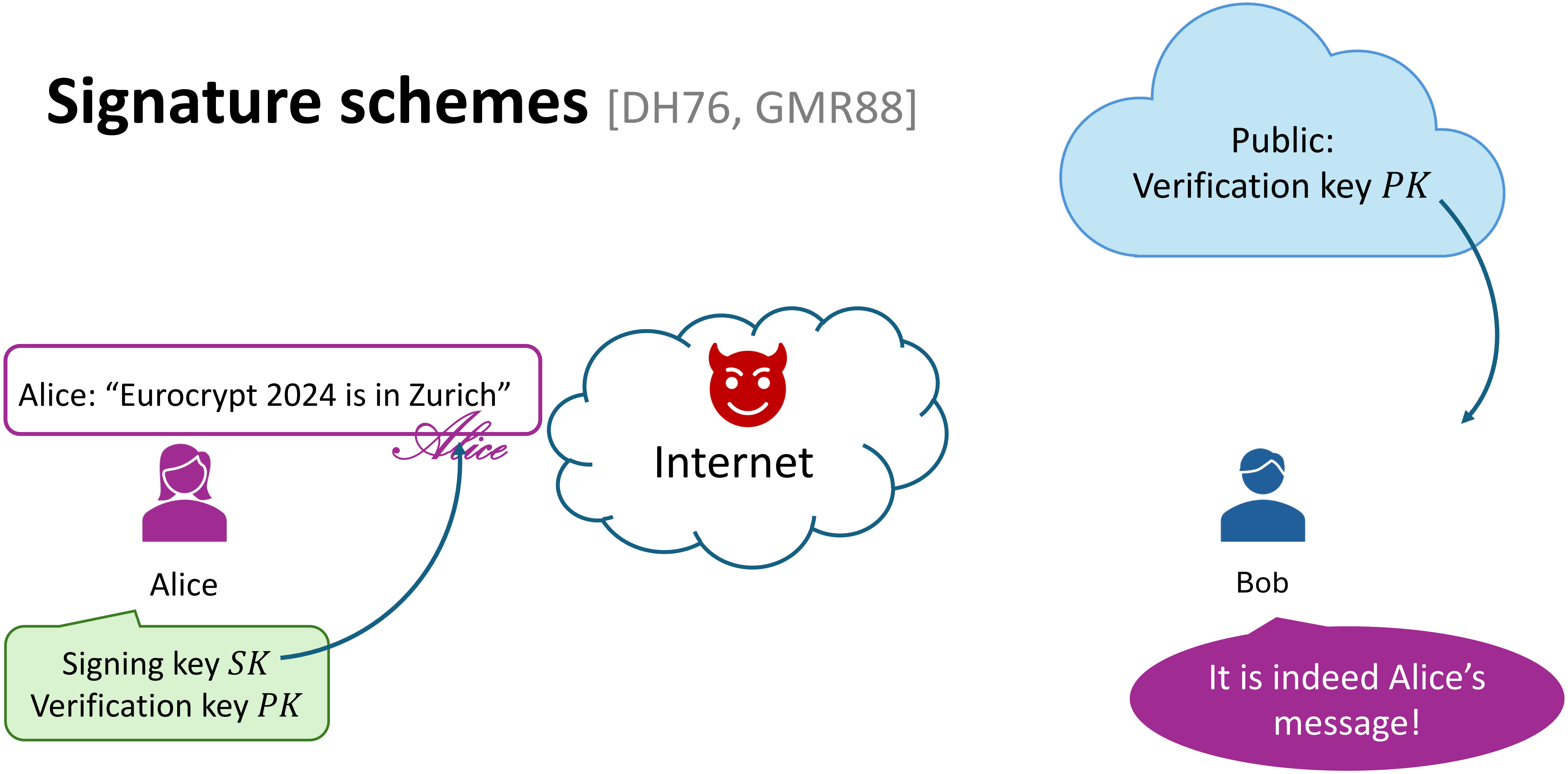
SPRINT: High-Throughput Robust Distributed Schnorr Signatures

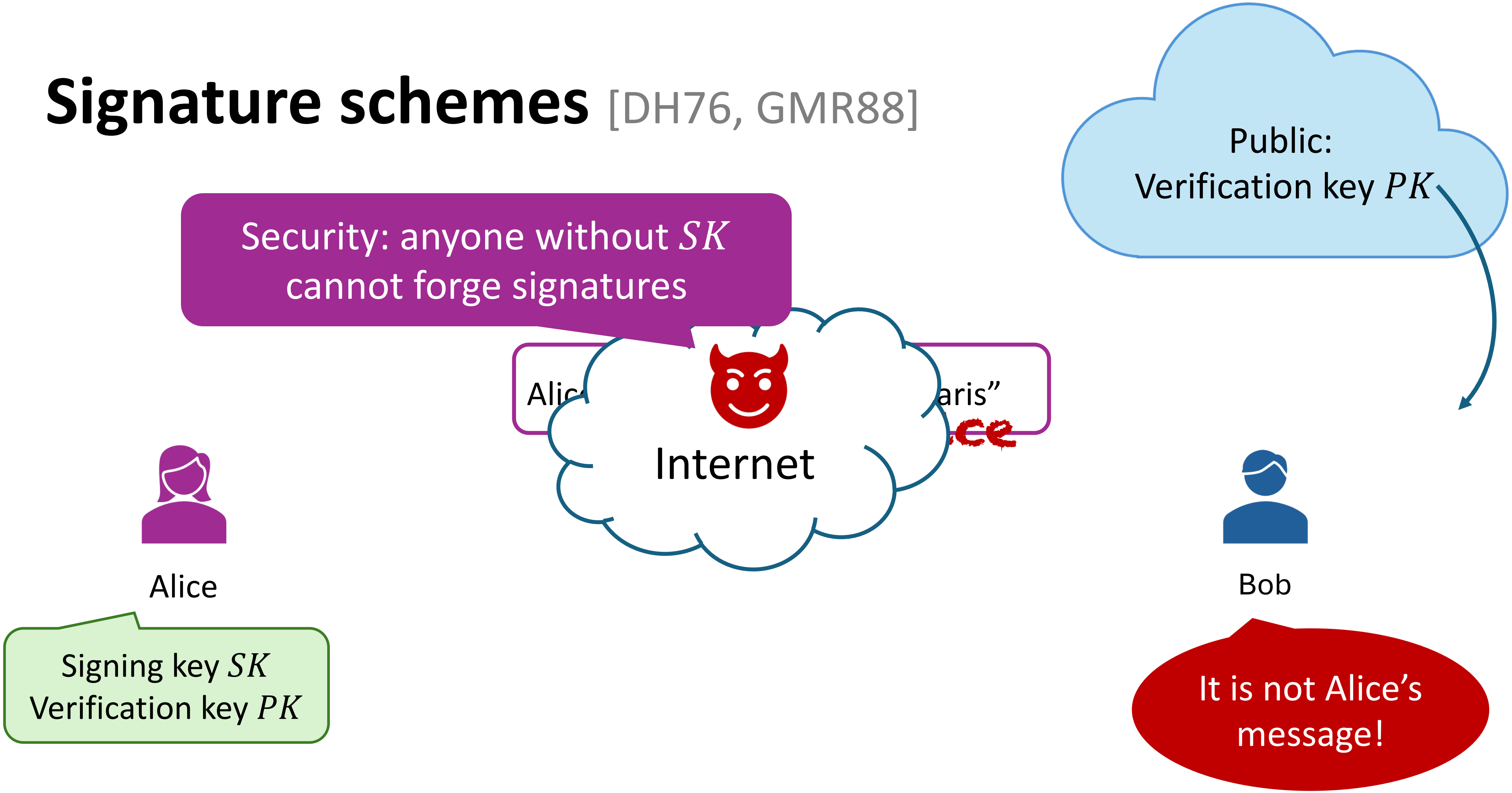
Fabrice Benhamouda* Shai Halevi* Hugo Krawczyk* Yiping Ma Tal Rabin*

