A deep dive in Threshold BLS Signature Scheme

Sourav Das

Category Labs

Presented at STPPA#8 (2025-Sep-18), NIST Gaithersburg
Special Topics on Privacy and Public Auditability, Event 8

souravdas1547@gmail.com

Outline

- Background on Bilinear pairing
- Boneh-Lynn Shacham Signature Scheme
- Threshold Secret Sharing
- Design of Threshold BLS Signature
- Security of Threshold BLS Signature
- Evaluation Results

Background on Bilinear Pairing

- Finite field $\mathbb{F}$ of prime order q
- Three groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of order q each
- Efficiently computable bilinear map.

$$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

1. Bilinearity:

$$\forall\, a, b \in \mathbb{F}^*,\ (g_1, g_2) \in (\mathbb{G}_1, \mathbb{G}_2), \quad e\left(g_1^a, g_2^b\right) = e(g_1, g_2)^{a \cdot b}$$

2. Non-degenerate:

Given generators $(g_1, g_2) \in (\mathbb{G}_1, \mathbb{G}_2)$, $e(g_1, g_2)$ generates $\mathbb{G}_T$

Boneh-Lynn-Sacham (BLS) Signatures

Bilinear pairing-based signature scheme

Key generation

$$\mathrm{sk} \coloneqq s \leftarrow \mathbb{F}$$
$$\mathrm{pk} \coloneqq g^s \in \mathbb{G}_1$$

Signing

$$\sigma \coloneqq \mathrm{H}(m)^s \in \mathbb{G}_2$$

Verification

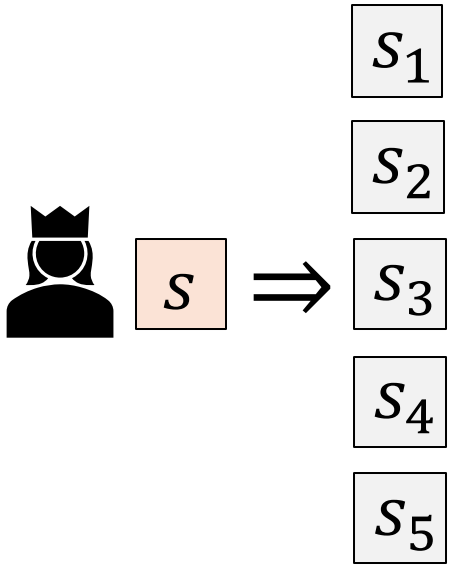
$$e\big(\mathrm{pk}, \mathrm{H}(m)\big) = e(g, \sigma)$$

Correctness:

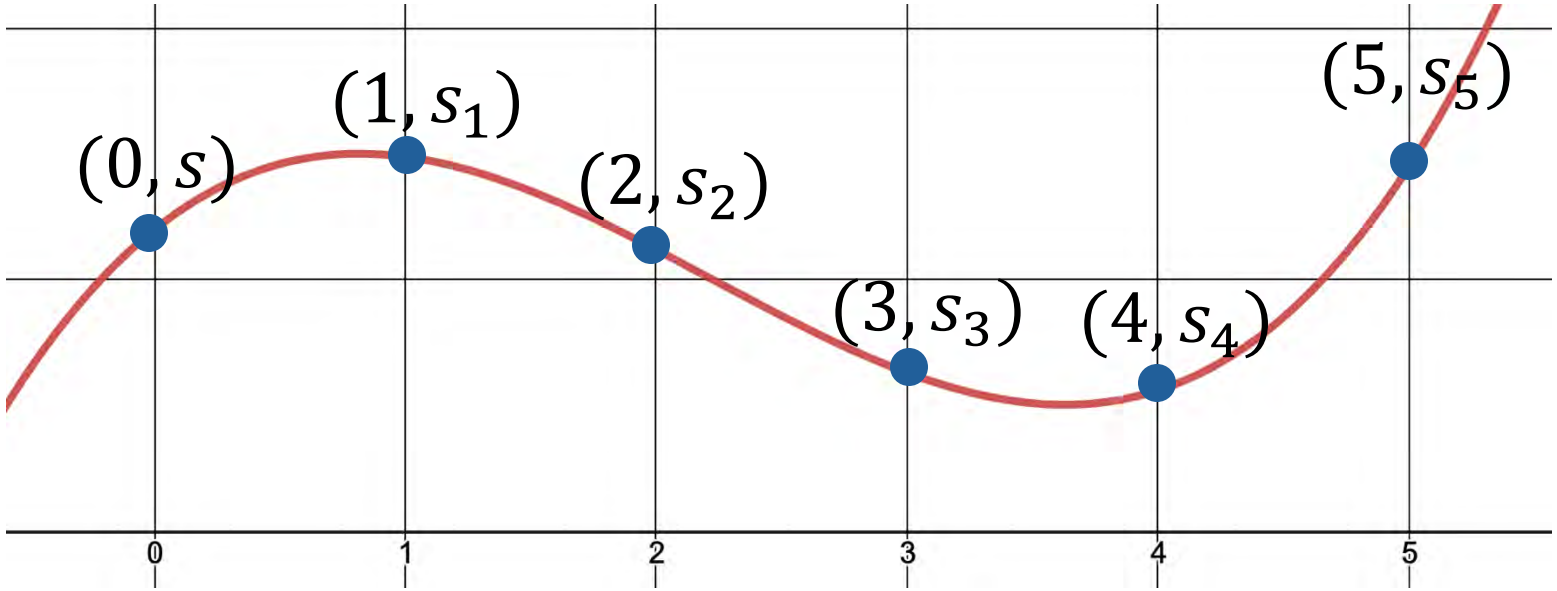
- LHS: $e\big(\mathrm{pk}, \mathrm{H}(m)\big) = e(g^s, \mathrm{H}(m)) = e\big(g, \mathrm{H}(m)\big)^s$
- RHS: $e(g, \sigma) = e(g, \mathrm{H}(m)^s) = e\big(g, \mathrm{H}(m)\big)^s$

Security: Computational Diffie-Hellman (CDH) in the random oracle model

(n, t) Threshold Secret Sharing



- Share a secret s into n shares
- $\geq t + 1$ shares reveals s
- $\leq t$ shares hides s
- Example: Shamir secret sharing



