Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance

Taha Adeel Mohammed

Indian Institute of Technology Hyderabad

April 27, 2024

Reference

Title

Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance

Authors

- Jack Doerner - Technion, Israel
- Yashvanth Kondi - Aarhus University, Denmark
- Eysa Lee, Abhi Shelat, LaKyah Tyner - Northeastern University, USA

Publication

- 2023 IEEE Symposium on Security and Privacy (SP)

Contents

1. Preliminaries
   - Anonymous Credentials
   - Threshold Signatures
2. BBS+ Signature Scheme
   - Key Generation
   - Signing
   - Verification
3. Thresholdizing the BBS+ Protocol
   - High Level Overview
   - Key Generation
   - Signing
4. Weak Partially-Blind Signing
5. Applications
6. Implementation and Benchmarks

Preliminaries

Anonymous Credentials

Anonymous Credentials

- Allow users to prove credentials without revealing their own identity.
- Basic security properties:
  - **Unlinkability:** The user's actions are not linked.
  - **Unforgeability:** Non-issuers cannot create valid credentials.
- Signature Scheme + ZKPoK of it satisfying some predicate.
- **Example:** A user can prove they are 18+ without revealing their age.

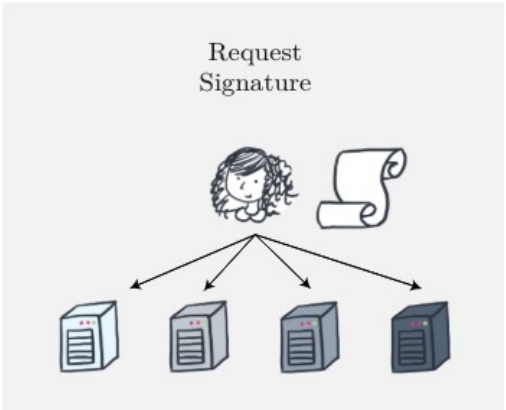
Blind Signatures

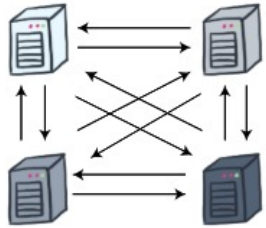
- **Weakly blind:** User message/identity is hidden from issuer too.
- **Strongly blind:** Generated signature is also hidden.

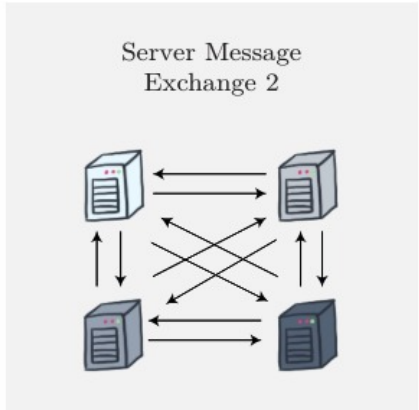
Threshold Signatures

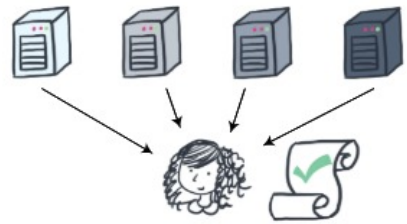
Motivation

- Credential issuer is a single point of failure.
- Anonymous credential forgery can be catastrophic.
- **Solution**: Distribute signing power across multiple servers.









Threshold Signatures

Issuer and its signing function replaced by an ideal functionality that computes same signing function when computed by the multiple servers.

Security Properties

Threshold signing protocol (with threshold t) with security against malicious adversaries *under composition*[1]:

- Same security properties as single honest issuer even if $t - 1$ servers are compromised.
- No properties of credential need to be reproven if signing protocol is composible.

[1] Universal Composability (UC) framework - R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS), 2001

BBS+ Signature Scheme

BBS+ Signature Scheme

- Anonymous signature scheme using bilinear pairings.
- Allows signing multiple messages (ℓ) at once.
- Size of signature independent of number of messages signed.
- Efficient ZKPoK that reveals selective attributes.
- Based on the q-Strong Diffie-Hellman (qSDH) assumption.