*Work done partially while at the Algorand Foundation.

Signature schemes [DH76, GMR88]



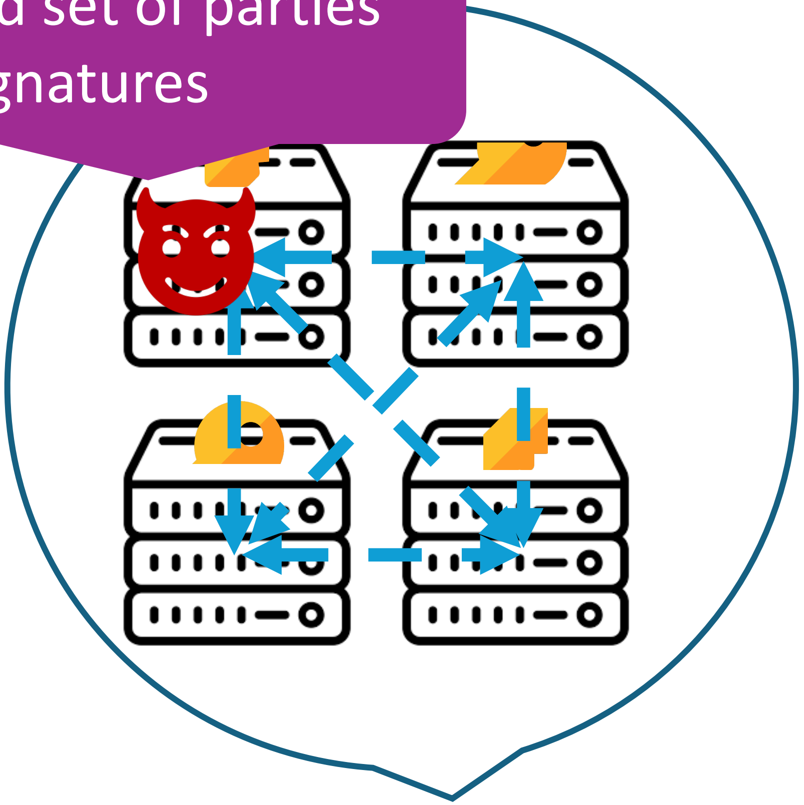
Signature schemes [DH76, GMR88]



Threshold signature [Des88, DF90, Ped91]

SK should be kept secret

Security: the corrupted set of parties cannot forge signatures

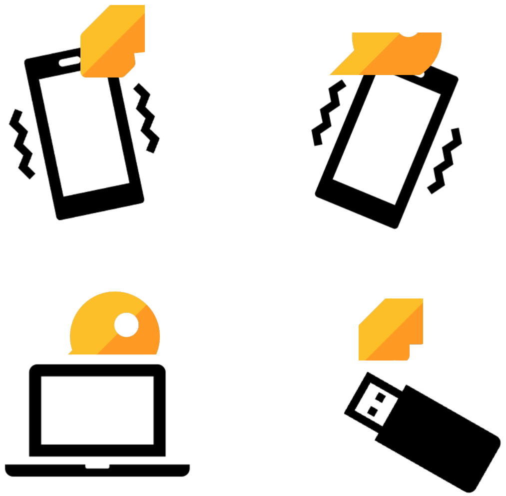


Alice: “Eurocrypt 2024 is in Zurich”

Alice

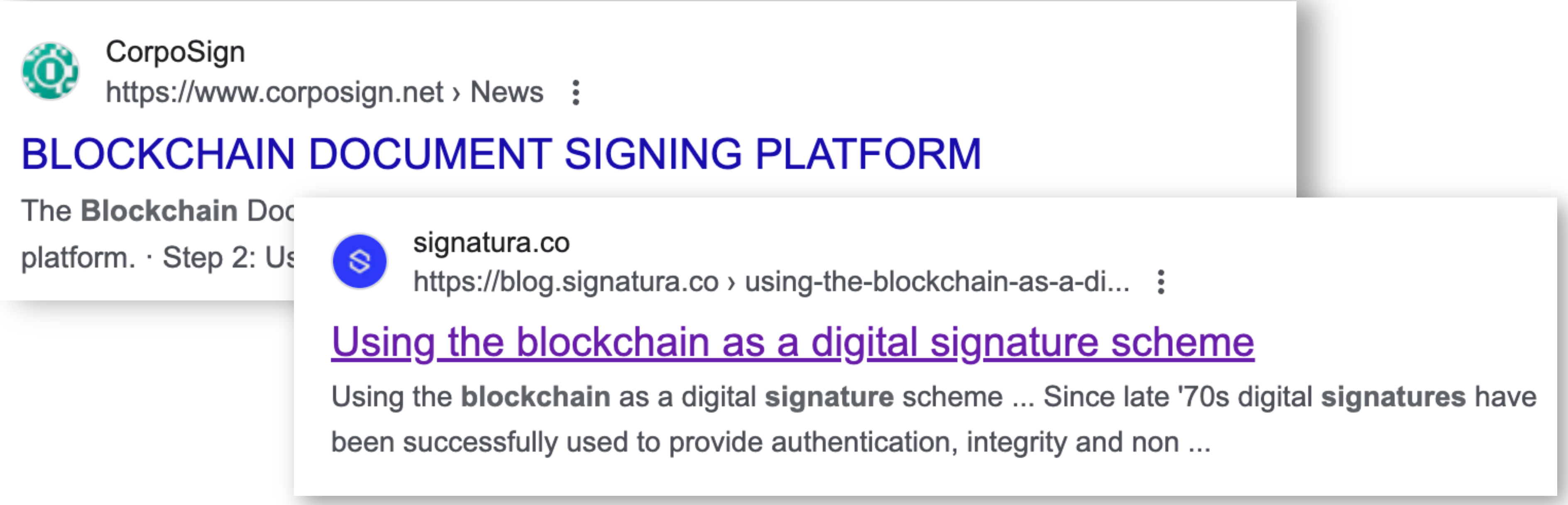
Threshold signature: applications

- Prior works are efficient in the setting of a small set of parties
  [KG20, CKM21, AB21, NRS21, BCK+22, TZ23, CKM23, …]