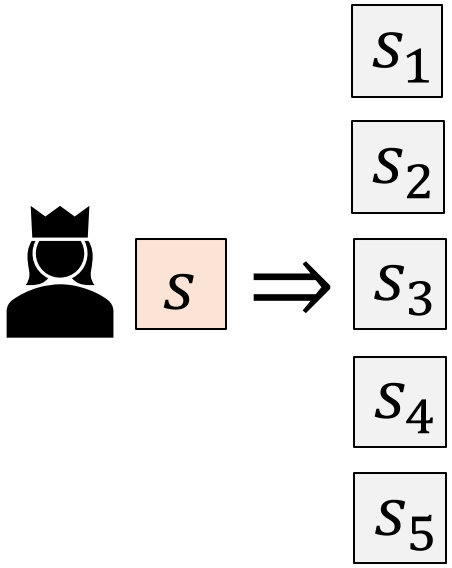
Lagrange Interpolation

$$(a_1, P(a_1)), (a_2, P(a_2)), \dots, (a_t, P(a_t)) \quad \Rightarrow \quad P(0)$$

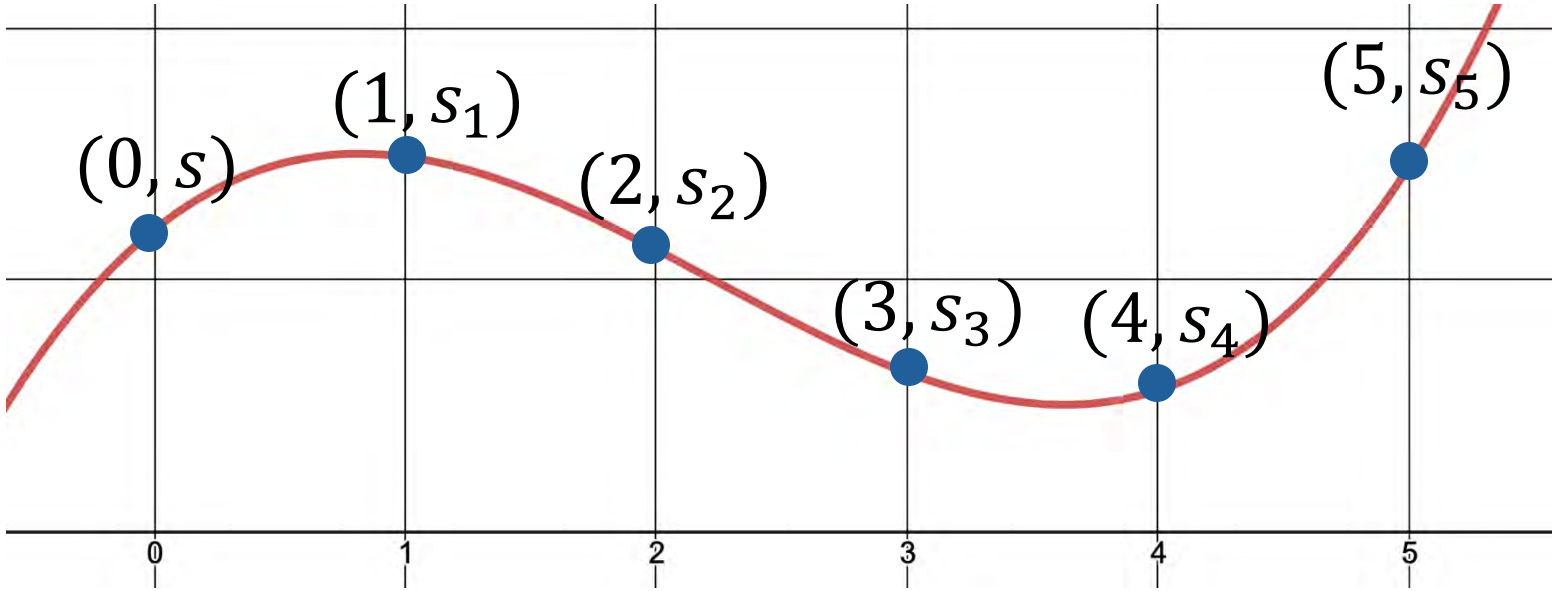
Lagrange polynomials: $\lambda_{a_i}(X) = \prod_{i \neq j} \frac{(X - a_j)}{(a_i - a_j)}$ $\Rightarrow$ $\lambda_{a_i}(a_i) = 1$, $\lambda_{a_i}(a_j) = 0$

$$P(X) = \sum_{\forall\, a_i} P(a_i) \cdot \lambda_{a_i}(X) \quad \Rightarrow \quad P(0) = \sum_{\forall\, a_i} P(a_i) \cdot \lambda_{a_i}(0)$$

