wallet

The **European Digital Identity (EUDI) Wallet** is intended to let people hold and present digitally issued attestations, affecting $\approx$ 450M users by late 2026.

With **anonymous credentials**, a holder can selectively disclose attributes and prove statements in zero knowledge, without disclosing a stable identifier.

But what if a credential is revoked?

A status manager updates the list of valid or revoked credential handles.

A holder must prove **membership** in the valid set or **non-membership** in the revoked set, without revealing its own handle.

Background: Flamini et al., *Comparing Privacy-Preserving Revocation for the EUDI Wallet*, ePrint 2026/1824, Secs. 1–3; Camenisch–Lysyanskaya (CRYPTO 2002).

EUDI Wallets and anonymous credentials

The wallet

The **European Digital Identity (EUDI) Wallet** is intended to let people hold and present digitally issued attestations, affecting $\approx$ 450M users by late 2026.

With **anonymous credentials**, a holder can selectively disclose attributes and prove statements in zero knowledge, without disclosing a stable identifier.

But what if a credential is revoked?

A status manager updates the list of valid or revoked credential handles.

A holder must prove **membership** in the valid set or **non-membership** in the revoked set, without revealing its own handle.

Accumulators compress revocation state into a small, verifiable digest.

Background: Flamini et al., *Comparing Privacy-Preserving Revocation for the EUDI Wallet*, ePrint 2026/1824, Secs. 1–3;

Camenisch–Lysyanskaya (CRYPTO 2002).

An anonymous credential contains a revocation handle

🏛 Government signed

```
{
  "credentialType":
    "Driver's license",
  "name":  "John von Neumann",
  "age":  35,
  "sex":  "male",
  "address":  "Princeton, NJ",
  "revocationHandle":
    "7A93...F21B"
}
```



John von Neumann

Fictional driver's-license example; not a standardized EUDI credential encoding.

An anonymous credential contains a revocation handle

🏛 Government signed

```
{
  "credentialType":
    "Driver's license",
  "name":  "John von Neumann",
  "age":  35,
  "sex":  "male",
  "address":  "Princeton, NJ",
  "revocationHandle":
    "7A93...F21B"
}
```



John von Neumann

Fictional driver's-license example; not a standardized EUDI credential encoding.

An issuer signs attributes

The revocation handle is **also an attribute** of the credential.

During a zero-knowledge (ZK) presentation, the holder proves validity and non-revocation *without revealing* this handle.

An anonymous credential contains a revocation handle

🏛 Government signed

```
{
  "credentialType":
    "Driver's license",
  "name":  "John von Neumann",
  "age":  35,
  "sex":  "male",
  "address":  "Princeton, NJ",
  "revocationHandle":
    "7A93...F21B"
}
```



John von Neumann

Fictional driver's-license example; not a standardized EUDI credential encoding.

An issuer signs attributes

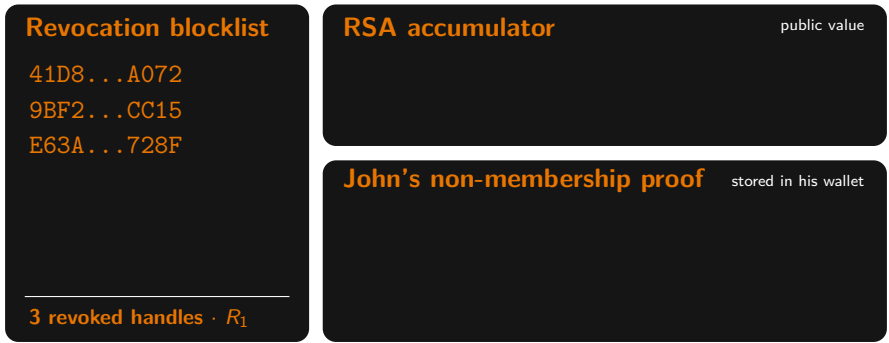
The revocation handle is **also an attribute** of the credential.

During a zero-knowledge (ZK) presentation, the holder proves validity and non-revocation *without revealing* this handle.

Revocation blocklist R_t

```
41D8...A072
9BF2...CC15
E63A...728F
```

Revocation updates change John's status proof



John's handle is hidden inside the credential and is **not yet revoked**.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

Revocation updates change John's status proof

Revocation blocklist

41D8...A072
9BF2...CC15
E63A...728F

3 revoked handles · R_1

RSA accumulator public value

A_1

0xF1FD4C22D4B998AC1C9DBF364C29C706DC0C6EA514DAB830
13D3057C94F4F3E7308202B6C1C6CF5F211CE138D057A32F

John's non-membership proof stored in his wallet

$\pi_1 = (a_1, B_1)$ ✅

a: 0x7063C75937163DBA3903E882FE6BEAE5

B: 0x744FC76724A19F1A8F125F8369F824FE
4489759FCE5CCA8BC73527926347D8CB

The accumulator A_1 appears; John holds a non-membership proof π_1 for R_1.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