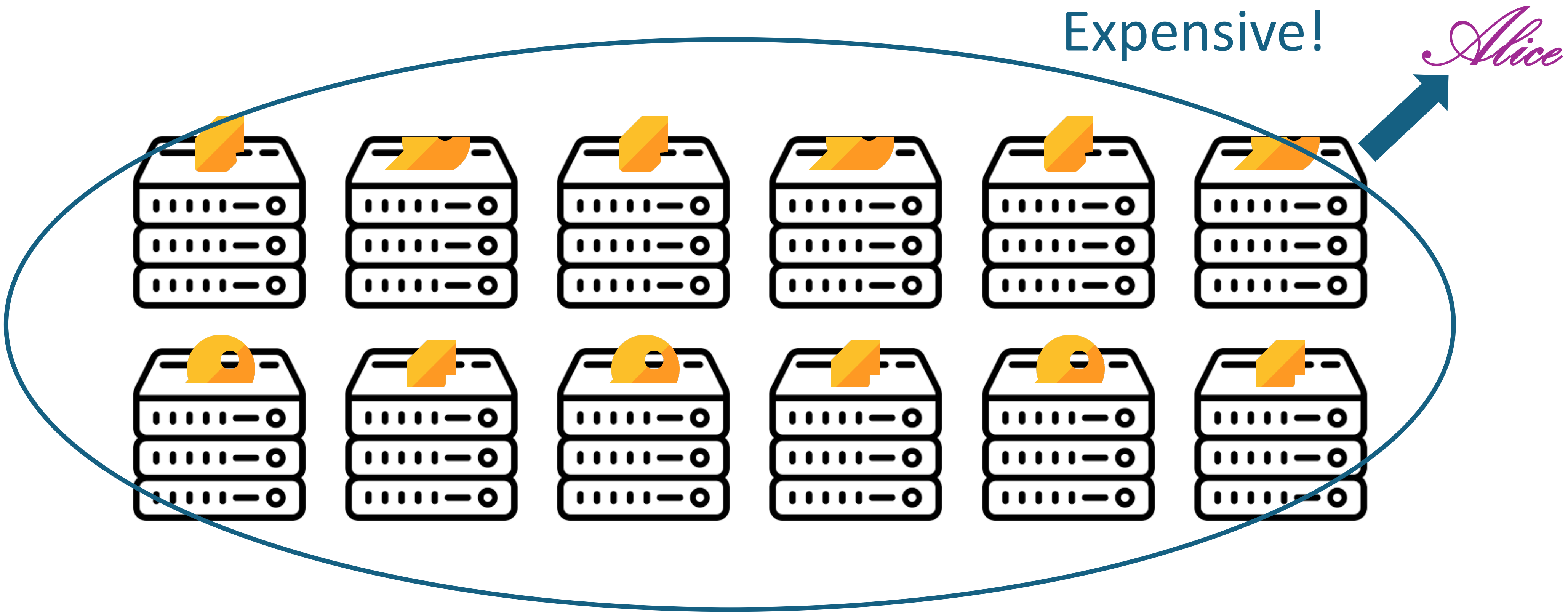
Threshold signature: applications

- This work deals with a large set of parties
- E.g., blockchain where #parties is in the hundreds



SPRINT: key ideas

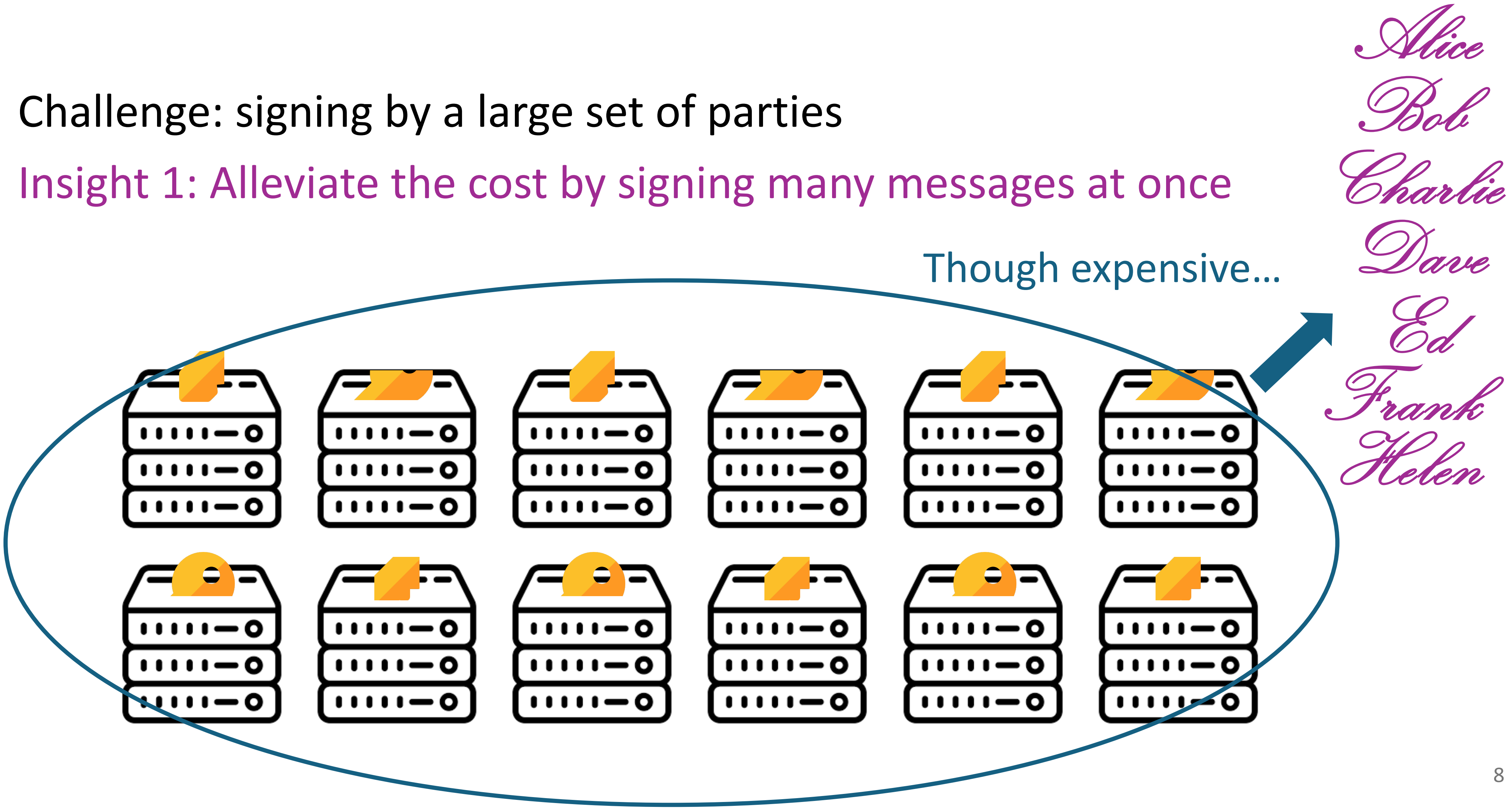
- Challenge: signing by a large set of parties