(n, t) Threshold Secret Sharing

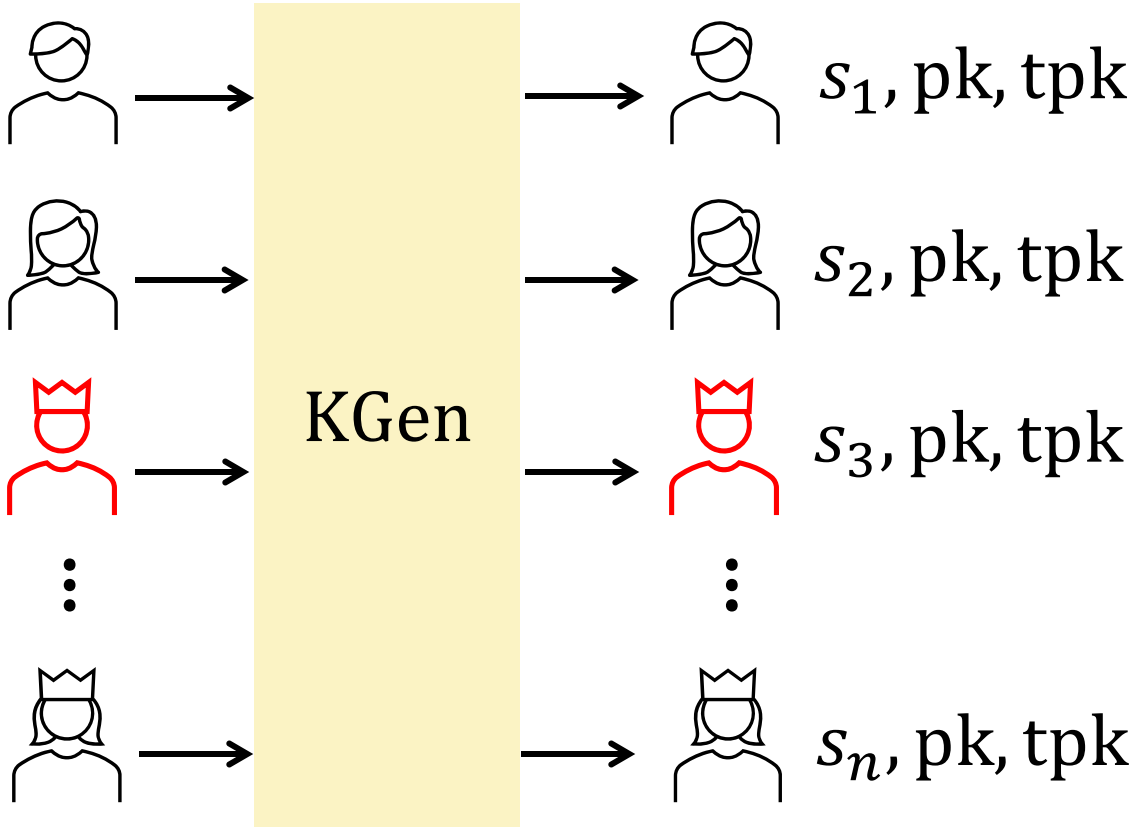


- Share a secret s into n shares
- $\geq t + 1$ shares reveals s
- $\leq t$ shares hides s
- Example: Shamir secret sharing



BLS Threshold signature [Boldyreva’03]

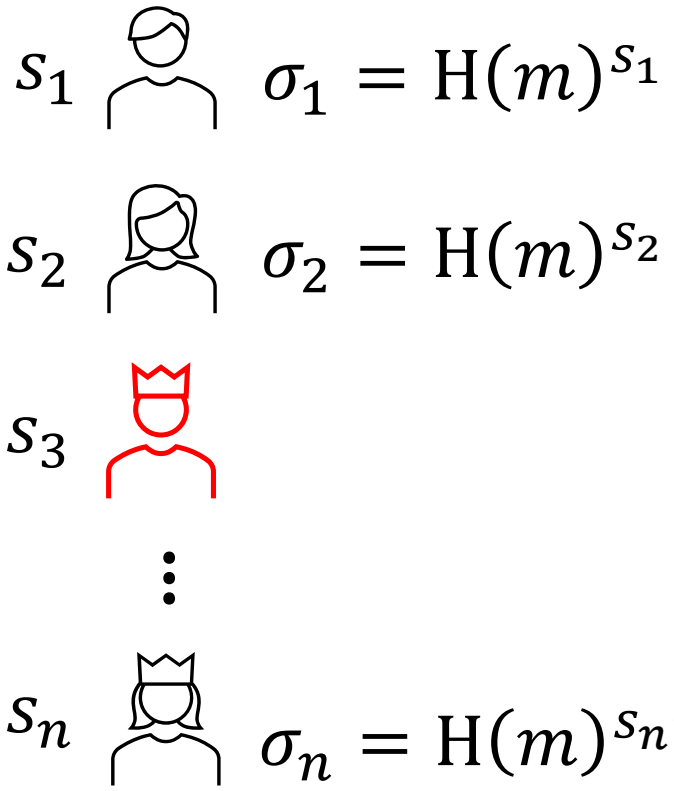
Key Generation



$$\{s_1, \dots, s_n\} \leftarrow \text{Share}(s)$$
$$\text{pk} = g^s$$
$$\text{tpk} = \{g^{s_1}, \dots, g^{s_n}\}$$

Signing

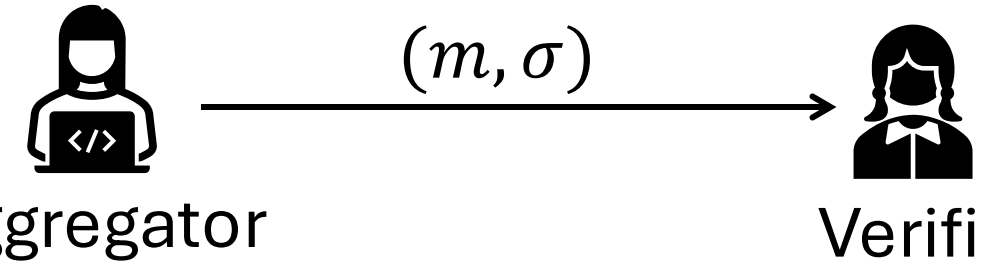
λ_i are the Lagrange coefficients.



