Radboud University

Privacy Friendly Revocation of Credentials

Thomas Luijkman
Maximilian Pohl
Wouter Doeland

May 16, 2024

Motivating example

- Imagine you are running a hotel that uses smart cards for everything
- To make sure that people can only access the parts of the hotel that they are supposed to, the smart card stores access rights
- At first, you decide that all these credentials can be tied to a single identifier
  - **Is this safe?**
  - **Why not? Can you come up with some examples of security/privacy risks using this approach?**

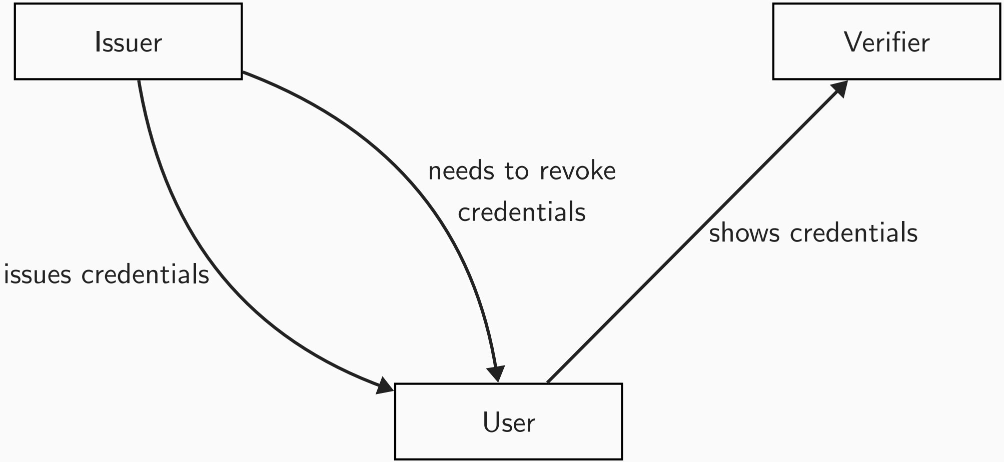
Motivating example

- We do not *have* to keep these access rights tied to an identifier.
- What if we don't? **Is this safe?**
  - **No!** Even without identifiers, the usage of these access rights could still be traced.
- So, we need even more anonymisation. How? **Fully anonymise the access rights!**
  - Don't show the access right, but just some mathematical proof that you indeed have it.
  - Different usages of the access right can no longer be traced.
- Is this without problems? **Still no!**

Motivating example

- We have fixed the **privacy** issues. But now **security** is in jeopardy! What if...
    - A key card gets stolen?
    - A guest accidentally takes their key card with them?
    - The hotel needs to remove a guest from the hotel earlier?
    - A guest loses their privilege to only one hotel amenity?
    - Every guest loses access to the same amenity?
    - The list goes on...
- Long story short: we need a way to **revoke** the access rights of users!
    - But how? These access rights were fully anonymized!
    - This lecture will discuss ways to solve this problem.

Generalizing the problem



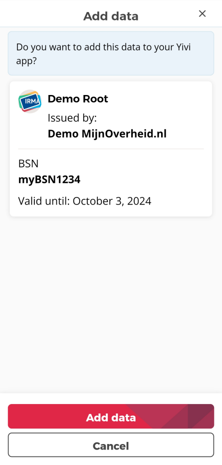
Problem statement

- So, we have **users**, **issuers** and **verifiers**.
- Instead of access rights we have **anonymous credentials**.
  - When using the credential, do not show the credential but prove that you have it using zero-knowledge proofs
  - Important property: **unlinkability**
- How do we **revoke** these anonymous credentials?
  - Important property: **unavoidability**
  - Note: this is different from revocable privacy!
- Multiple factors to be considered when designing revocation schemes