SPRINT: key ideas

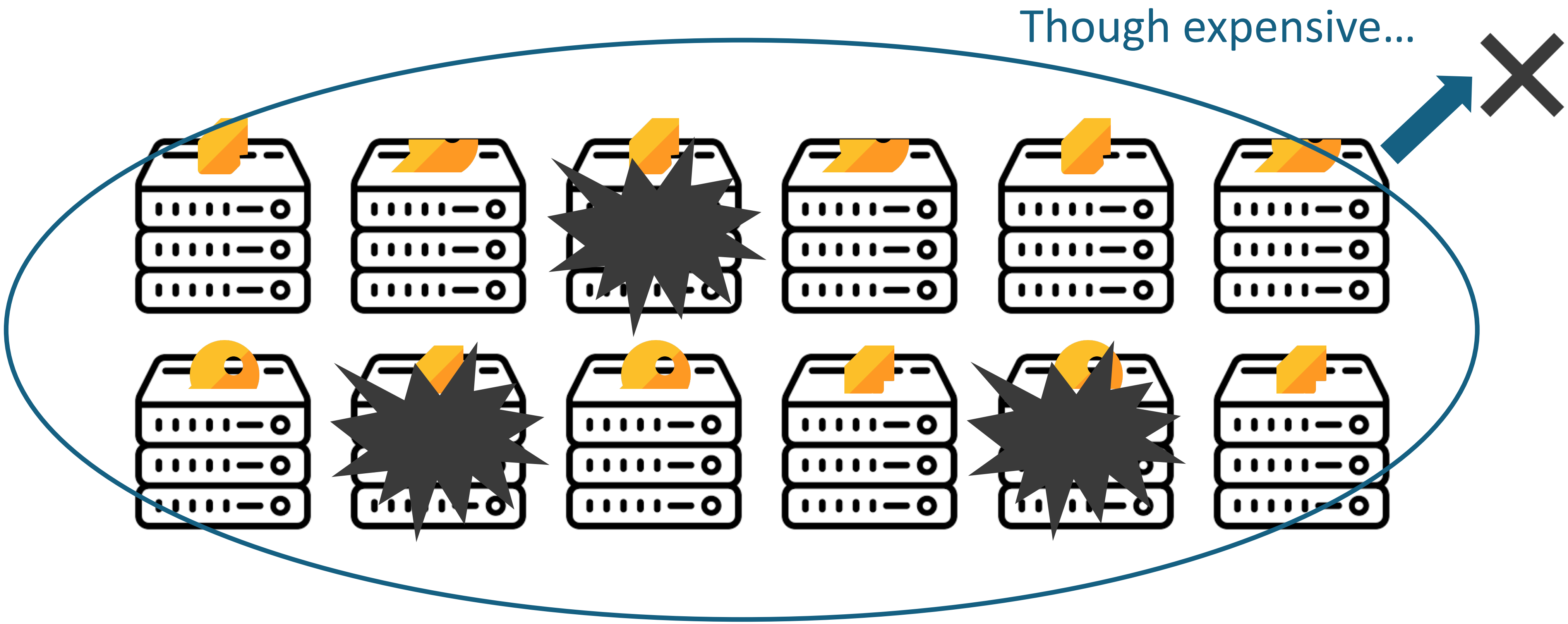
- Challenge: signing by a large set of parties
- Insight 1: Alleviate the cost by signing many messages at once

Though expensive…



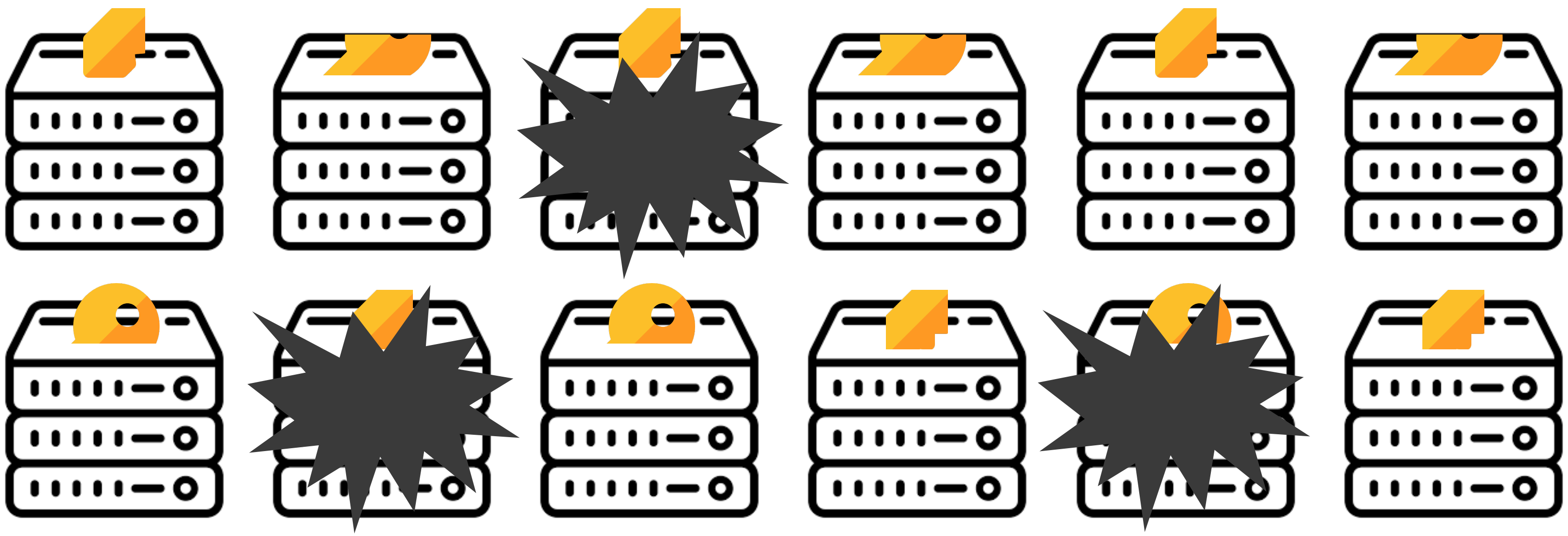
SPRINT: key ideas

- Challenge: signing by a large set of parties
- Insight 1: Alleviate the cost by signing many messages at once