$$\mathrm{H}(m)^s = \sigma = \prod_{i \in T} \sigma_i^{\lambda_i}$$

$$e(\mathrm{pk}, \mathrm{H}(m)) = e(g, \sigma)$$

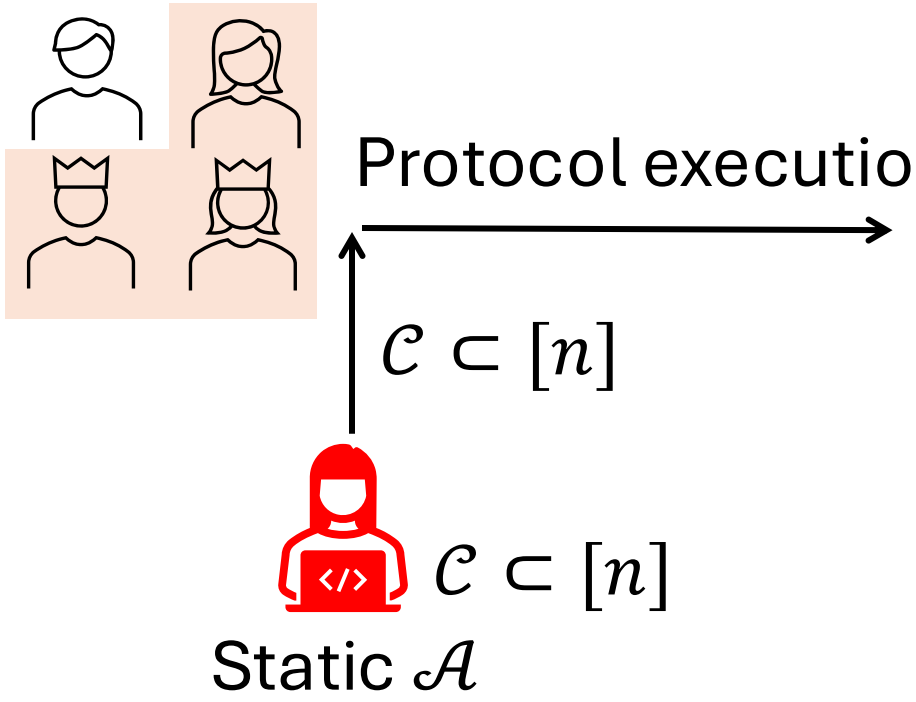
$$\mathrm{pk} = g^s$$



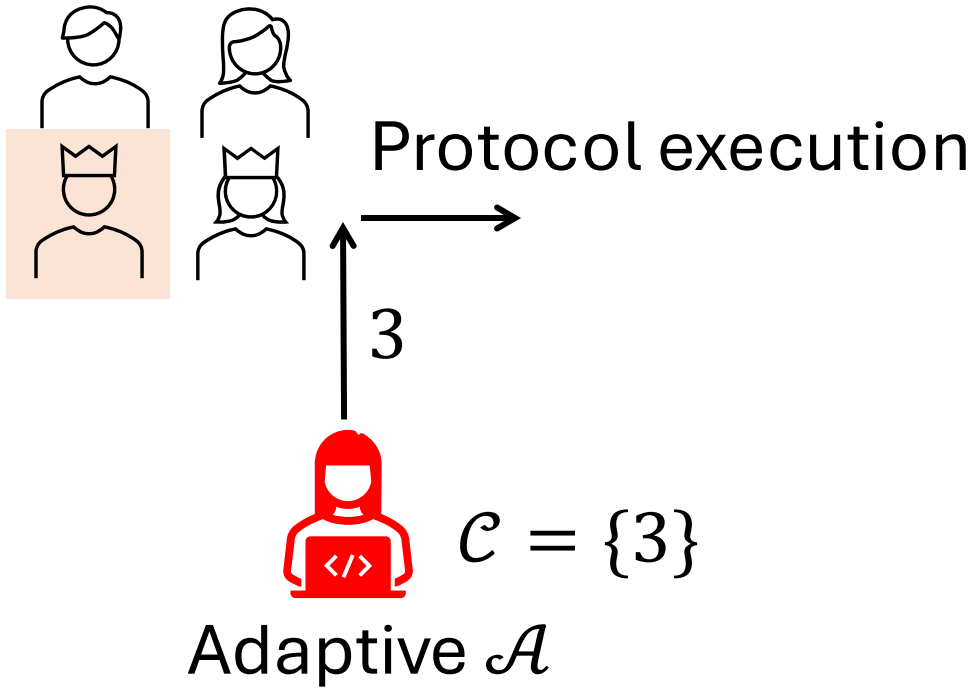
1. Non-interactive signing
2. Constant signature size
3. Constant verification cost
4. Unique signatures

