THRESHOLD CALL PREVIEW TALKS #3

Octopus-Robot and Octopus-Taptop

Robust Threshold BBS(+) and Two-Party BBS(+)

TEAM MEMBERS

Man Ho Au · Robert H. Deng · Liming Gao · Bowen Jiang · Xianhui Lu · Tian Qiu · **Guofeng Tang** · Ruida Wang · Haiyang Xue · Guomin Yang · Chengru Zhang

Speaker Guofeng Tang, Singapore Management University

Presented on 2026-10-07



TEAM

Octopus

PART I

Octopus-Robot

Robust threshold BBS(+)

ePrint 2026/651

PART II

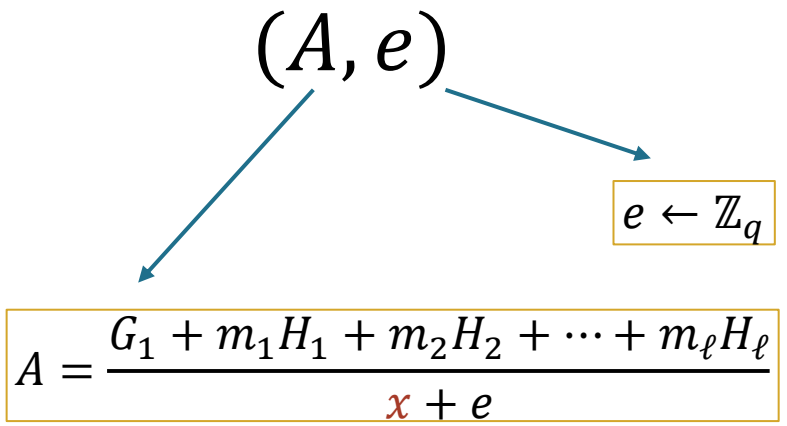
Octopus-Taptop

Two-party BBS(+)

ePrint 2026/573

What BBS Signature is

- Secret key: $x \in \mathbb{Z}_q$
- Public key: $X = xG_2 \in \mathbb{G}_2, G_1, H_1, \cdots, H_\ell \in \mathbb{G}_1$
- Messages: $msg = (m_1, m_2, \cdots, m_\ell)$



Standardization: W3C Verifiable Credentials Group and IETF

What BBS+ Signature is

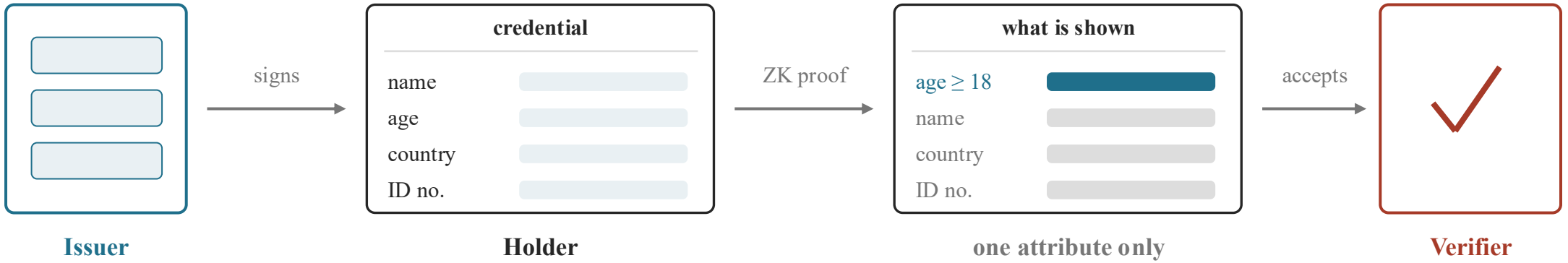
- Secret key: $x \in \mathbb{Z}_q$
- Public key: $X = xG_2 \in \mathbb{G}_2, G_1, H_0, \cdots, H_\ell \in \mathbb{G}_1$
- Messages: $msg = (m_1, m_2, \cdots, m_\ell)$

$$(A, e, s)$$

$$e, s \leftarrow \mathbb{Z}_q$$

$$A = \frac{G_1 + sH_0 + m_1H_1 + m_2H_2 + \cdots + m_\ell H_\ell}{x + e} \in \mathbb{G}_1$$