SPRINT: key ideas

- Challenge: signing by a large set of parties
- Insight 1: Alleviate the cost by signing many messages at once
- Insight 2: Eliminate the effects caused by malicious parties



SPRINT: a bird's-eye view

- SPRINT has security and robustness

Critical when the set of parties is large

- Two-round message-independent preprocessing
- One round non-interactive signing: many messages at once

Theorem

Optimal resilience if we want robustness

Given $n \geq 3t + 1$ parties of which t are corrupted,
SPRINT generates $(n - 2t)$ signatures.

SPRINT: a bird's-eye view

- SPRINT has security and robustness

Critical when the set of parties is large

- Two-round message-independent preprocessing
- One round non-interactive signing: many messages at once

Theorem (ir

Tradeoff between resilience and throughput

Given $n \geq 3t + 2a - 1$ parties of which t are corrupted, SPRINT generates $a(n - 2t)$ signatures.

SPRINT: a bird's-eye view

- SPRINT has security and robustness
- Two-round message-independent preprocessing
- One round non-interactive signing: many messages at once

Quadratic number of signatures

Constant amortized cost

t	a	#signatures	bcast scalars/group elements per signature
$n/4$	$n/8$	$n^2/16$	~34
$n/5$	$n/5$	$3n^2/25$	~18

Feasible even when $n = 1000$

SPRINT: a bird's-eye view

- SPRINT has security and robustness
- Two-round message-independent preprocessing
- One round non-interactive signing: many messages at once

Quadratic number of signatures

Constant amortized cost

t	a	#signatures	bcast scalars/group elements per signature
$n/4$	$n/8$	$n^2/16$	~34
$n/5$	$n/5$	$3n^2/25$	~18

Less corruption

More signatures

Smaller amortized cost

Feasible even when $n = 1000$

Outline

- **Preliminaries**
- SPRINT techniques
  - Extreme packing and SIMD
  - Early-termination agreement
- Details and discussion

Schnorr signature

- Notation: Throughout this talk, we use additive notation for groups
  - A group $\mathbb{G}$ of order p in which DL is hard, generator G
  - Hash function $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_p$

Signing (secret) key	$s \xleftarrow{\$} \mathbb{Z}_p$
Verification (public) key	$S \coloneqq s \cdot G$

Schnorr signature

- Notation:
  - A group $\mathbb{G}$ of order p in which DL is hard, generator G
  - Hash function $\mathcal{H}: \{0,1\}^* \rightarrow \mathbb{Z}_p$

Secret s
Public S

$\underline{\text{Sign}}$

presignature

$r \xleftarrow{\$} \mathbb{Z}_p\,, \quad R \leftarrow r \cdot G$

$e = \mathcal{H}(S, R, M)$

$\phi = r + es$

Output (R, ϕ)

Public S

$\underline{\text{Verify}(S, M, (R, \phi))}$

Let $e = \mathcal{H}(S, R, M)$

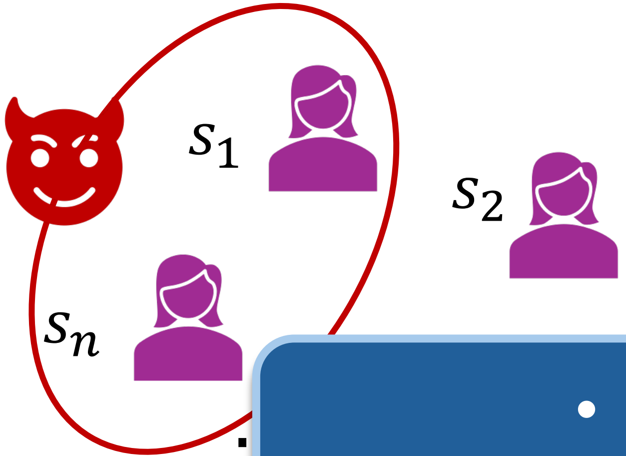
Check if $\phi \cdot G = R + e \cdot S$

$$\phi \cdot G = (r + es) \cdot G = R + e \cdot S$$

Threshold Schnorr

(Degree-t) sharing of s

- Assuming:
  - Signing key s is Shamir-shared to $[s] = (s_1, \dots, s_n)$ with threshold t
  - Verification key S is known to all



$$\underline{\mathrm{ThrSign}([s], S, M)}$$

$$[r] \xleftarrow{\$} \mathbb{Z}_p\ , \ \ R = r \cdot G$$

$$e = \mathcal{H}(S, R, M)$$

Presignature generation

- Only sign a single message
- May not be secure or robust

SPRINT: main techniques

- "Extreme packing + SIMD computation"

Many $[r] \xleftarrow{\$} \mathbb{Z}_p$
Many $R = r \cdot G$

Operate on one share,
sign multiple messages

"Insight 1: Alleviate the cost by signing many messages at once"

- "An early-termination agreement" (this work assumes async. setting)

Ensure good $[r] \xleftarrow{\$} \mathbb{Z}_p$
and hence good $R = r \cdot G$

Not exactly a DL-DKG, but sufficiently
good for signature purpose

"Insight 2: Eliminate the effects caused by malicious parties"

Outline

- Preliminaries
- **SPRINT techniques**
  - Detailed → **Extreme packing and SIMD**
  - Brief → Early-termination agreement
- Details and discussion

Orthogonal to security/robustness

$$[r] \xleftarrow{\$} \mathbb{Z}_p, \ R = r \cdot G$$

[GJKR07]

- Parameters: n parties with t collusion
- Sketch (strawman presignature generation):
  one round, each party contributes a random polynomial

$$[r] \xleftarrow{\$} \mathbb{Z}_p,\ R = r \cdot G$$

[GJKR07]

- Round 1. Each P_i (dealer) sends to each P_j (shareholder) a share $H_i(j)$

	P_1	P_2	P_3	…	P_n
P_1	$H_1(1)$	$H_1(2)$	$H_1(3)$	…	$H_1(n)$
P_2	$H_2(1)$	$H_2(2)$	$H_2(3)$	…	$H_2(n)$
P_3	$H_3(1)$	$H_3(2)$	$H_3(3)$	…	$H_3(n)$
			…		
P_n	$H_n(1)$	$H_n(2)$	$H_n(3)$	…	$H_n(n)$