Security of BLS Threshold Signature

Threat model: Static vs Adaptive Corruption

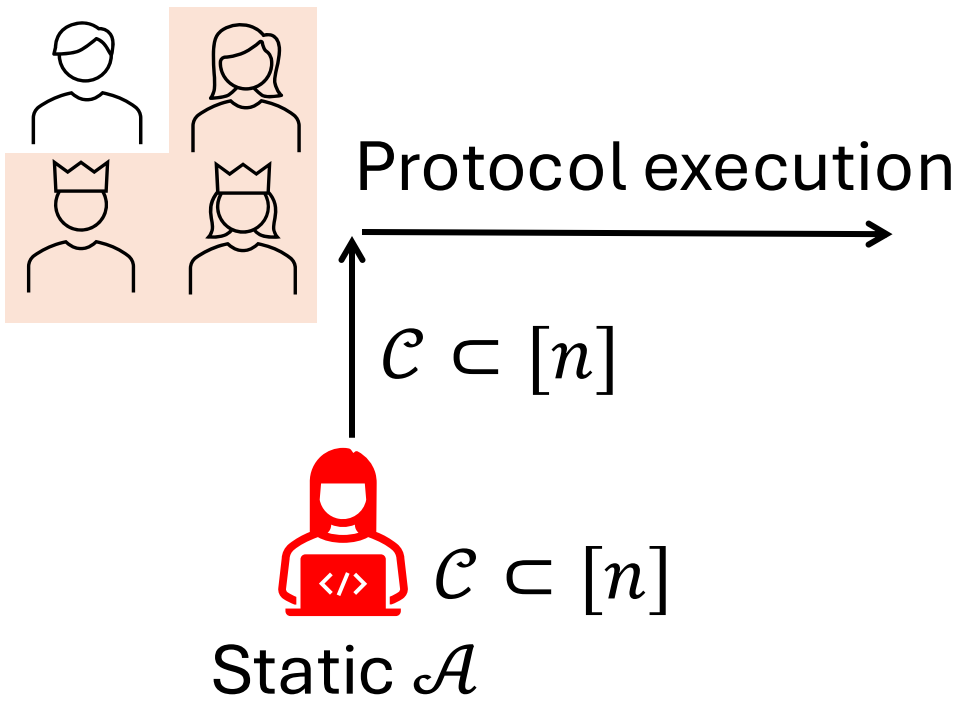


Decide corrupt parties before the protocol

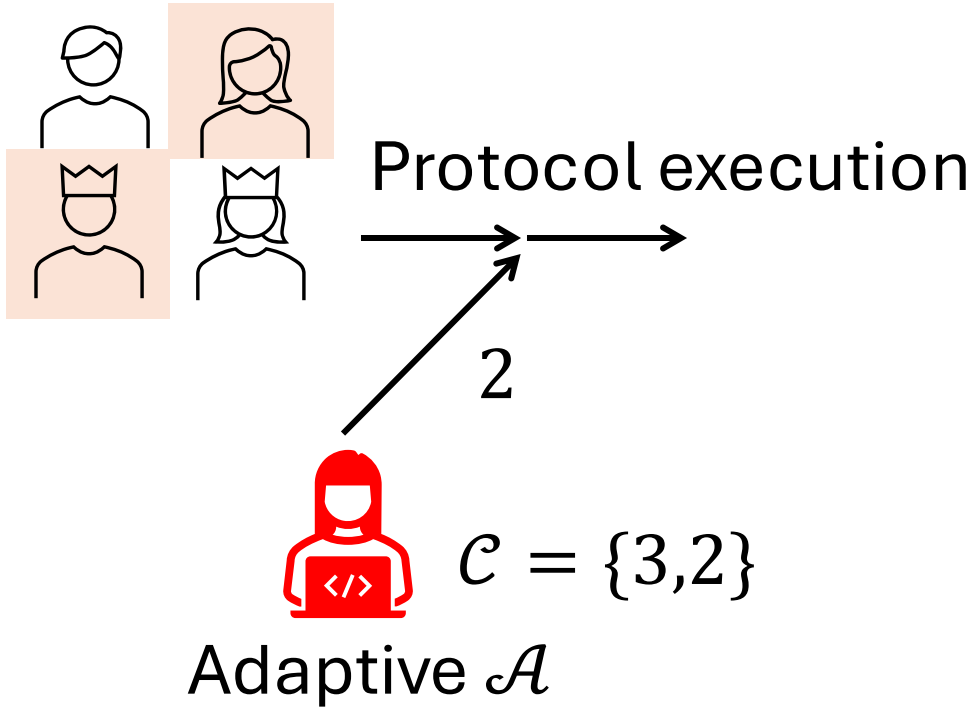


No timing restriction on corruption decision

Threat model: Static vs Adaptive Corruption

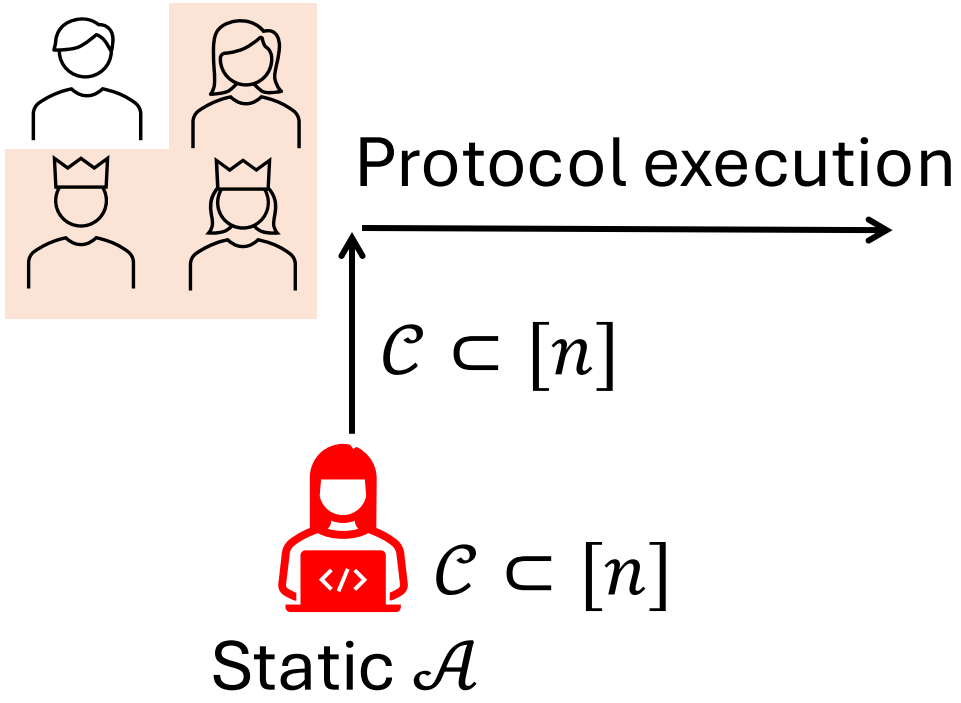