Revocation updates change John's status proof

Revocation blocklist

41D8...A072
9BF2...CC15
E63A...728F
+ 2A4E...9B31

4 revoked handles · R_2

RSA accumulator public value

A_2

0xDBA7D553C082E3E6E975D60C5C0B13C98CC90EF91C9A482C
6C9A7C778FD26EE2B204A4F0A66CA191FB1F84F92F725DEA

John's non-membership proof stored in his wallet

$\pi_2 = (a_2, B_2)$ ✅

a: 0xF97164D2BF58C665B488A208F28A3DD8

B: 0xCD98FEFCFF76AFA6FAD359551B3BF51E
5A0FD9B59166D28D0555822024A1D8F8

An unrelated revocation changes $R_1 \to R_2$: both A_2 and John's proof π_2 change.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

Revocation updates change John's status proof

Revocation blocklist

41D8...A072
9BF2...CC15
E63A...728F
2A4E...9B31
+ 54C2...6FD0

5 revoked handles · R_3

RSA accumulator public value

A_3

0x70DFAEADD89A2E62BB62F9C130A2EA06397E50B4847790E7
DEEAA9FA9CA96627C11B6655748663FC9980FD0774160CCA

John's non-membership proof stored in his wallet

$\pi_3 = (a_3, B_3)$ ✅

a: 0x4123EB1E5A5550484A4A4A871C589227

B: 0xC5755D953C96CB9EC3CF59FB390DEE1B
CECAFE4B86DD5CFE00F85D83372D485E

Another driver's license is revoked: John remains valid, but his proof must be refreshed again.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

Revocation updates change John's status proof

Revocation blocklist

41D8...A072
9BF2...CC15
E63A...728F
2A4E...9B31
54C2...6FD0
+ 7A93...F21B
John's handle (bad driver)

6 revoked handles · R_4

RSA accumulator public value

A_4
0x58A4556D4187FCF9C038F20E7BDA82EF90E8EF84D2D669D4
765507A2E8805B1B9AEA800C561F452B405D316FBA85BDFA

John's non-membership proof stored in his wallet

No valid non-membership proof

$x_{\text{John}} \in R_4$ ✖

Driver's license revoked
the wallet cannot update or create
a non-membership proof anymore.

In this fictional example, John's license is finally revoked after his notoriously bad driving.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

Revocation updates change John's status proof

Revocation blocklist

41D8...A072
9BF2...CC15
E63A...728F
2A4E...9B31
54C2...6FD0
7A93...F21B
John's handle (bad driver)

6 revoked handles · R_4

RSA accumulator public value

A_4

0x58A4556D4187FCF9C038F20E7BDA82EF90E8EF84D2D669D4
765507A2E8805B1B9AEA800C561F452B405D316FBA85BDFA

John's non-membership proof stored in his wallet

No valid non-membership proof

$x_{\text{John}} \in R_4$ ✖

Driver's license revoked
the wallet cannot update or create
a non-membership proof anymore.

Once $x_{\text{John}} \in R_4$, John can no longer prove non-membership in the revocation set.

Schematic values: hexadecimal strings are illustrative (not computed RSA witnesses).

The problem in one sentence

How can an offline wallet refresh its **non-revocation proof**

without revealing **which credential** it holds?

The problem in one sentence

How can an offline wallet refresh its **non-revocation proof** without revealing **which credential** it holds?

Privacy

Credential presentations should remain unlinkable.

The problem in one sentence

How can an offline wallet refresh its **non-revocation proof** without revealing **which credential** it holds?

Privacy

Credential presentations should remain unlinkable.

Freshness

Revocation state changes while wallets are offline.

The problem in one sentence

How can an offline wallet refresh its **non-revocation proof** without revealing **which credential** it holds?

Privacy

Credential presentations should remain unlinkable.

Freshness

Revocation state changes while wallets are offline.

Mobile costs

Wallets cannot continually download and process all changes.

Who talks to whom? Revocable credentials



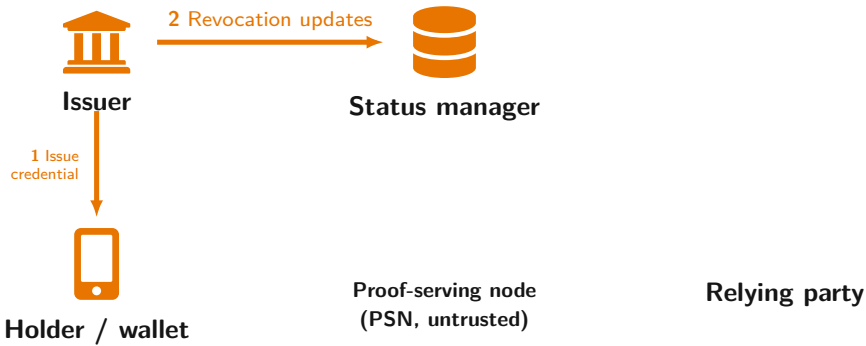
Status manager

Proof-serving node (PSN, untrusted)

Relying party

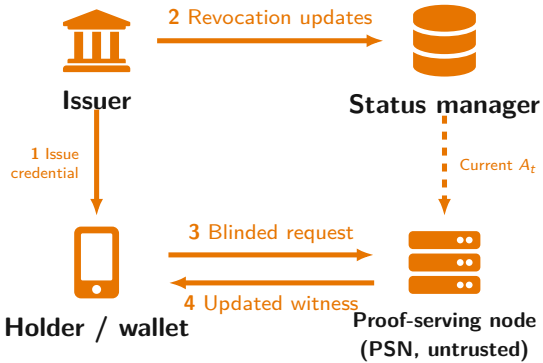
First: the issuer gives the holder a signed credential and witness.