What BBS(+) is used for

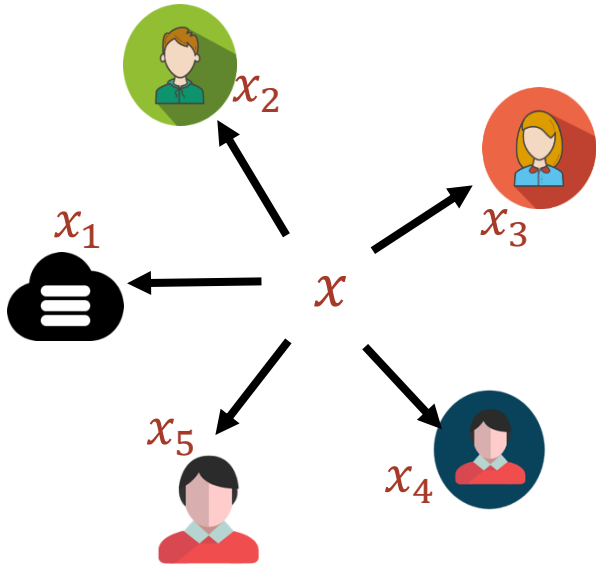


- Supports signing multiple attributes in a single constant-size signature of credential.
- Enables succinct zero-knowledge proofs for selective disclosure.
- A key building block for anonymous credentials and privacy-preserving authentication.

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. (CRYPTO 2004)
[MWY06] Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k -TAA. In: SCN. vol. 4116,pp. 111–125. Springer (2006)
[MWY13] Au, M.H., Susilo, W., Mu, Y., Chow, S.S.M.: Constant-size dynamic k –times anonymous authentication. IEEE Syst. J. 7(2), 249–261 (2013)

Threshold BBS(+)

- A single issuer/signer introduces key compromise, a **single point of failure**. To mitigate these risks, **threshold BBS(+)** is required.
- Example: **2-out-of-5** threshold structure



At least **3** parties

(A, e, s)

$e, s \leftarrow \mathbb{Z}_q$

$$A = \frac{G_1 + sH_0 + m_1H_1 + m_2H_2 + \cdots + m_\ell H_\ell}{x + e}$$

Our Results

Robot: Robust threshold BBS(+)

- General t-out-of-n threshold structure
- Only requires two rounds
- Satisfies robustness when $n \geq 2t + 1$
- Identifiable abort when $n \geq t + 1$

Taptop: Two-party BBS(+)

- Optimize specifically for two parties
  - 5 times less communication
  - 5 times faster
- Only requires two passes

PART I

Octopus-Robot

Robust Threshold BBS(+) in Two Rounds



ePrint 2026/651

Criteria of Threshold Signatures

- **The number of communication rounds**



- **Robustness**: guarantee the success of each signing protocol as long as $t + 1$ honest parties are present

- **Robustness** can
  - Prevent DoS (Denial of Service) Attacks
  - Ensure Valid Output.

More rounds lead to higher network latency

Previous Works of Threshold BBS(+)

sent by each party

Schemes	Rounds	Robust	Communication	Computation
DKLST23	3	No	$O(n)$	$O(n)$
WMC24	4	Yes	$O(1)$	$O(n)$
TX25	3	Yes	$O(n)$	$O(n^2)$

n is the number of signing parties

- DKLST23 is **not robust**, will **abort** if a **single party behaves maliciously**.
- WMC24 is robust with $O(1)$ communication, it needs **4 rounds** of interaction.
- TX25 is robust and three-round, but at the cost of **higher complexities**.