Decide corrupt parties before the protocol

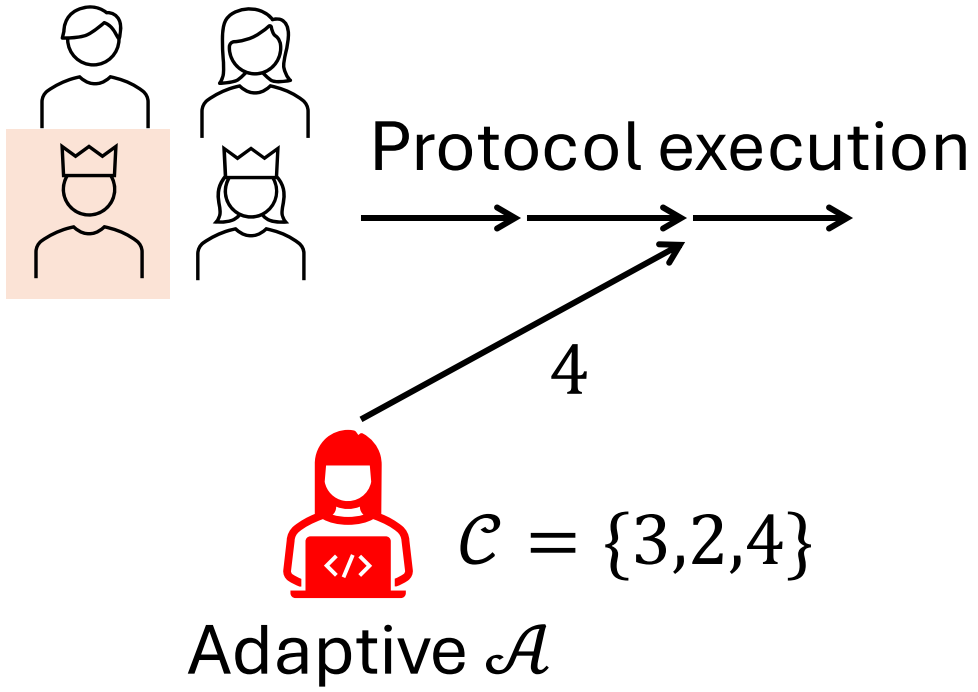


No timing restriction on corruption decision

Threat model: Static vs Adaptive Corruption

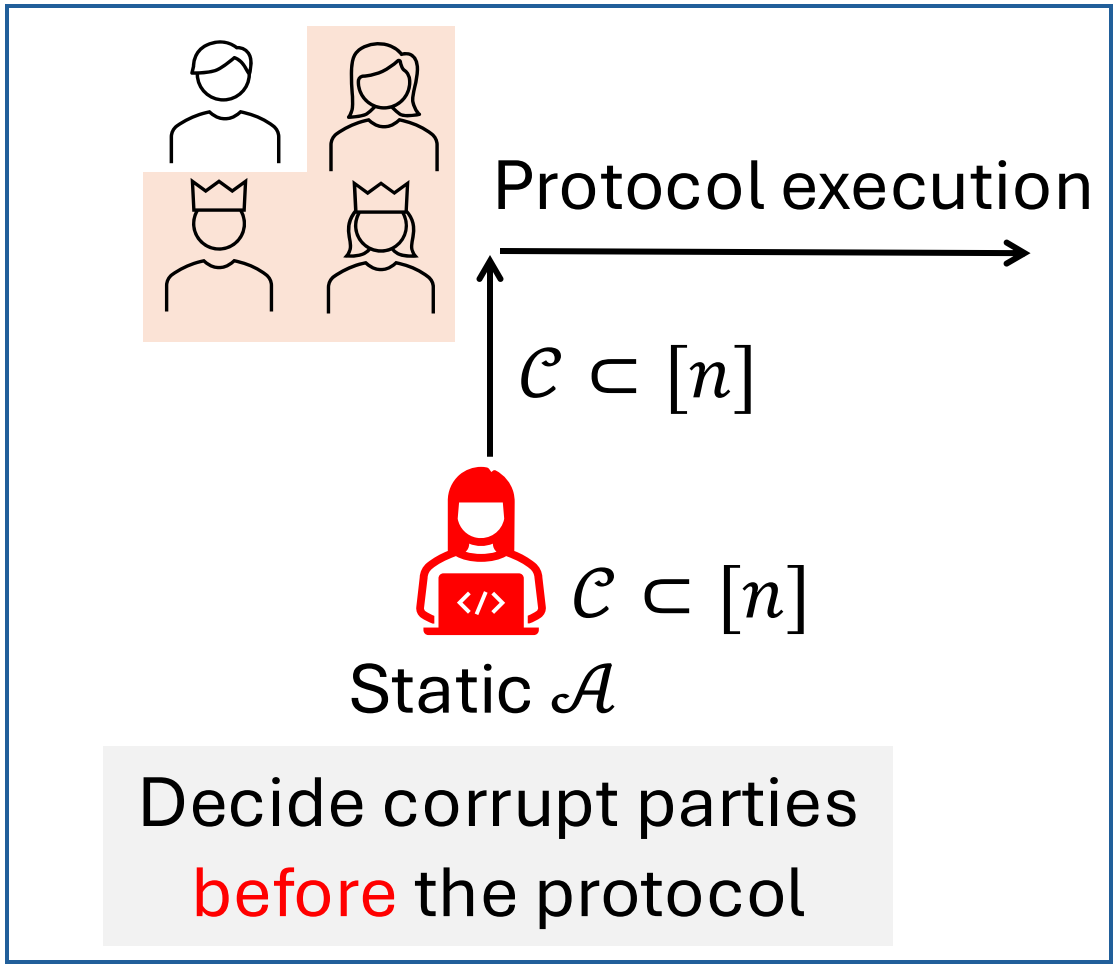


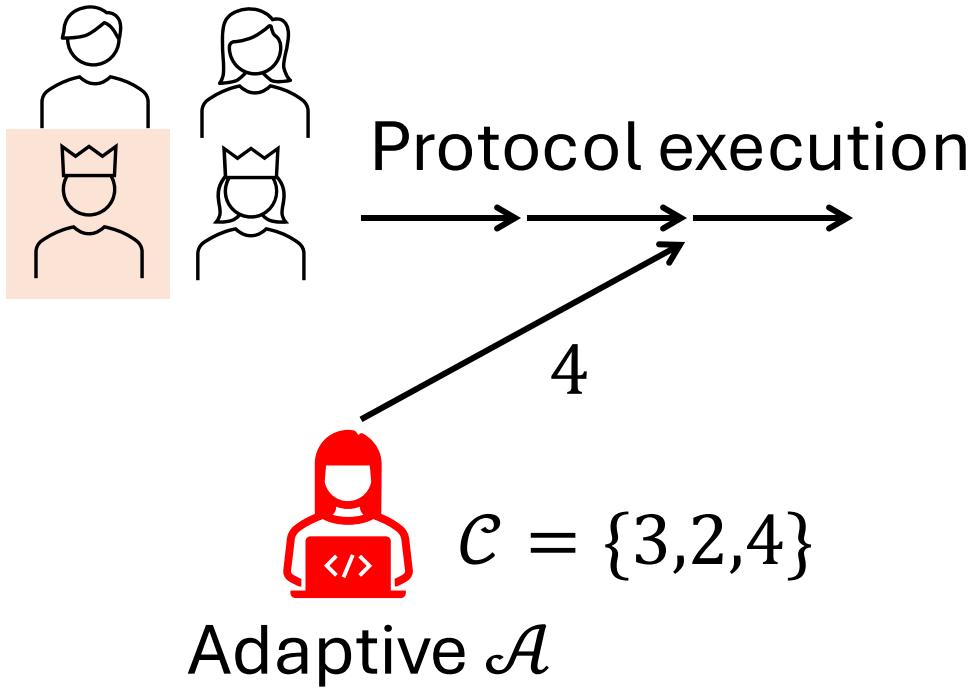
Decide corrupt parties before the protocol