Who talks to whom? Revocable credentials



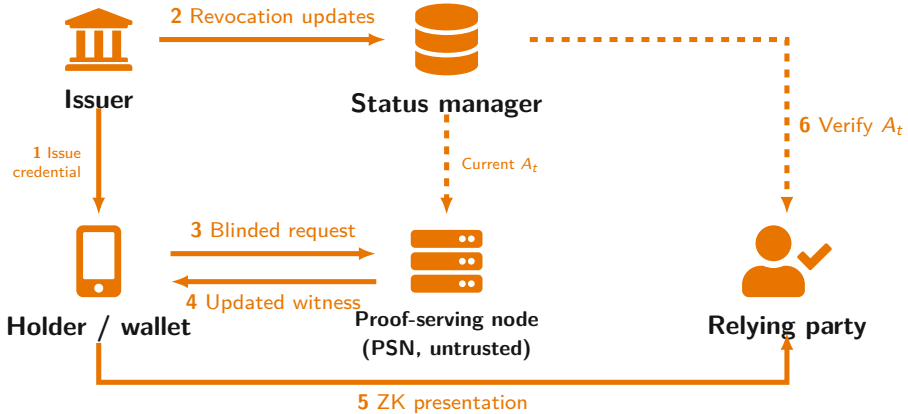
Then: the public revocation state changes while the wallet is offline.

Who talks to whom? Revocable credentials



Next: the PSN refreshes a blinded witness without learning the identifier.

Who talks to whom? Revocable credentials



Finally: the holder proves validity and non-revocation to a relying party.

Additional applications

Registration-based encryption

Set: registered users' public-key identifiers.

Changes: users register keys; registrations may expire or be withdrawn.

Proof: **membership** — the target identity/key is registered.

Additional applications

Registration-based encryption

Set: registered users' public-key identifiers.

Changes: users register keys; registrations may expire or be withdrawn.

Proof: **membership** — the target identity/key is registered.

Stateless blockchains

Set: unspent transaction outputs (UTXOs), or alternatively spent serial numbers.

Changes: transactions create coins and spend existing ones.

Proof: **membership** in the UTXO set, or **non-membership** in the spent set.

Additional applications

Registration-based encryption

Set: registered users' public-key identifiers.

Changes: users register keys; registrations may expire or be withdrawn.

Proof: **membership** — the target identity/key is registered.

Stateless blockchains

Set: unspent transaction outputs (UTXOs), or alternatively spent serial numbers.

Changes: transactions create coins and spend existing ones.

Proof: **membership** in the UTXO set, or **non-membership** in the spent set.

The same witness-update problem appears whenever these authenticated sets evolve.

Revocation needs fresh witnesses

Valid-handle accumulator

Accumulate non-revoked handles V_t.

Wallet proves $x \in V_t$.

Revocation $\Rightarrow$ deletion.

Flamini et al., ePrint 2026/1824, Sec. 5.1: 19,178 issued / day; 548 revoked / day; approx. 20 MB certificate revocation list (CRL). Revocations are batched by configurable epochs; these figures do not specify update frequency.

Background: Camenisch–Lysyanskaya (CRYPTO 2002); Li–Li–Xue (ACNS 2007).

Revocation needs fresh witnesses

Valid-handle accumulator

Accumulate non-revoked handles V_t.

Wallet proves $x \in V_t$.

Revocation $\Rightarrow$ deletion.

Revoked-handle accumulator

Accumulate revoked handles R_t.

Wallet proves $x \notin R_t$.

Revocation $\Rightarrow$ append-only addition.

Flamini et al., ePrint 2026/1824, Sec. 5.1: 19,178 issued / day; 548 revoked / day; approx. 20 MB certificate revocation list (CRL). Revocations are batched by configurable epochs; these figures do not specify update frequency.

Background: Camenisch–Lysyanskaya (CRYPTO 2002); Li–Li–Xue (ACNS 2007).

Revocation needs fresh witnesses

Valid-handle accumulator	Revoked-handle accumulator
Accumulate non-revoked handles V_t.	Accumulate revoked handles R_t.
Wallet proves $x \in V_t$.	Wallet proves $x \notin R_t$.
Revocation $\Rightarrow$ deletion.	Revocation $\Rightarrow$ append-only addition.

We study both membership and non-membership delegation.

Flamini et al., ePrint 2026/1824, Sec. 5.1: 19,178 issued / day; 548 revoked / day; approx. 20 MB certificate revocation list (CRL). Revocations are batched by configurable epochs; these figures do not specify update frequency.

Background: Camenisch–Lysyanskaya (CRYPTO 2002); Li–Li–Xue (ACNS 2007).

Revocation needs fresh witnesses

Valid-handle accumulator	Revoked-handle accumulator
Accumulate non-revoked handles V_t.	Accumulate revoked handles R_t.
Wallet proves $x \in V_t$.	Wallet proves $x \notin R_t$.
Revocation $\Rightarrow$ deletion.	Revocation $\Rightarrow$ append-only addition.

We study both membership and non-membership delegation.