Key Generation

BBS+Gen($\mathcal{G}, \ell$)

1. $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, q)$
2. $H = \{h_0, h_1, \ldots, h_\ell\}$, where $h_i \xleftarrow{\$} \mathbb{G}_1$
3. $x \xleftarrow{\$} \mathbb{Z}_q^*$, $X = x \cdot g_2$
4. $sk = (H, x)$, $pk = (H, X)$
5. Return (sk, pk)

Signing

BBS+Sign($sk, m \in \mathbb{Z}_q^\ell$)

1. $sk = (H, x)$
2. Nonces: $r \xleftarrow{\$} \mathbb{Z}_q$, $s \xleftarrow{\$} \mathbb{Z}_q$
3. Compute:

$$A = \frac{g_1 + s \cdot h_0 + \sum_{i=1}^{\ell} m_i \cdot h_i}{\boxed{x + r}}$$

4. Return $\sigma = (A, r, s)$

Verification

BBS+Ver(pk, σ, $m \in \mathbb{Z}_q^\ell$)

1. $pk = (H, X)$, $\sigma = (A, r, s)$
2. Check:

$$e(A, X + r \cdot g_2) = e(g_1 + s \cdot h_0 + \sum_{i=1}^{\ell} m_i \cdot h_i, g_2)$$

3. Output 1 if and only if equality holds.

Thresholdizing the BBS+ Protocol

Threshold BBS+ Protocol

High Level Overview

- Main difficulty is that the signing involves computing the inverse of secret value $(x + r)$.
- Tackled using Bar-Ilan and Beaver secure inversion technique.[2]
- Three phases:
  - **Key Gen:** Initial setup and distribution of secret shares of x.
  - **Signing:** Client chooses t servers and gets signature share $A_i = (R_i, u_i)$ from each, which it combines and verifies to get final signature.
  - **Verification:** Same as original BBS+ verification.

[2] Bar-Ilan, O., Beaver, D.: Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pp. 201-209. ACM (1988)

Key Generation

1. Share of the secret key x_i for party $\mathcal{P}_i$:
   - $\hat{x}_i(\cdot) \xleftarrow{\$}$ $(t-1)$ degree polynomials.
   - Send $\hat{x}_i(j)$ to every party $\mathcal{P}_j$, $j \neq i$.
   - $x_i = \sum_{j=1}^{n} \hat{x}_i(j)$.
2. Public key share $X_i = x_i \cdot g_2$.
   - Commit-and-release and ZKPoK of discrete log as correctness checks.
3. Public key $X = x \cdot g_2$: found by taking any size t subsets of X_is.
4. H: Commit-and-release of random $\mathbb{G}_1$ elements.
5. Output $pk = (H, X)$

Signing

Client $\mathcal{C}$ approaches t servers to sign on message m.

1. Using standard secret sharing, the t signing parties agree on random values c, r, s, with each party having share c_i, r_i, s_i.
2. The signing parties perform secret sharing of $u = c \cdot (x + r)$
3. Each party $\mathcal{P}_i$ can now compute
$$R_i = c_i \cdot (g_1 + s \cdot h_0 + \sum_{i=1}^{\ell} m_i \cdot h_i, g_2)$$
4. Client can now compute the signature using the shares (R_i, u_i) as:
$$A = \frac{\sum_i R_i}{\sum_i u_i} = \frac{c \cdot (g_1 + s \cdot h_0 + \sum_{i=1}^{\ell} m_i \cdot h_i)}{c \cdot (x + r)}$$

Weak Partially-Blind Signing

Weak Partially-Blind Signing

Client does not send m to the signing parties. Instead:

1. $\mathcal{C}$: Masking nonce $= s_0 \xleftarrow{\$} \mathbb{Z}_q$
2. $\mathcal{C}$: Computes and sends (along with ZKPoK of s_0)

$$B' = s_0 \cdot h_0 + \sum_{i=1}^{\ell} m_i \cdot h_i$$

3. $\mathcal{P}_i$: Determines s as usual, and computes

$$B = g_1 + s \cdot h_0 + B'$$

4. $\mathcal{C}$: Computes final signature as $\sigma = (A, r, s + s_0)$

Applications

Applications

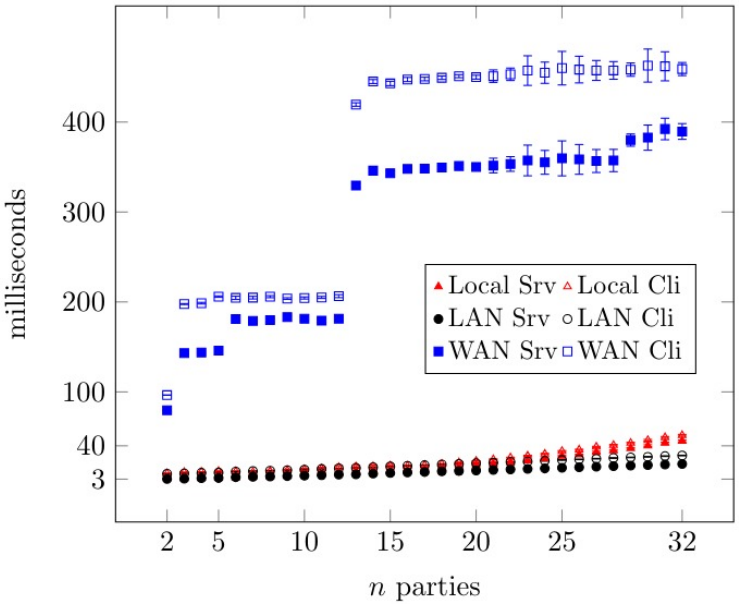
- The weak patially-blind signing protocol can be used to **coalesce multiple credentials** into a single compact credential.
- The thresholdizing technique can be used for other similar signature schemes, such as Verifiable Random Function (VRF) scheme.
- The scheme can be extended to support **strong blind signatures** with some added computational complexity.

Implementation and Benchmarks

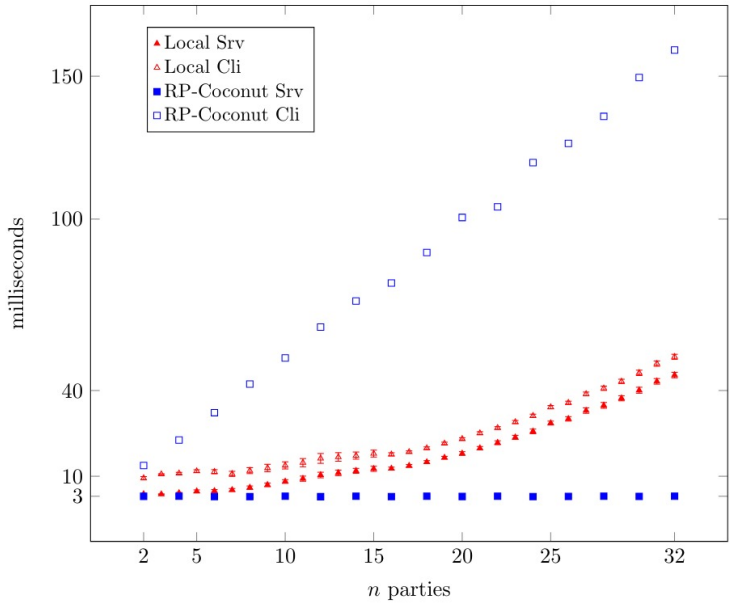
Implementation and Benchmarks

- The authors implemented the protocol in Rust. (6400 LOC!)
- They run the simulations for different server counts - n and different network environments - local, LAN, WAN.
- They also compare their benchmarks against other previous anonymous threshold signatures.

Results



(a) Signing time vs n



(b) BBS+ vs RP-Coconut

Thank you!