No timing restriction on corruption decision

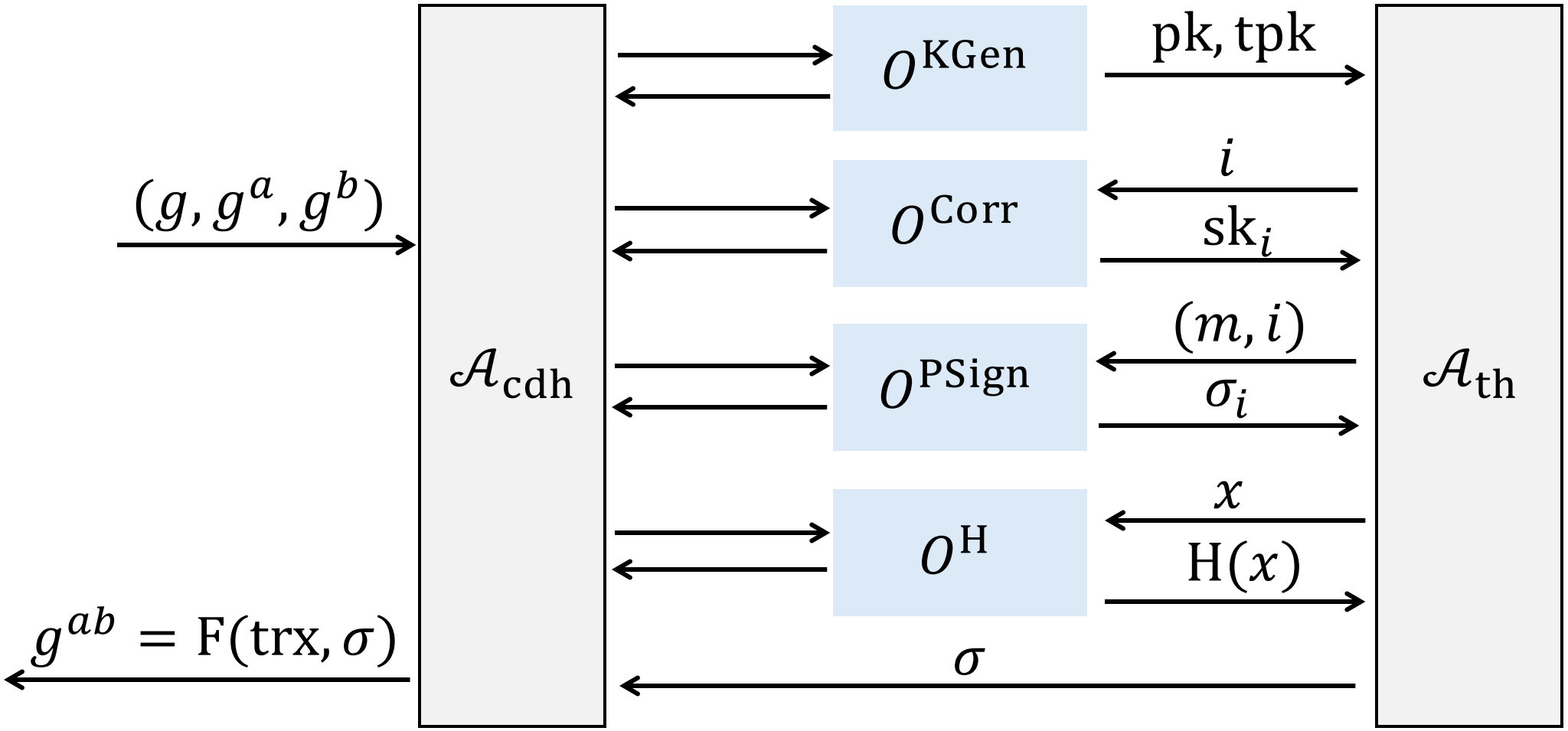
Threat model: Static vs Adaptive Corruption





Static Security of Threshold BLS Signature

High-level framework



O^{Corr} is the trickiest to simulate

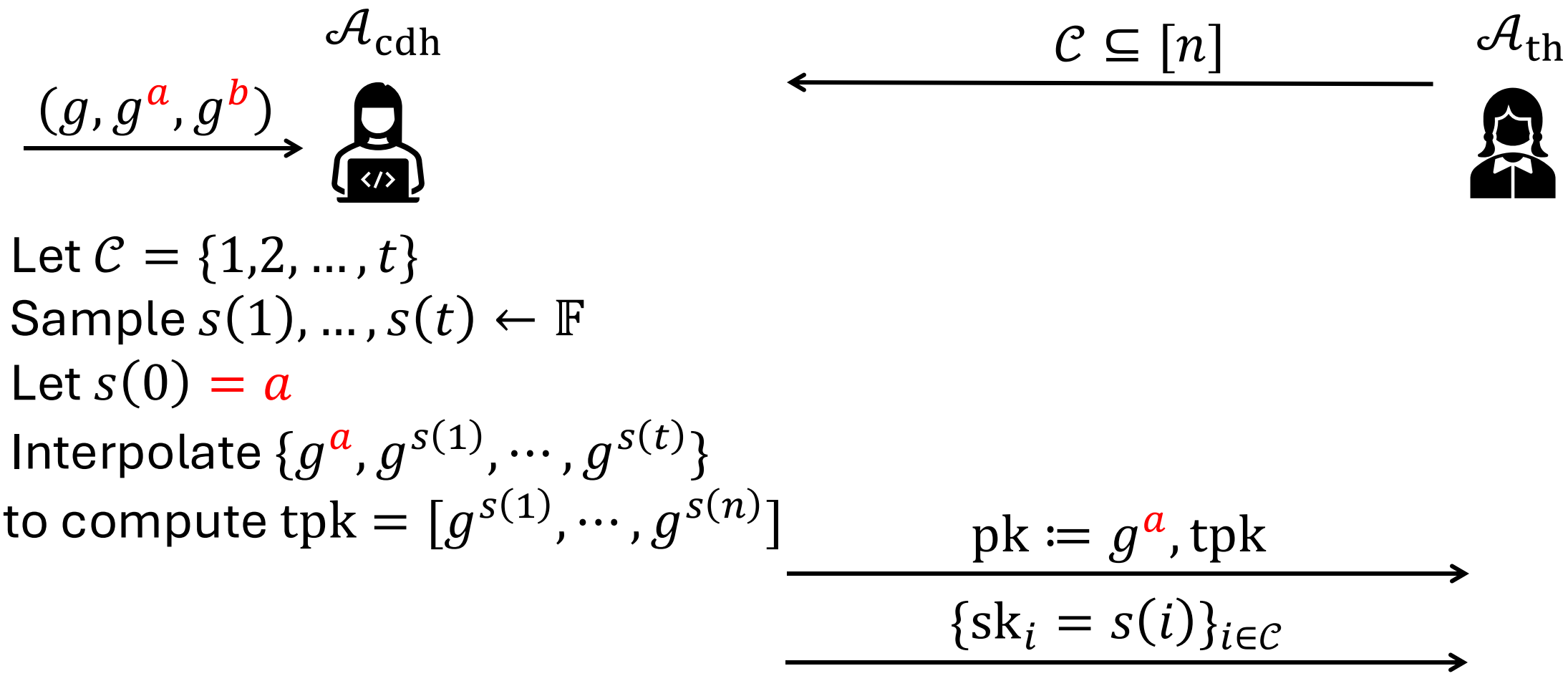
Static Security of Threshold BLS: Breaking CDH