548 revocations / day in Italy's CIE eID data

Flamini et al., ePrint 2026/1824, Sec. 5.1: 19,178 issued / day; 548 revoked / day; approx. 20 MB certificate revocation list (CRL). Revocations are batched by configurable epochs; these figures do not specify update frequency.

Background: Camenisch–Lysyanskaya (CRYPTO 2002); Li–Li–Xue (ACNS 2007).

Why not update locally?

Offline for weeks $\longrightarrow$ many updates k

Why not update locally?

Offline for weeks $\longrightarrow$ many updates k



Why not update locally?

Offline for weeks $\longrightarrow$ many updates k



Traditional local update

Receive/process data scaling with k; high bandwidth and battery demand.

Why not update locally?

Offline for weeks $\longrightarrow$ many updates k



Traditional local update

Receive/process data scaling with k; high bandwidth and battery demand.

Desired private delegation

Send a **constant-size** blinded request; perform **constant** online client work in k.

Why not update locally?

Offline for weeks $\longrightarrow$ many updates k



Traditional local update

Receive/process data scaling with k; high bandwidth and battery demand.

Desired private delegation

Send a **constant-size** blinded request; perform **constant** online client work in k.

But the PSN must not learn x.

Our contributions

1 **Private delegation syntax.** Formalize the blind, update, verify, and unblind interface, and its correctness, privacy, and soundness goals.

Our contributions

1 **Private delegation syntax.** Formalize the blind, update, verify, and unblind interface, and its correctness, privacy, and soundness goals.

2 **Protocols for RSA and bilinear accumulators.** Private delegation of both (non-)membership proof updates, with distinct privacy guarantees.

Our contributions

1. **Private delegation syntax.** Formalize the blind, update, verify, and unblind interface, and its correctness, privacy, and soundness goals.

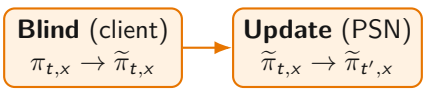
2. **Protocols for RSA and bilinear accumulators.** Private delegation of both (non-)membership proof updates, with distinct privacy guarantees.

3. **Implementation and performance evaluation.** Open-source Rust prototypes and benchmarks comparing client and server costs across RSA, class-group, and bilinear instantiations.

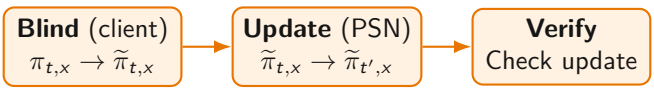
Our interface: blind, update, verify, unblind

Blind (client)
$\pi_{t,x} \to \widetilde{\pi}_{t,x}$

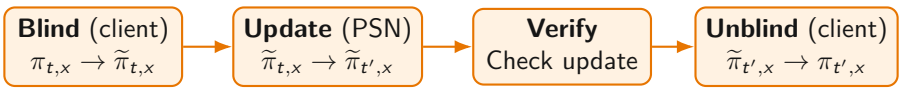
Our interface: blind, update, verify, unblind



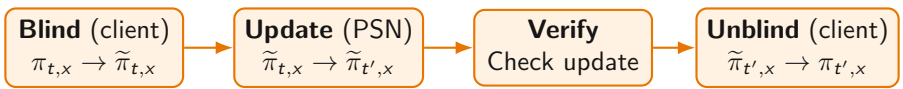
Our interface: blind, update, verify, unblind



Our interface: blind, update, verify, unblind

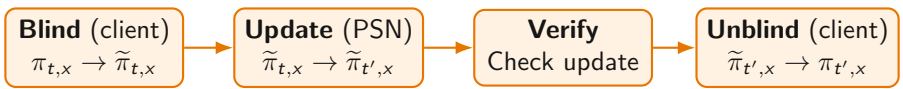


Our interface: blind, update, verify, unblind



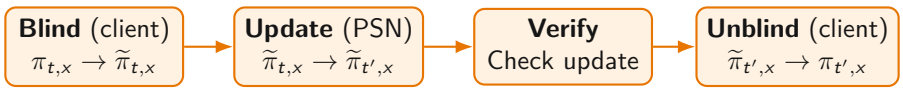
- **Correctness:** refreshed witness verifies against $A_{t'}$.

Our interface: blind, update, verify, unblind



- **Correctness:** refreshed witness verifies against $A_{t'}$.
- **Privacy:** blinded request hides the target element within the stated threat model.

Our interface: blind, update, verify, unblind



- **Correctness:** refreshed witness verifies against $A_{t'}$.
- **Privacy:** blinded request hides the target element within the stated threat model.
- **Soundness:** malicious PSN cannot persuade the client to accept a bad update.

RSA accumulators: membership proof creation

Let S_t contain prime representatives, $P_t = \prod_{s \in S_t} s$, and $A_t = g^{P_t} \bmod N$.

MemProofCreate(A_t, S_t, x)

1. Require $x \in S_t$.

RSA accumulators: membership proof creation

Let S_t contain prime representatives, $P_t = \prod_{s \in S_t} s$, and $A_t = g^{P_t} \bmod N$.

MemProofCreate(A_t, S_t, x)

1. Require $x \in S_t$.
2. Compute $W_x \leftarrow g^{\prod_{s \in S_t \setminus \{x\}} s} \pmod N$.

RSA accumulators: membership proof creation

Let S_t contain prime representatives, $P_t = \prod_{s \in S_t} s$, and $A_t = g^{P_t} \bmod N$.