Polynomial H_1 of degree t

$$[r] \xleftarrow{\$} \mathbb{Z}_p,\ R = r \cdot G$$

[GJKR07]

- Round 1. Each P_j locally adds the shares: let $H = \sum_{i=1}^{n} H_i$, and $r_j := H(j)$

	P_1	P_2	P_3	…	P_n
	$H_1(1)$	$H_1(2)$	$H_1(3)$	…	$H_1(n)$
	$H_2(1)$	$H_2(2)$	$H_2(3)$	…	$H_2(n)$
	$H_3(1)$	$H_3(2)$	$H_3(3)$	…	$H_3(n)$
			…		
+	$H_n(1)$	$H_n(2)$	$H_n(3)$	…	$H_n(n)$
	$H(1)$	$H(2)$	$H(3)$	…	$H(n)$

Polynomial H_1 of degree t

Defines $r = H(0), r_j = H(j)$

$$[r] \xleftarrow{\$} \mathbb{Z}_p, \ R = r \cdot G$$

[GJKR07]

- Round 1. P_i broadcast $R_i \coloneqq H_i(0) \cdot G$, then $R = \sum_{i=1}^{n} R_i$

$$\begin{array}{lr}
P_1 & H_1(0) \cdot G \\
P_2 & H_2(0) \cdot G \\
P_3 & H_3(0) \cdot G \\
 & \cdots \\
P_4 & + \ H_n(0) \cdot G \\
\hline
 & H(0) \cdot G
\end{array}$$

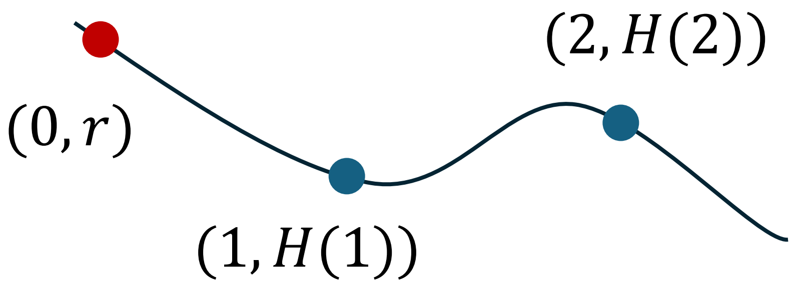
Addition in the group

Presignature R

“Many” $[r] \xleftarrow{\$} \mathbb{Z}_p$ and $R = r \cdot G$ using packed secret sharing

Standard Shamir

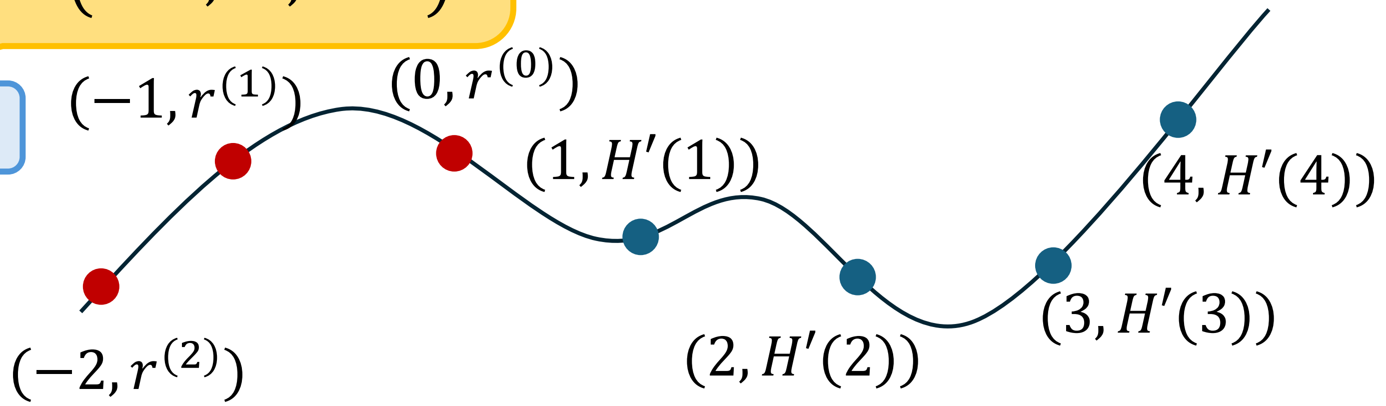
- Polynomial H of degree t
- Hide the secret at point 0
- Collusion threshold t



Sharing of $\left(r^{(1)}, \ldots, r^{(a)}\right)$, presignatures $\left(R^{(1)}, \ldots, R^{(a)}\right)$

Packed Shamir [FY92]

- Polynomial H' of degree $t + a - 1$
- Hide a secrets at $0, -1, \ldots, -a+1$
- Collusion threshold t



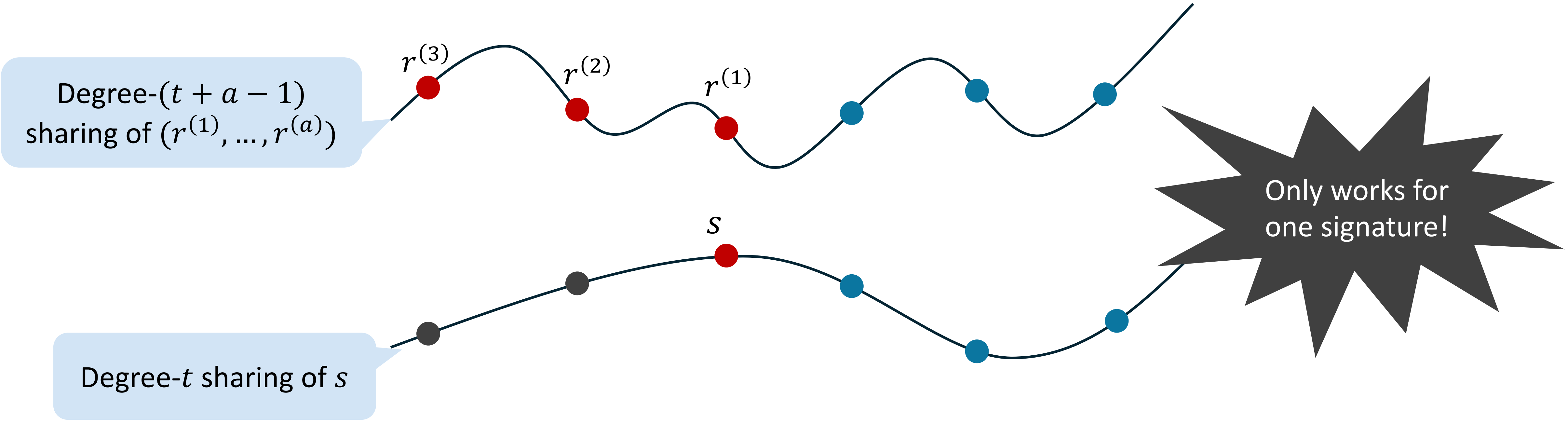
Still, each party holds only one share

Compute sharing of $\left(\phi^{(k)} := r^{(k)} + e^{(k)}s\right)_{k=1,\ldots,a}$