$\mathcal{A}_{\mathrm{cdh}}$

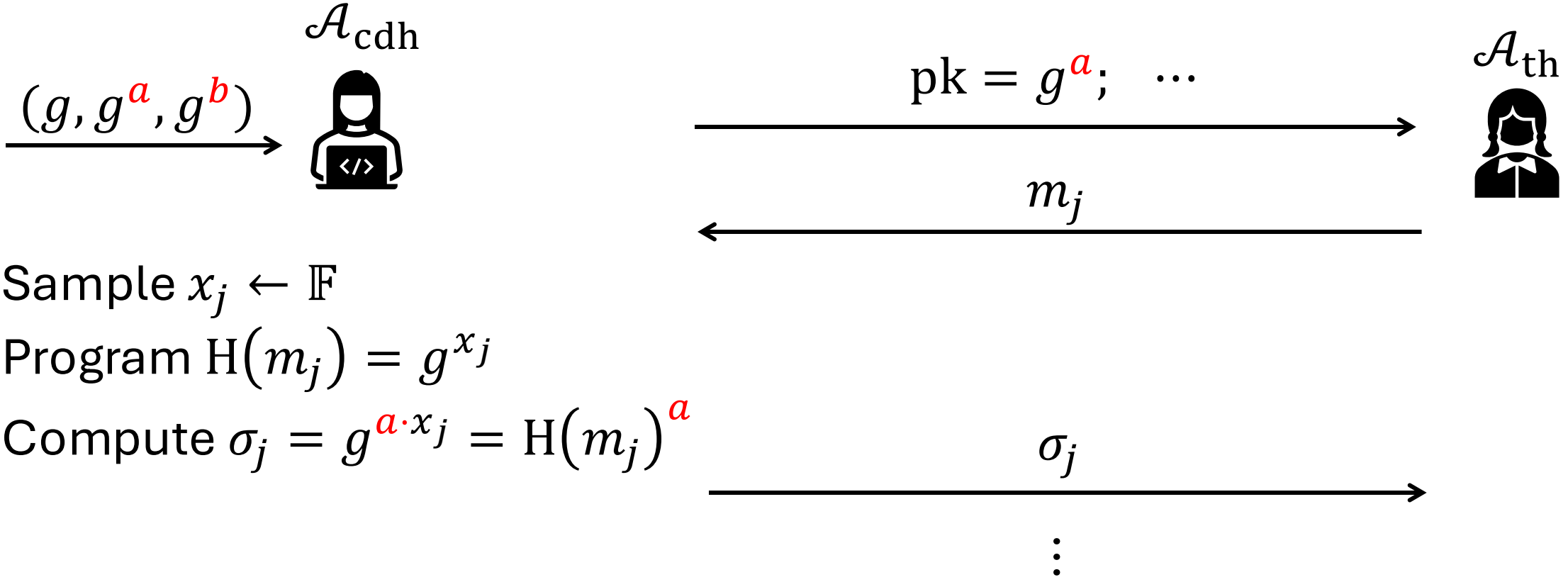
(g, g^{a}, g^{b}) →

$\mathcal{A}_{\mathrm{th}}$

Simulating Corruption Queries

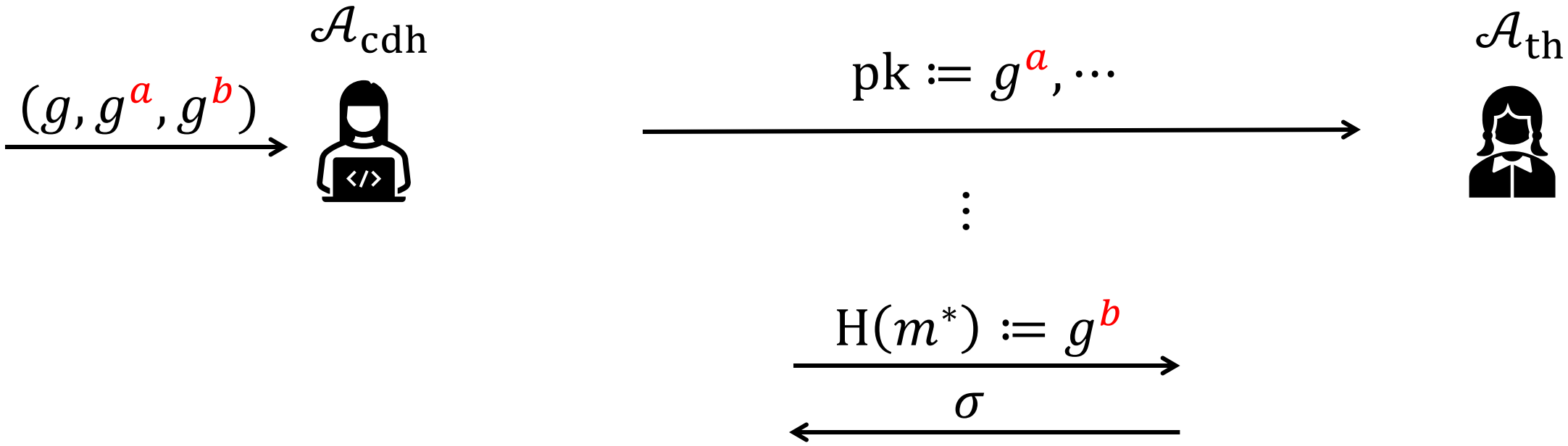


Simulating Signing Queries



1. Sample $x_j \leftarrow \mathbb{F}$
2. Program $\text{H}(m_j) = g^{x_j}$
3. Compute $\sigma_j = g^{a \cdot x_j} = \text{H}(m_j)^a$

Breaking CDH



$$e\big(\mathrm{pk}, \mathrm{H}(m^*)\big) = e(g, \sigma)$$
$$\Rightarrow e\big(g^a, g^b\big) = e(g, \sigma) \Rightarrow \sigma = g^{ab}$$

$\sigma = g^{ab}$ is the CDH solution!

See our paper [DR24] for adaptive security proof.

Evaluation Results

Some evaluation results

- Implementation in Golang (gnark-crypto)
- bls12381 elliptic curve
- t3.2xlarge AWS, 32 GB RAM, 8 virtual cores, 2.50GHz CPU

Scheme	Signing time (ms)	Partial signature verification time (ms)	Partial signature size (bytes)
Boldyreva-I	0.81	1.12	96
Boldyreva-II	1.20	0.76	160
Adaptive BLS	3.92	2.16	224

Common case aggregation time (for t=64) is 7.7 ms for all schemes!

Source: https://github.com/sourav1547/adaptive-bls

Summary

- Background on Bilinear pairing
- Boneh-Lynn Shacham (BLS) Signature
- Threshold Secret Sharing
- Design of Threshold BLS Signature
- Security of Threshold BLS Signature
- Evaluation Results

Implementation exercise (in Python)

https://github.com/sourav1547/cs598ftd/tree/main/bls

Thank you! (https://sourav1547.github.io/)