MemProofCreate(A_t, S_t, x)

1. Require $x \in S_t$.
2. Compute $W_x \leftarrow g^{\prod_{s \in S_t \setminus \{x\}} s} \pmod N$.
3. **return** W_x.

RSA accumulators: membership proof creation

Let S_t contain prime representatives, $P_t = \prod_{s \in S_t} s$, and $A_t = g^{P_t} \bmod N$.

MemProofCreate(A_t, S_t, x)

1. Require $x \in S_t$.
2. Compute $W_x \leftarrow g^{\prod_{s \in S_t \setminus \{x\}} s} \pmod N$.
3. **return** W_x.

MemVer(A_t, W_x, x)

$$W_x^x \stackrel{?}{=} A_t \pmod N$$

RSA membership updates: elementary algebra

Add k new elements to S_t, that is $S_{t'} := S_t \cup \{s_i\}_{i=1}^{k}$. Let $\Delta = \prod_{i=1}^{k} s_i$. Then

$$A_{t'} = A_t^{\Delta}, \qquad W_{t',x} = W_{t,x}^{\Delta}.$$

RSA membership updates: elementary algebra

Add k new elements to S_t, that is $S_{t'} := S_t \cup \{s_i\}_{i=1}^{k}$. Let $\Delta = \prod_{i=1}^{k} s_i$. Then

$$A_{t'} = A_t^{\Delta}, \qquad W_{t',x} = W_{t,x}^{\Delta}.$$

Obstacle

Sending the unblinded witness to the PSN may reveal x: it can test candidate identifiers.

RSA membership updates: elementary algebra

Add k new elements to S_t, that is $S_{t'} := S_t \cup \{s_i\}_{i=1}^{k}$. Let $\Delta = \prod_{i=1}^{k} s_i$. Then

$$A_{t'} = A_t^{\Delta}, \qquad W_{t',x} = W_{t,x}^{\Delta}.$$

Obstacle

Sending the unblinded witness to the PSN may reveal x: it can test candidate identifiers.

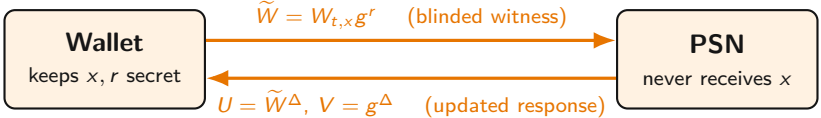
Can the PSN exponentiate a *blinded* witness instead?

RSA membership: private delegated update



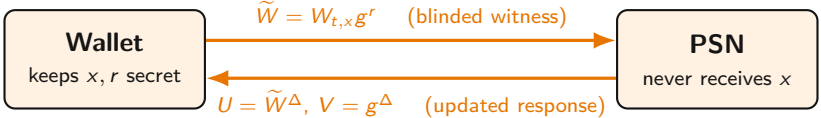
1. Wallet samples r and sends $\widetilde{W} = W_{t,x} g^r$.

RSA membership: private delegated update



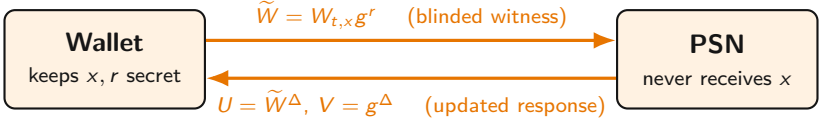
1. Wallet samples r and sends $\widetilde{W} = W_{t,x} g^r$.
2. PSN computes $U = \widetilde{W}^{\Delta}$ and $V = g^{\Delta}$.

RSA membership: private delegated update



1. Wallet samples r and sends $\widetilde{W} = W_{t,x} g^r$.
2. PSN computes $U = \widetilde{W}^{\Delta}$ and $V = g^{\Delta}$.
3. PSN supplies consistency proofs (DLEq) relating $(A_t, A_{t'})$, $(\widetilde{W}, U)$, and (g, V).

RSA membership: private delegated update



1. Wallet samples r and sends $\widetilde{W} = W_{t,x} g^r$.
2. PSN computes $U = \widetilde{W}^{\Delta}$ and $V = g^{\Delta}$.
3. PSN supplies consistency proofs (DLEq) relating $(A_t, A_{t'})$, $(\widetilde{W}, U)$, and (g, V).
4. Wallet recovers $W_{t',x} = \dfrac{U}{V^r} = W_{t,x}^{\Delta}$.

What about revocation via a blacklist?

$$R_t = \{\text{revoked credential handles}\}$$

What about revocation via a blacklist?

$$R_t = \{\text{revoked credential handles}\}$$

- The wallet proves its hidden x is **not in** R_t.

What about revocation via a blacklist?

$$R_t = \{\text{revoked credential handles}\}$$

- The wallet proves its hidden x is **not in** R_t.
- The authority **only appends** newly revoked handles.