- What we have:
  - Sharing of a random values: $\left(r^{(1)}, \ldots, r^{(a)}\right)$
  - Presignatures $\left(R^{(1)}, \ldots, R^{(a)}\right)$
  - Messages $(M^{(1)}, \ldots, M^{(a)})$
  - Public values $(e^{(1)}, \ldots, e^{(a)})$

Compute sharing of $\left(r^{(k)} + e^{(k)} s\right)_{k=1,\dots,a}$

- Simpler: compute sharing of $\left(r^{(k)} + \boxed{}\, s\right)_{k=1,\dots,a}$



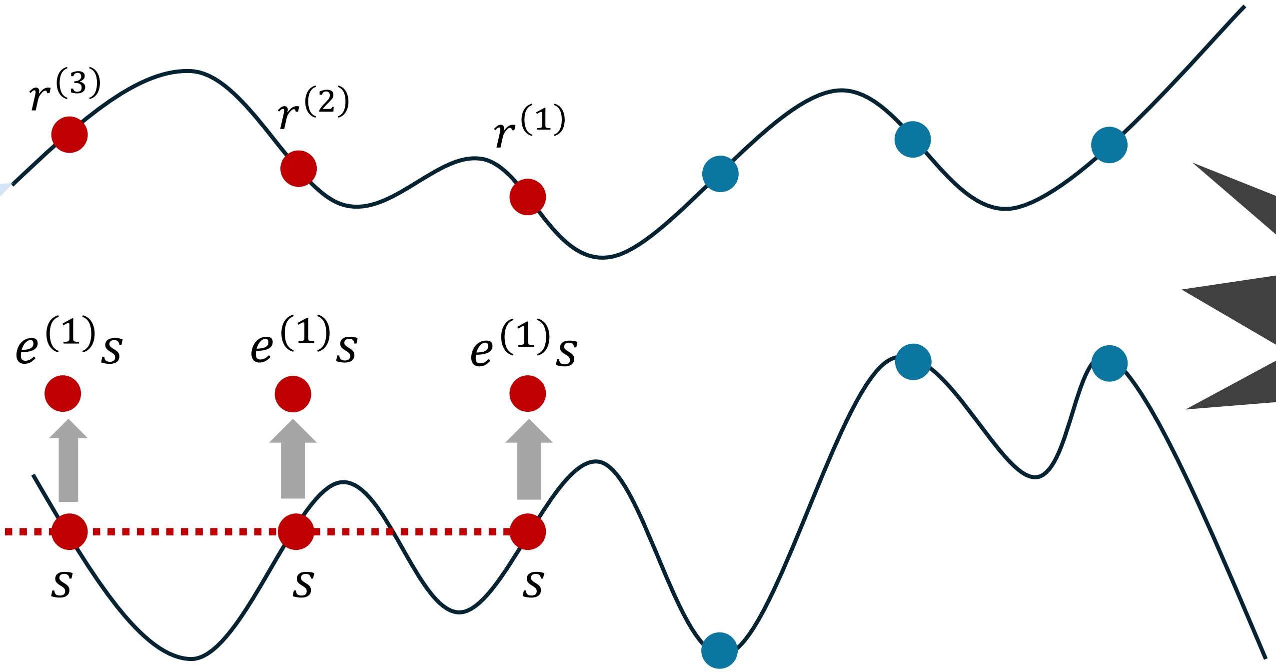
Compute sharing of $\left(r^{(k)} + e^{(k)} s\right)_{k=1,\ldots,a}$

- Compute sharing of $\left(r^{(k)} + \boxed{}\, s\right)_{k=1,\ldots,a}$

How to deal with $e^{(k)}$ in packed sharing?

Degree-$(t+a-1)$ sharing of $(r^{(1)}, \ldots, r^{(a)})$

Degree-$(t+a-1)$ sharing of $(s, \ldots, s)$ i.e., a multiples of s



Still works for one signature…

SPRINT: SIMD technique

- Packed sharing of $\left(r^{(1)}, \dots, r^{(a)}\right)$
- Packed sharing of $(s, s, \dots, s)$
- Public values $(e^{(1)}, \dots, e^{(a)})$

Polynomial H of degree $t + 2a - 2$

Polynomial F of degree $t + a - 1$

Polynomial E of degree $a - 1$

- The packed sharing of $\left(r^{(1)}, \dots, r^{(a)}\right) + \left(e^{(1)}, \dots, e^{(a)}\right) \cdot s$ can be computed as

$$H + E \cdot F$$

- Each party P_j locally computes $H(j) + E(j) \cdot F(j)$
- A little loss in resilience: $n \geq t + 2a - 1$ instead of $n \geq t + 1$

Extreme packing using super-invertible matrices

- We defined $H = \sum_{i=1}^{n} H_i$ for signing randomness
- Each H_i itself can be used as signing randomness $\Rightarrow$ boost by a factor of n?
- #random polynomials = #honest dealers = b

But we don't know which!

Super-invertible matrix [HN06] $\cdot \begin{bmatrix} H_1 \\ \cdots \\ \cdots \\ \cdots \\ H_n \end{bmatrix} = \begin{bmatrix} U_1 \\ \cdots \\ \cdots \\ U_b \end{bmatrix}$

They are independently random

Result in b sharing of length-a randomness

Outline

- Preliminaries
- **SPRINT techniques**
  - Detailed → Extreme packing and SIMD
  - Brief → **Early-termination agreement**
- Details and discussion

Ensure good sharing of $(r^{(1)}, \dots, r^{(a)})$

- Parameters: n parties with t collusion
- A bad party: contribute a polynomial such that reconstruction fails!
- Sketch (robust presignature generation):
  - Round 1. Each party contributes a polynomial (supposed to be the right degree)
  - New Round 2. Agree on a set of good polynomials (“good” = has the right degree)
- Result: agree on at least $n - t$ good polynomials (in the async. setting)
- So we have $b \geq n - 2t$ good random polynomials
- In total $a(n - 2t)$ sigantures

Outline

- Preliminaries
- SPRINT techniques
  - Extreme packing and SIMD
  - Early-termination agreement
- **Details and discussion**

Details

- Hashing and re-randomization [GS22]
  - $\delta = \mathcal{H}(S, \mathrm{QUAL}, \{R^{(i)}, M^{(i)}\}_{i \in \{1,\ldots,ab\}})$
  - $\Delta = \delta \cdot G$
  - Use $R + \Delta$ to replace the previous R
- The use of Feldman commitments to polynomials
  - Not a secure DKG: slightly biased key when adversary is rushing [GJKR07]
  - We proved that for signature purpose it is fine
- Dynamic committees and how to sub-sample them

Summary

SPRINT
- $\Omega(n^2)$ signatures per run assuming $\Omega(n)$ corruption
- Two-round presignature generation + one round non-interactive signing

- Concurrent security:
  [Shoup23] makes SPRINT concurrently secure in a black-box way
- Can we achieve better tradeoff between resilience and efficiency?