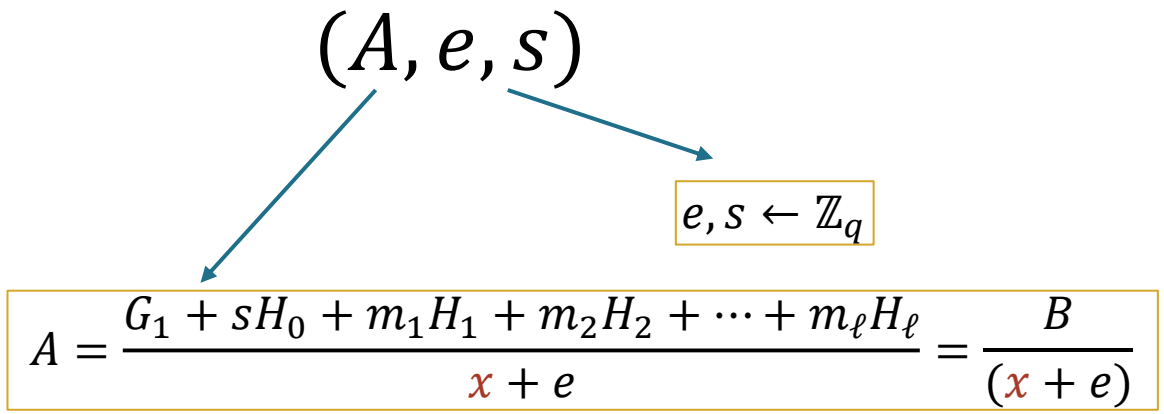
Our Threshold BBS(+)

- **Motivations**: Robustness, Fewer rounds, Lower communication

- **Our Results**

Schemes	Rounds	Robust	Communication	Computation
DKLST23	3	No	$O(n)$	$O(n)$
WMC24	4	Yes	$O(1)$	$O(n)$
TX25	3	Yes	$O(n)$	$O(n^2)$
Ours	**2**	**Yes**	$\boldsymbol{O(1)}$	$\boldsymbol{O(n)}$

The Challenges



For avoiding inversion operation

$$A = \frac{\gamma B}{\gamma(x + e)}$$

- γ, x are both shared secrets
- B, e are public

Task 1. Generate random (e, s) distributedly

Task 2. Generate γB and $\gamma(x + e)$, and get A

Our Techniques

Task 1: Generate (e, s)

- **With Threshold Verifiable Random Functions (TVRF)**
- **Core Idea: TVRF enables one-round robust generation of unbiased (e, s)**
  - Only One Round: Each party sends a share (e_i, s_i), derived by a VRF, along with a zero-knowledge proof that ensures (e_i, s_i) is valid (i.e., unbiased)
  - Threshold Structure: as long as $t + 1$ parties present valid (e_i, s_i), the final (e, s) is valid

Our Techniques

Task 2: Generate γB and $\gamma(x+e)$

- **With Threshold Linearly Homomorphic Encryption (TLHE)**
- **Core idea: TLHE enables 2-round robust generation of γB and $\gamma(x+e)$, where its 1st round can be merged with Task 1**
- Rewrite $\gamma B = \gamma B_1 + s\gamma H_0$: $B_1 = G_1 + m_1 H_1 + m_2 H_2 + \cdots + m_\ell H_\ell$
- Rewrite $\gamma(x+e) = \gamma x + \gamma e$

$EG.Enc(\gamma_i B_1)$, $EG.Enc(\gamma_i H_0)$, $CL.Enc(\gamma_i)$, $\gamma_i \odot CL.Enc(x)$