What about revocation via a blacklist?

$$R_t = \{\text{revoked credential handles}\}$$

- The wallet proves its hidden x is **not in** R_t.
- The authority **only appends** newly revoked handles.
- Its RSA non-membership witness has the form (a, B), where

$$A_t^a B^x = g.$$

What about revocation via a blacklist?

$$R_t = \{\text{revoked credential handles}\}$$

- The wallet proves its hidden x is **not in** R_t.
- The authority **only appends** newly revoked handles.
- Its RSA non-membership witness has the form (a, B), where

$$A_t^a B^x = g.$$

- We want a fresh witness without giving the PSN x.

RSA accumulators: non-membership proof creation

Let $P_t = \prod_{s \in S_t} s$ and $A_t = g^{P_t} \bmod N$.

NonMemProofCreate(A_t, S_t, x)

1. Require $x \notin S_t$ and $\gcd(P_t, x) = 1$.

RSA accumulators: non-membership proof creation

Let $P_t = \prod_{s \in S_t} s$ and $A_t = g^{P_t} \bmod N$.

NonMemProofCreate(A_t, S_t, x)

1. Require $x \notin S_t$ and $\gcd(P_t, x) = 1$.
2. Compute $(a, b) \leftarrow \text{Bezout}(P_t, x)$, so $aP_t + bx = 1$.

RSA accumulators: non-membership proof creation

Let $P_t = \prod_{s \in S_t} s$ and $A_t = g^{P_t} \bmod N$.

NonMemProofCreate(A_t, S_t, x)

1. Require $x \notin S_t$ and $\gcd(P_t, x) = 1$.
2. Compute $(a, b) \leftarrow \text{Bezout}(P_t, x)$, so $aP_t + bx = 1$.
3. Compute $B \leftarrow g^b \pmod N$.

RSA accumulators: non-membership proof creation

Let $P_t = \prod_{s \in S_t} s$ and $A_t = g^{P_t} \bmod N$.

NonMemProofCreate(A_t, S_t, x)

1. Require $x \notin S_t$ and $\gcd(P_t, x) = 1$.
2. Compute $(a, b) \leftarrow \text{Bezout}(P_t, x)$, so $aP_t + bx = 1$.
3. Compute $B \leftarrow g^b \pmod N$.
4. **return** $\pi_{t,x} = (a, B)$.

RSA accumulators: non-membership proof creation

Let $P_t = \prod_{s \in S_t} s$ and $A_t = g^{P_t} \bmod N$.

NonMemProofCreate(A_t, S_t, x)

1. Require $x \notin S_t$ and $\gcd(P_t, x) = 1$.
2. Compute $(a, b) \leftarrow \text{Bezout}(P_t, x)$, so $aP_t + bx = 1$.
3. Compute $B \leftarrow g^b \pmod N$.
4. **return** $\pi_{t,x} = (a, B)$.

NonMemVer$(A_t, (a, B), x)$

$$\boxed{A_t^a B^x \stackrel{?}{=} g \pmod N}$$

RSA non-membership: a one-shot construction

1. Wallet chooses a fresh random prime q; sends $y = xq$.

RSA non-membership: a one-shot construction

1. Wallet chooses a fresh random prime q; sends $y = xq$.
2. PSN computes Bézout coefficients against current $P_{t'}$:

$$aP_{t'} + by = 1.$$

RSA non-membership: a one-shot construction

1. Wallet chooses a fresh random prime q; sends $y = xq$.
2. PSN computes Bézout coefficients against current $P_{t'}$:

$$aP_{t'} + by = 1.$$

3. PSN returns $(a, B = g^b)$; wallet outputs (a, B^q).

RSA non-membership: a one-shot construction

1. Wallet chooses a fresh random prime q; sends $y = xq$.
2. PSN computes Bézout coefficients against current $P_{t'}$:

$$aP_{t'} + by = 1.$$

3. PSN returns $(a, B = g^b)$; wallet outputs (a, B^q).

Correctness

$A_{t'}^a (B^q)^x = g^{aP_{t'} + bqx} = g.$

RSA non-membership: a one-shot construction

1. Wallet chooses a fresh random prime q; sends $y = xq$.
2. PSN computes Bézout coefficients against current $P_{t'}$:

$$aP_{t'} + by = 1.$$

3. PSN returns $(a, B = g^b)$; wallet outputs (a, B^q).

Correctness

$A_{t'}^a (B^q)^x = g^{aP_{t'} + bqx} = g.$

Privacy is weaker than for membership delegation: high-entropy secret identifiers, one-shot query, suitable factor sizes.

RSA non-membership delegation: verify and unblind

VerBlindNonMemProofUpd$(A_{t'}, y, (a, B))$

1. Parse (a, B) from the PSN's response; let $y = xq$ be the blinded request.
2. **return** 1 if $\boxed{A_{t'}^{a} B^{y} = g \pmod{N}}$; otherwise 0.

RSA non-membership delegation: verify and unblind

VerBlindNonMemProofUpd$(A_{t'}, y, (a, B))$

1. Parse (a, B) from the PSN's response; let $y = xq$ be the blinded request.
2. **return** 1 if $\boxed{A_{t'}^{a} B^{y} = g \pmod{N}}$; otherwise 0.

UnBlindNonMemProof$(A_{t'}, q, (a, B))$

1. Recover the local secret blinding prime q.
2. Compute $B' \leftarrow B^{q} \pmod{N}$.
3. **return** $\pi_{t',x} \leftarrow (a, B')$.