Yivi Demo

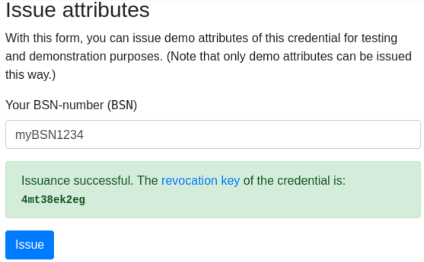
Yivi Demo — Enrollment

(1) Enrollment QR code

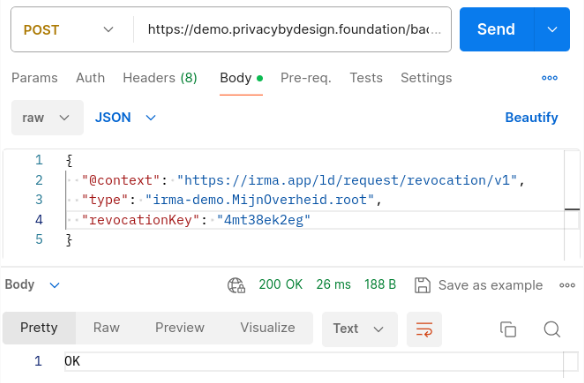


(2) Yivi App

Yivi Demo — Revocation



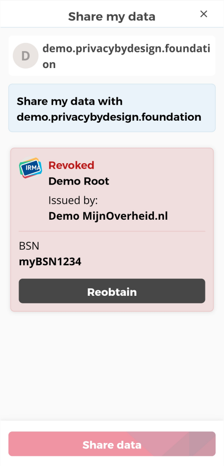
(3) Revocation value from issuer



(4) Revocation request

Yivi Demo — Login attempt

(5) Login QR code



(6) Yivi App

Idemix

Idemix — Recap

Public key

$$A \leftarrow \left(\frac{Z}{S^{v} \cdot R_k^{sk} \cdot R_r^{x} \cdot \prod_{i=1}^{L} R_i^{m_i}} \right)^{\frac{1}{e} \mod \varphi(n)} \mod n$$

Public key

safe RSA modulus

Users secret key

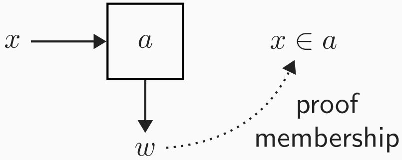
Signed attributes

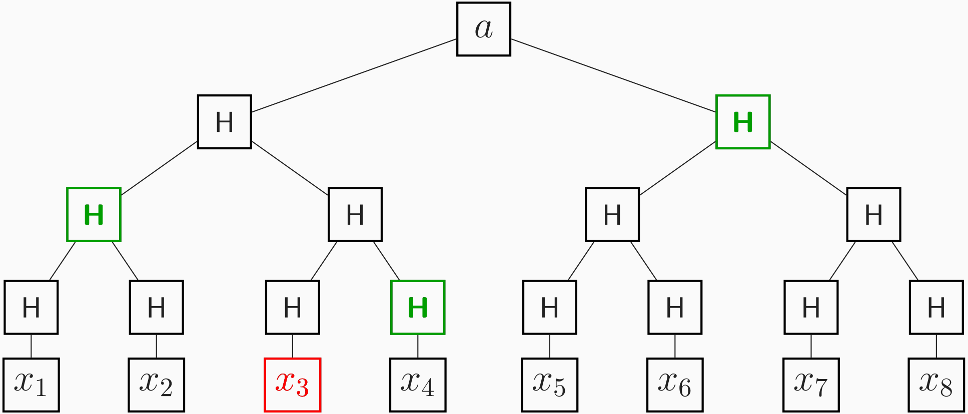
Revocation value

$\Rightarrow (A, e, v)$ are the signature over the message $(sk, x, m_1, \ldots, m_L)$

Accumulators

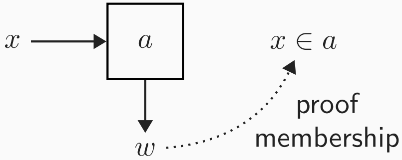
- Accumulates values to a **fixed size**
  - $\Rightarrow$ easy **proof of membership**
- There exist different variants
  - static
  - additive
  - subtractive
  - dynamic
- while each of those can be either
  - positive
  - negative
  - universal





$\Rightarrow$ H $\hat{=}\ w$

- Accumulates transactions to a **fixed size**
  - $\Rightarrow$ easy **proof of membership**
- There exist different variants
  - static
  - additive
  - subtractive
  - dynamic
- while each of those can be either
  - positive
  - negative
  - universal



- RSA-B was first introduced by Camenisch and Lysyanskaya in 2002
- More formalized by Baldimtsi et al. in 2017
- Private key of the issuer: $sk : (p, q)$ with p, q safe prime
  $\Rightarrow$ $p = 2p' + 1$ and $q = 2q' + 1$ with p', q' prime
- Public key: $pk : n = pq$
- The domain D are all odd, positive prime integers x
- Operations take place in $QR_n = ((\mathbb{Z}/n\mathbb{Z})^*)^2$, i.e., the group of quadratic residues within the multiplicative integers modulo n

What is a safe prime?