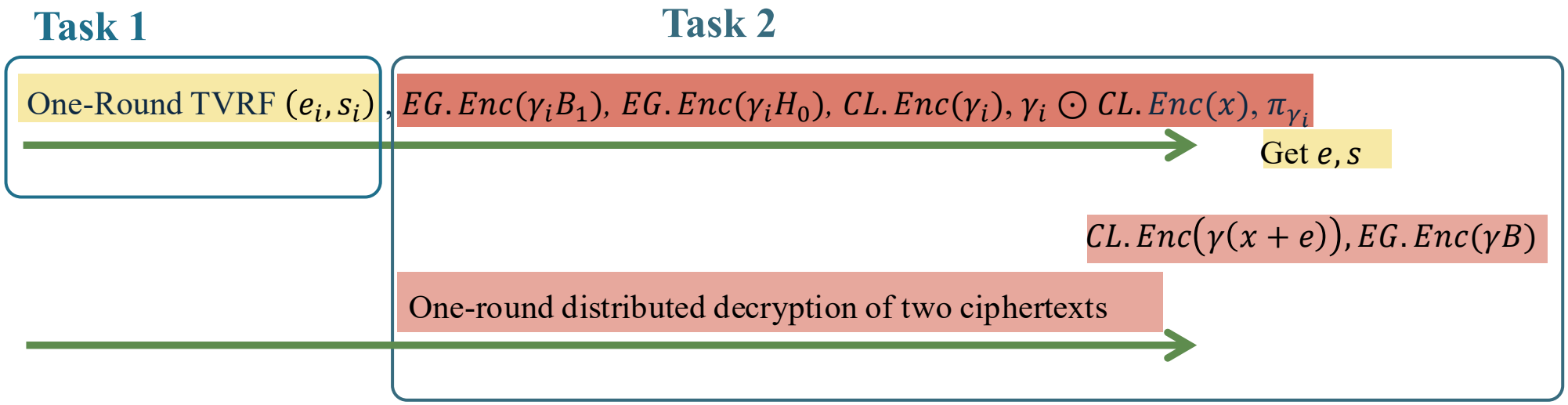
Can be merged with Task 1

Get e, s from Task 1

$CL.Enc(\gamma(x+e)), EG.Enc(\gamma B)$

One-round distributed decryption of two ciphertexts

Our Signing Protocol



π_{γ_i}: γ_i's consistency proof for these ciphertexts

- **Two Rounds:** handle **2 tasks** in two rounds
- **Robustness:** if $t + 1$ parties present correct partial decryption, output valid signature.
- **Constant Communication**

Performance

- Communication (KB)

n^*	Tools	2	5	10	15	20	25	30
DKLST23 [25] non-robust	OT	75.9	302.4	679.9	1057.4	1434.9	1812.4	2189.9
TX25 [47]	CL	**1.56**	4.5	9.4	14.3	19.2	24.1	29
WMC24 [49]	CL	3.23	3.23	3.23	3.23	3.23	3.23	3.23
Ours	CL	2.02	**2.02**	**2.02**	**2.02**	**2.02**	**2.02**	**2.02**

- Computation (ms)

n^*	Tools	2	5	10	15	20	25	30
DKLST23 [25] non-robust	OT	4	9.5	20	29	38	46	55
TX25 [47]	CL	**201**	713	2248	4513	7521	11299	15827
WMC24 [49]	CL	478	976	1769	2573	3405	4232	5079
Ours	CL	260	**503**	**924**	**1346**	**1788**	**2206**	**2626**

PART II

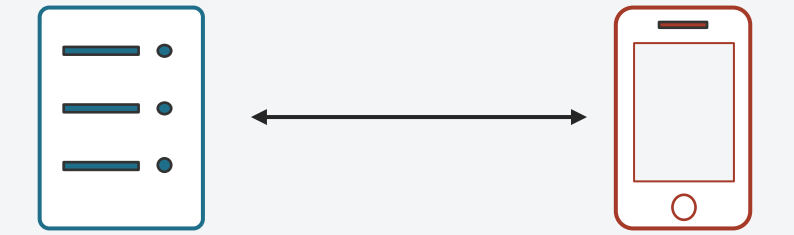
Octopus-Taptop

Two-Party BBS(+) in Two Passes



ePrint 2026/573

Two-Party BBS(+)



$x = x_1 + x_2 \bmod q$
(easy to extend to 2-out-of-n settings)