RSA non-membership delegation: verify and unblind

VerBlindNonMemProofUpd$(A_{t'}, y, (a, B))$

1. Parse (a, B) from the PSN's response; let $y = xq$ be the blinded request.
2. **return** 1 if $\boxed{A_{t'}^{a} B^{y} = g \pmod{N}}$; otherwise 0.

UnBlindNonMemProof$(A_{t'}, q, (a, B))$

1. Recover the local secret blinding prime q.
2. Compute $B' \leftarrow B^{q} \pmod{N}$.
3. **return** $\pi_{t',x} \leftarrow (a, B')$.

A publicly checkable response, followed by a single unblinding exponentiation.

Weaker privacy guarantees for non-membership delegation

What is supported?

One-shot RSA non-membership delegation under the paper's weak privacy setting; identifiers must remain hard to identify from the server's auxiliary information.

Weaker privacy guarantees for non-membership delegation

What is supported?

One-shot RSA non-membership delegation under the paper's weak privacy setting; identifiers must remain hard to identify from the server's auxiliary information.

What is *not* supported?

Two requests xq_1, xq_2 expose x through $\gcd(xq_1, xq_2) = x$. Public disclosure of x also permits retrospective testing.

Weaker privacy guarantees for non-membership delegation

What is supported?

One-shot RSA non-membership delegation under the paper's weak privacy setting; identifiers must remain hard to identify from the server's auxiliary information.

What is *not* supported?

Two requests xq_1, xq_2 expose x through $\gcd(xq_1, xq_2) = x$. Public disclosure of x also permits retrospective testing.

This is a privacy limitation, it'd be interesting to improve on this!

Weaker privacy guarantees for non-membership delegation

What is supported?

One-shot RSA non-membership delegation under the paper's weak privacy setting; identifiers must remain hard to identify from the server's auxiliary information.

What is *not* supported?

Two requests xq_1, xq_2 expose x through $\gcd(xq_1, xq_2) = x$. Public disclosure of x also permits retrospective testing.

This is a privacy limitation, it'd be interesting to improve on this!

Takeaway: Membership delegation and non-membership delegation have different privacy guarantees.

What did we gain?

For holders

$O(1)$ online computation and communication in the number of missed updates (RSA protocols).

Less synchronization burden for intermittently connected wallets.

What did we gain?

For holders

$O(1)$ online computation and communication in the number of missed updates (RSA protocols).

Less synchronization burden for intermittently connected wallets.

For deployments

Outsource update-dependent work to an untrusted PSN.

Membership protocol provides publicly checkable consistency; non-membership has a direct final-witness check.

What did we gain?

For holders

$O(1)$ online computation and communication in the number of missed updates (RSA protocols).

Less synchronization burden for intermittently connected wallets.

For deployments

Outsource update-dependent work to an untrusted PSN.

Membership protocol provides publicly checkable consistency; non-membership has a direct final-witness check.

Useful especially when updates accumulate during long offline periods.

Implementation at a glance

- Implemented in **Rust**; open source on GitHub.
- Main libraries: **Arkworks** and the **Zengo class group** library.
- Benchmarks use **criterion-rs**; each experiment repeated **50** times.
- Parameters: $\lambda = 128$, **3072-bit RSA**, **BLS12-381**, and **6784-bit** class-group discriminants.
- Hardware: consumer laptop with **Apple M2** processor and 16 GB RAM.



GitHub repository

Goal: constant-time online clients; server bears the update-dependent work.

Client performance evaluation

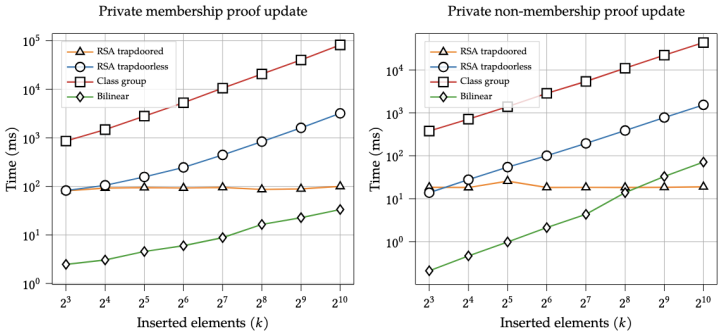
Constant-time client algorithms: runtime and variance

	RSA		Class group		Bilinear	
Algorithm	Avg.	Var.	Avg.	Var.	Avg.	Var.
BlindMemProof	18.82	4.60	56.68	15.37	**0.94**	0.00
UnBlindMemProof	10.71	0.00	46.58	0.00	**0.21**	0.00
BlindNonMemProof	13.15	0.60	8.03	0.69	**0.75**	0.00
UnBlindNonMemProof	6.25	0.00	40.82	0.01	**0.32**	0.00
VerBlindMemProofUpd	90.95	0.01	513.39	2.65	**8.23**	0.00
VerBlindNonMemProofUpd	20.29	0.01	161.65	0.30	**8.31**	0.00

Interpretation: the wallet stays light. Every measured client operation finishes in milliseconds; even verification stays well below one second.

Source: Soóki-Tóth et al., Table 4. Avg. and variance are reported in ms; 50 benchmark repetitions on Apple M2.