What operations does a dynamic accumulator need to support?

- **Addition** of attribute x to accumulator a:

$$w = a^{x^{-1} \mod p'q'} \mod n$$

- **Deletion** of a attribute y from the accumulator a:

$$a_{t+1} = a_t^{y^{-1} \mod p'q'} \mod n$$

- **Update witness** w to new accumulator

$$bx + cy = 1$$

$$w_{t+1} = w_t^c a^b \mod n$$

- **Verify membership** of attribute x in accumulator a

$$a \stackrel{?}{=} w^x \mod n$$

Zero knowledge Proof!

1. Issuer wants to revoke a credential
2. Delete attribute y from accumulator a_t
3. Distribute a_{t+1} and y to all users and verifiers
4. Users have to update their witness w
5. Verifiers should only accept most recent accumulator a_{latest}

Accumulators — Conclusion

+ Credentials can't be linked even after revocation
+ Proving and verification are $O(1)$

– User and verifiers must receive updates for every **revocation**
– Users have to update their witness for every **revocation**
  $\Rightarrow$ Doing an extended Euclidean algorithm

Verifier-Local Revocation and improvements proposed by Lueks et al.

Verifier-Local Revocation (VLR)[4]

- Introduced in 2003 by Ateniese, Song, and Tsudik
- Goal: provide efficient revocation of credentials without communicating to the end-user machine
- Add **Revocation List** (RL) to signature verification algorithm.
  - Contains a token for each revoked user
  - Only signatures of unrevoked users are accepted

\+ **Preserves privacy of unrevoked users**

\+ **Works on smart cards**

– **Reveals signatures of revoked users**

– **Revocation check for the verifier not efficient**

- Issuer: Issue Revocable Credential
  - Pick $r \xleftarrow{\$} \mathbb{Z}_q$
  - Issue C(r) to user
- User: Prove Possession of Revocation Token
  - Choose random $g \in G$
  - Show (g, g^r)
- Verifier: Verify Revocation Token
  - Loop over $r_i \in$ *RL*: if $g^{r_i} = g^r$: Fail!

VLR Problems

- Revocation check scales linearly with $|RL|$
- Privacy Weakness: Signature Reveal of Revoked Users

1. User Bob (with revocation token r) shows credentials and proves possession of their revocation token in σ
2. Verifier checks with current revocation list RL: **Verify**$(RL, \sigma) = \text{OK}$
3. Verifier does not know who signed σ as this is hidden by the algorithm. The verifier stores σ.
4. Repeat steps 1-3 a couple of times.
5. Later, Bobs revocation token r is added to the list of revoked credentials RL'.
6. Now the verifier checks again with updated RL': **Verify**$(RL', \sigma) = \text{NOT OK}$
7. **The verifier can now link the actions performed by Bob together!**

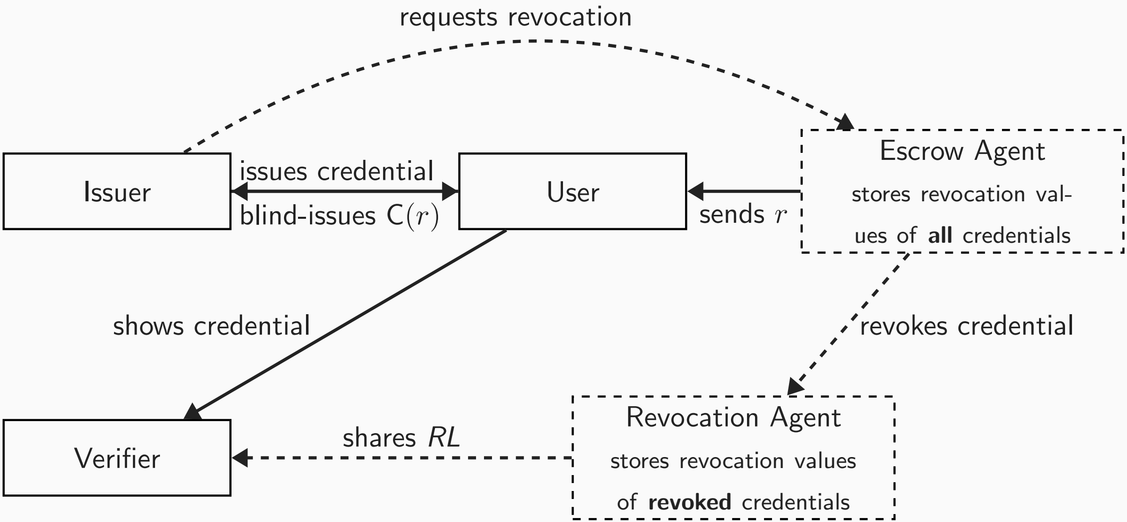
Lueks et al.[5] — Solution for the weaknesses of VLR