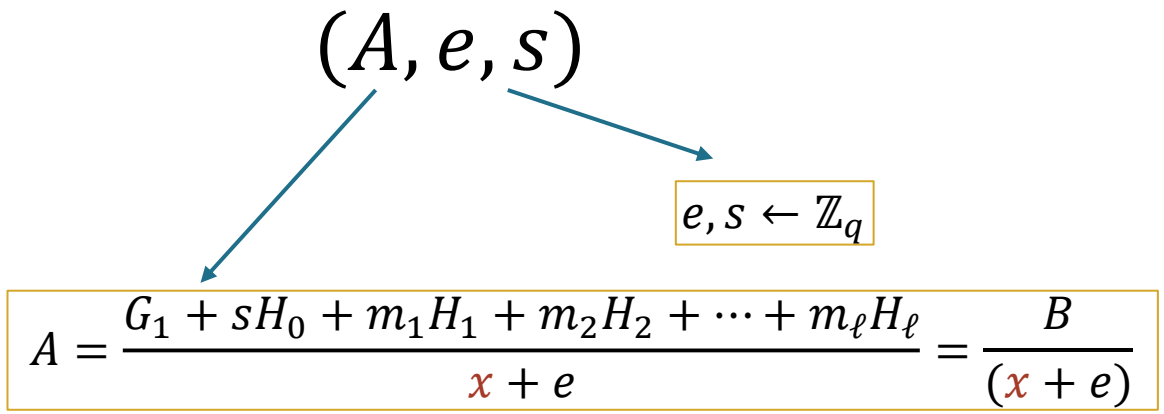
Why we focus on this case

- Two-party setting is a special case and most common in practice.
- NIST required

Our Two-Party BBS(+)

Schemes	Passes	Communication (KB)	Computation (ms)	Tools
DKLST23	3	151.8	8	OT
WMC24	4	6.46	936	CL
TX25	3	3.12	402	CL
Robot (n = 2)	**2**	**4.04**	**520**	**CL**
Taptop	**2**	**0.85**	**108**	**CL**

The Challenges



For avoiding inversion operation

$$A = \frac{\gamma B}{\gamma(x + e)}$$

- γ, x are both shared secrets
- B, e are public

Task 1. Generate random (e, s) distributedly

Task 2. Generate γB and $\gamma(x + e)$, and get A

Our Techniques

Task 1: Two parties generate (e, s)

- **With Verifiable Random Functions (VRF)**
- **Core Idea: Only the second party can cheat when generating (e, s)**

P_1 — the first-moving party

$\beta_1 \leftarrow \mathcal{Y}$

P_2 — the second-moving party

$(\beta_2, \pi_{\text{Vrf}}) \leftarrow \text{VRF-Eval}(sk_2, H_1(m \,||\, sid))$