Server performance evaluation



PSN update runtime versus inserted elements k

Server performs the update-dependent work. Runtime depends on k and the instantiation; online client cost is independent of k.

Source: Soóki-Tóth et al., performance evaluation, Fig. 9.

Open problems

1. **Reusable RSA non-membership privacy:** avoid GCD leakage across repeated updates while retaining cheap clients.

Open problems

1. **Reusable RSA non-membership privacy:** avoid GCD leakage across repeated updates while retaining cheap clients.

2. **Real wallet evaluation:** benchmark latency, bandwidth, and power against local updates on constrained devices.

Open problems

1. **Reusable RSA non-membership privacy:** avoid GCD leakage across repeated updates while retaining cheap clients.
2. **Real wallet evaluation:** benchmark latency, bandwidth, and power against local updates on constrained devices.
3. **Beyond RSA:** optimize bilinear accumulator delegation and batch use cases.

Open problems

1. **Reusable RSA non-membership privacy:** avoid GCD leakage across repeated updates while retaining cheap clients.

2. **Real wallet evaluation:** benchmark latency, bandwidth, and power against local updates on constrained devices.

3. **Beyond RSA:** optimize bilinear accumulator delegation and batch use cases.

4. **Post-quantum security:** Design private delegation protocols for post-quantum secure accumulators, e.g., the lattice-based accumulator of Kemmoe, Lysyanskaya, and Nguyen [PKC'26].

Open problems

1. **Reusable RSA non-membership privacy:** avoid GCD leakage across repeated updates while retaining cheap clients.

2. **Real wallet evaluation:** benchmark latency, bandwidth, and power against local updates on constrained devices.

3. **Beyond RSA:** optimize bilinear accumulator delegation and batch use cases.

4. **Post-quantum security:** Design private delegation protocols for post-quantum secure accumulators, e.g., the lattice-based accumulator of Kemmoe, Lysyanskaya, and Nguyen [PKC'26].

5. **Multi-server setting** The revocation set might be shared among many different servers and entities.

Thank you!

Questions?

Private Delegation of (Non-)Membership Proof Updates

Applications to Revocable Anonymous Credentials

István András **Seres** · Princeton University

ePrint

GitHub

Selected references I

- Soóki-Tóth, Glasz, Kavousi, Seres. *Private Delegation of (Non-)Membership Proof Updates in Cryptographic Accumulators*. IACR ePrint 2026/832.
- Benaloh and de Mare. *One-Way Accumulators: A Decentralized Alternative to Digital Signatures*. EUROCRYPT 1993.
- Camenisch and Lysyanskaya. *Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials*. CRYPTO 2002.
- Li, Li, and Xue. *Universal Accumulators with Efficient Nonmembership Proofs*. ACNS 2007.

Selected references II

- Camenisch, Kohlweiss, and Soriente. *An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials*. PKC 2009.
- Papamanthou, Tamassia, and Triandopoulos. *Optimal Verification of Operations on Dynamic Sets*. CRYPTO 2011.
- Christ and Bonneau. *Limits on Revocable Proof Systems, with Implications for Stateless Blockchains*. FC 2023.
- Bonneau, Chen, Christ, Karantaidou. *Merkle Mountain Ranges Are Optimal: On Witness Update Frequency for Cryptographic Accumulators*. CRYPTO 2025.
- Abusalah et al. *Lower Bounding Update Frequency in Short Accumulators and Vector Commitments*. ePrint 2025.
- All references and proofs: `https://eprint.iacr.org/2026/832`.

References: EUDI deployment context

- Andrea Flamini, Anja Lehmann, Giada Sciarretta, Mario Scuro, Nicola Smaniotto, Alessandro Tomasi, Silvio Ranise. *Comparing Privacy-Preserving Revocation for the EUDI Wallet*. IACR ePrint 2026/1824.
- **Empirical motivation (Sec. 5.1):** Italian CIE data: 19,178 daily issuances; 548 daily revocations; approx. 440,000 revoked certificates in a 20 MB CRL.
- **Architectural distinction (Sec. 2):** revocations are enacted at epoch boundaries. The number of revocations per epoch depends on the deployment's epoch length.

Backup: RSA accumulator notation

$$P_t = \prod_{s \in S_t} s, \quad A_t = g^{P_t} \pmod{N}.$$

Membership

$$W_x^x = A_t \pmod{N}.$$

Non-membership

$$aP_t + bx = 1 \implies A_t^a (g^b)^x = g \pmod{N}.$$

Backup: Repeated queries are linkable

$$y_1 = xq_1, \quad y_2 = xq_2$$

$$\boxed{\gcd(y_1, y_2) = x}$$

- Figure 5 deliberately targets a weaker, one-shot privacy guarantee.
- A known candidate x permits retrospective testing: $\gcd(x, y) \neq 1$.
- Applicability depends on identifier secrecy and the delegation workflow.

Backup: Private delegation at a glance

RSA membership

$\widetilde{W} = Wg^r$

$U = \widetilde{W}^{\Delta}, \quad V = g^{\Delta}$

$W' = U/V^r$

RSA non-membership

$y = xq$

$aP' + by = 1, \; B = g^b$

$\pi' = (a, B^q)$

The privacy guarantees differ between these two protocols.