- Build upon Verifier Local Revocation
- However, in the signature don't take random generator g, instead:
  1. Split time into epochs
  2. Compute for epoch ϵ and verifier V: $g_{\epsilon,V} = H(\epsilon||V)$
- Don't share r as revocation value to the verifier directly, instead:
  1. Set up a revocation agent
  2. Revocation agent computes for epoch ϵ and verifier V: $g_{\epsilon,V} = H(\epsilon||V)$
  3. Create revocation list for revoked tokens $r_i, \dots, r_j$: $RL = \{g_{\epsilon,V}^{r_i}, \dots, g_{\epsilon,V}^{r_j}\}$
  4. Share *RL* with verifier V for epoch ϵ
  5. On verifying, the verifier checks if the computed revocation value is in RL

Lueks et al. — Handling of revocation tokens

- To avoid giving the issuer too much power:
  1. Set up a trusted Escrow Agent *EA*
  2. *EA* generates revocation tokens and maps these to IDs: (ID_i, r_i)
  3. Using blind-issuing, allow users to obtain a credential from the issuer containing this revocation token r, while the issuer never sees r
  4. The issuer stores ID_i for revocation
- The issuer can revoke a token:
  1. Send a request to *EA* with token ID_i
  2. *EA* will revoke the token r_i by sharing it with the Revocation Agent
  3. The Revocation Agent updates its revocation list: $RL' = \{\ldots, g_{\epsilon,V}^{r_i}\}$

Lueks et al. — Overview



Comparison of schemes

	Accumulators	VLR	Solution by Lueks et al.
User can be offline	No	Yes	Yes
Proving complexity	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Verifying complexity	$\mathcal{O}(1)$	$\mathcal{O}(\|RL\|)$	$\mathcal{O}(1)$
Security	$+$	$+$	$+$
Privacy	$+$	$+$/-	$+$

Conclusion

Today:

- Problem of privacy-friendly revocation of credentials
- Revocation flow in Yivi: Accumulators
- Verifier Local Revocation
- Improved version by Lueks et al.
- Comparison of revocation schemes

Future reading:

- Lapon et al. “Analysis of Revocation Strategies for Anonymous Idemix Credentials” [6]
- Lueks et al. “Fast revocation of attribute-based credentials for both users and verifiers” [5]
- IRMA Docs Revocation: `https://irma.app/docs/revocation/`

[1] Ralph C. Merkle. **“Protocols for Public Key Cryptosystems”.** In: *1980 IEEE Symposium on Security and Privacy*. ISSN: 1540-7993. Apr. 1980, pp. 122–122. DOI: 10.1109/SP.1980.10006.

[2] Jan Camenisch and Anna Lysyanskaya. **“Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials”.** In: *Advances in Cryptology — CRYPTO 2002*. Ed. by Gerhard Goos et al. Vol. 2442. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 61–76. ISBN: 978-3-540-45708-4. DOI: 10.1007/3-540-45708-9_5.

[3] Foteini Baldimtsi et al. **"Accumulators with Applications to Anonymity-Preserving Revocation".** In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. Paris: IEEE, Apr. 2017, pp. 301–315. ISBN: 978-1-5090-5762-7. DOI: 10.1109/EuroSP.2017.13.

[4] Giuseppe Ateniese, Dawn Song, and Gene Tsudik. **"Quasi-Efficient Revocation of Group Signatures".** In: *Financial Cryptography*. Ed. by Gerhard Goos et al. Vol. 2357. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 183–197. ISBN: 978-3-540-36504-4. DOI: 10.1007/3-540-36504-4_14.

[5] Wouter Lueks et al. **"Fast revocation of attribute-based credentials for both users and verifiers".** In: *Computers & Security* 67 (June 2017), pp. 308–323. ISSN: 01674048. DOI: 10.1016/j.cose.2016.11.018.

[6] Jorn Lapon et al. **"Analysis of Revocation Strategies for Anonymous Idemix Credentials".** In: *Communications and Multimedia Security*. Ed. by Bart De Decker et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 3–17. ISBN: 978-3-642-24712-5. DOI: 10.1007/978-3-642-24712-5_1.