$(e, s) \leftarrow H(\beta_1 || \beta_2)$ **unbiased, at no additional pass**

Our Techniques

Task 2: Generate γB and $\gamma(x+e)$

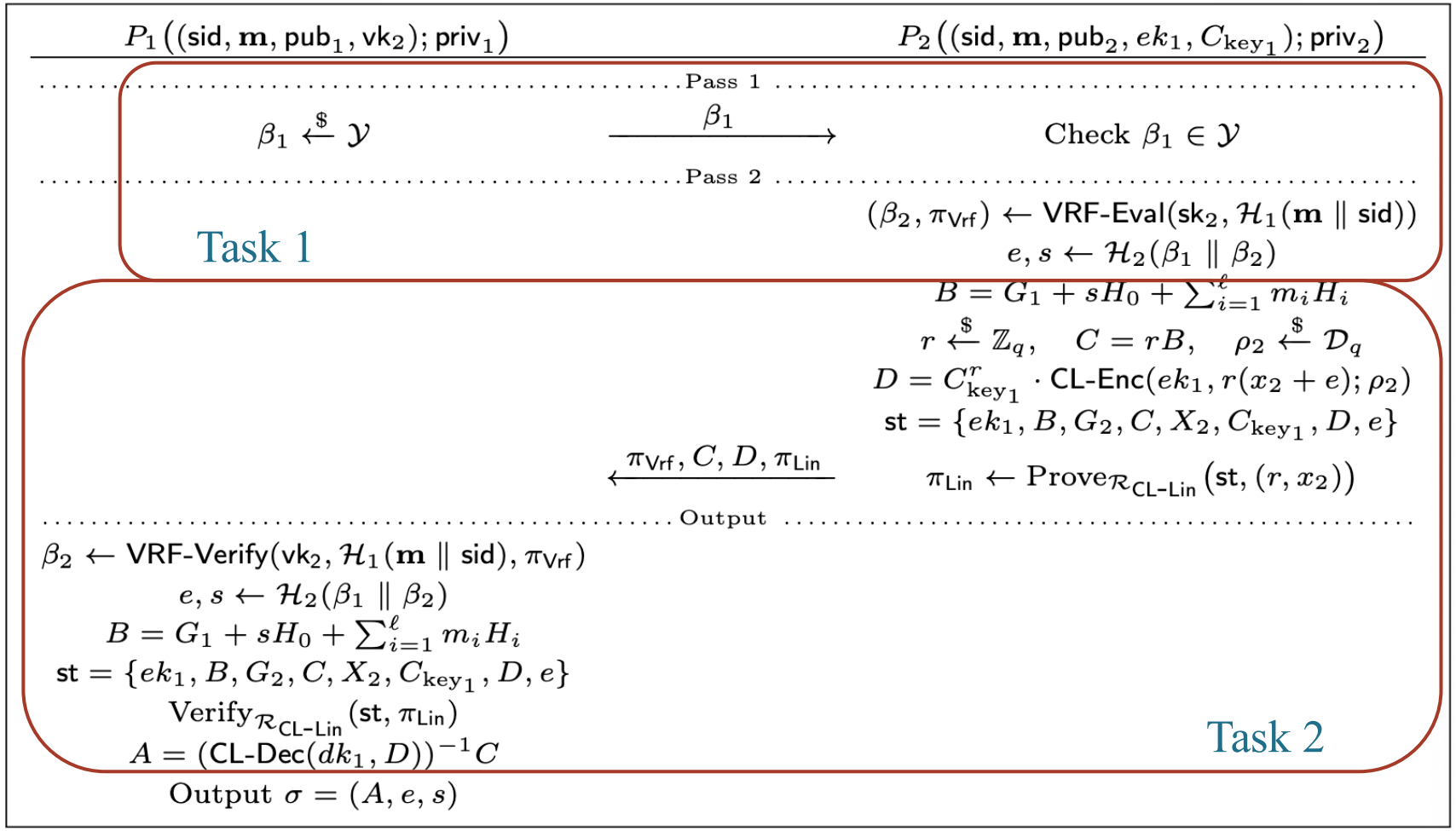
- **With Linearly Homomorphic Encryption (LHE) under P_1's public key**
- **Core idea: The mask γ is temporary, so we let one party determine it**

P_2 get (e, s) from Task 1, pick $\gamma \leftarrow \mathbb{Z}_q$, send $\gamma B, \gamma \odot Enc(x+e)$

P_1 decrypt the ciphertext to obtain $\gamma(x+e)$

P_1 output $A = \gamma B / \gamma(x+e)$

Our Two-Party Signing Protocol



RESULT

Implementations

PoC Implementations

Open source

Octopus-Robot

github.com/Jiangjiang-jiang/Multiparty-BBS-Plus-Implementation

Octopus-Taptop

github.com/Xiaofei-Wu-20/Two-Party_Two-Passes_BBS_plus

NIST Submission

Ongoing

Preparing submission materials for the NIST Multi-Party Threshold Cryptography call.

SUMMARY

Conclusions

Octopus-Robot

Robust threshold BBS(+)

- General t-out-of-n threshold signing in two rounds
- Robustness when $n \geq 2t+1$
- Identifiable abort when $n \geq t+1$
- Constant communication per party
- Need broadcast channel
- ePrint 2026/651

Octopus-Taptop

Two-party BBS(+)

- Signing in only two passes
- 0.85 KB communication and 108 ms computation
- Only point-to-point channel
- ePrint 2026/573

Both PoC implementations are open source, and the NIST submission is in progress.

ONGOING WORK

Toward Adaptive Security

Static security

Prior work and ours

The adversary fixes the set of corrupted parties before the protocol starts.

- DKLST23, WMC24, TX25
- **Octopus-Robot**
- **Octopus-Taptop**

Adaptive security

Under review

The adversary may corrupt parties at any time during execution, based on everything it has seen.

- **Our new adaptively secure construction of threshold BBS(+)**

None of the existing threshold BBS(+) schemes, including the two presented today, is proven secure against adaptive corruptions.

THRESHOLD CALL PREVIEW TALKS #3

Thank you

Questions and comments are welcome

Octopus-Robot
ePrint 2026/651

Octopus-Taptop
ePrint 2026/573



TEAM

Octopus