Thanks!

Backup slides

Robust presignature generation

Parameters: n parties with t corrupted

- Round 1. Each party P_i chooses a random degree-t polynomial H_i, broadcasts Feldman commitments to H_i

P_1 $\quad H_1(0) \cdot G \quad H_1(1) \cdot G \quad H_1(2) \cdot G \quad H_1(3) \cdot G \quad \ldots \quad H_1(t) \cdot G$

Feldman commitment to H_1

Feldman commitment

Everyone can verify a given r_{1j} claimed to be $H_1(j)$ is indeed correct:

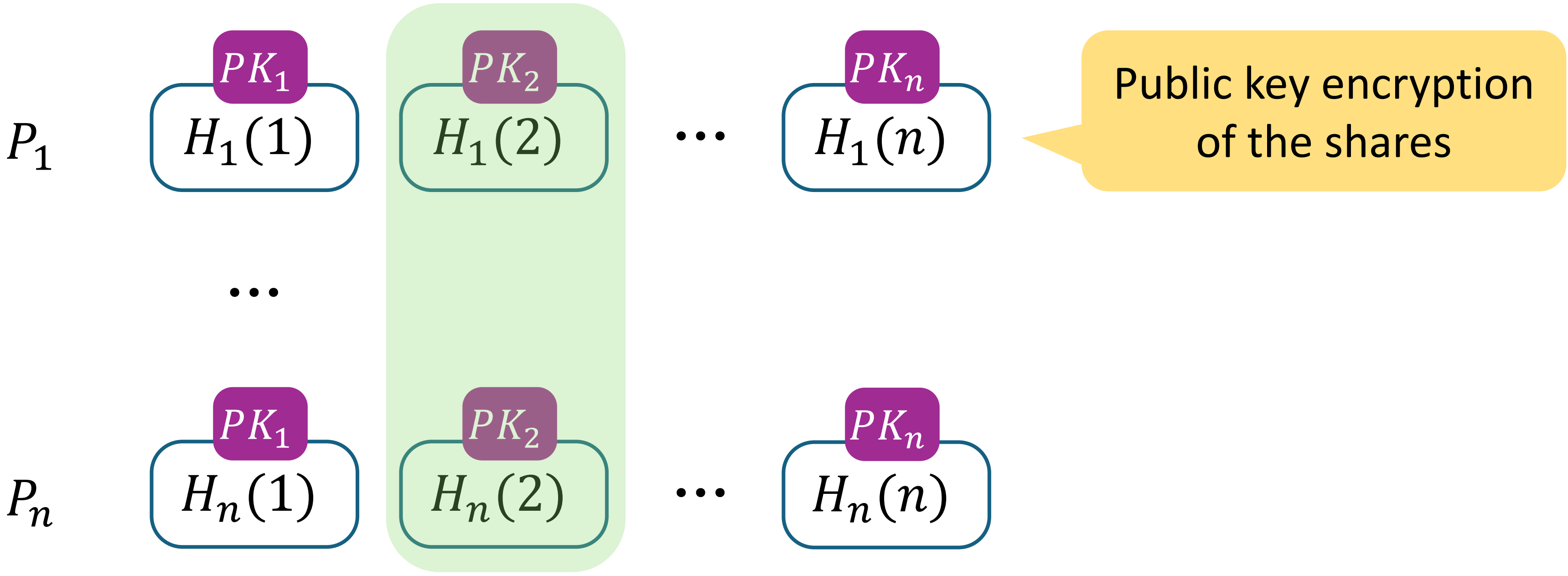
- For $j = 0, \ldots, t$ it is easy to check: compare $r_{1j} \cdot G$ with the commitment
- For $j = t+1, \ldots, n$, we can use "interpolation on the exponent":

Compare $r_{1j} \cdot G$ with $H_1(j) \cdot G = \lambda_0\,(H_1(0) \cdot G) + \lambda_1(H_1(1) \cdot G) + \cdots + \lambda_t(H_1(t) \cdot G)$

Can compute this!

Robust presignature generation

- Round 1. Party i broadcasts $\mathrm{Enc}(PK_j, r_{ij})$ where $r_{ij} = H_i(j)$



Robust presignature generation

- Round 2. Parties agree on a set QUAL that contains dealers who send valid shares (that lie on a polynomial of degree-t)

P_1 | PK_1: $H_1(1)$ | $\cdots$ | PK_j: r_{1j} | $\cdots$ | PK_n: $H_1(n)$

Public key encryption of the shares

- P_j decrypts to r_{1j}
- P_j verifies r_{1j} against Feldman commitment to H_1: is r_{1j} equal $H_1(j)$?
- How can an honest P_j convince others that P_1 is bad?

The simple agreement protocol

ElGamal encryption

- Observation 1: publicly verifiable complaint enabled by PKE of shares
- If P_j failed the verification against P_1's share, create a verifiable complaint:
  - r_{1j}
  - ZKP of decrypting the ciphertext $\mathrm{Enc}(PK_j, r_{1j})$

Proof of DL

- Each shareholder: exclude dealers who were complained about

Just ends here!

The simple agreement protocol

- Observation 2: no need for "completeness"
- Completeness [Groth-Shoup23]: all honest parties eventually have valid shares ⇒ possible to forgo polynomial commitments and rely on error correction
- We use verifiable complaints to disqualify bad dealers; we do not to help the complaining shareholders to get any more shares

Robust presignature generation: recap

- Round 1: Each party contributes a polynomial H_i

  (broadcast PKE of shares of H_i and Feldman commitment to H_i)
- Round 2: Each party broadcasts verifiable complaint if it has any

How many signatures we get?

- Agree on a set QUAL of “correct” polynomials H_i’s (in the async. setting):

$$|\mathrm{QUAL}| + |😈| \geq n - t$$

- Exclude those H_i’s that are correct but not random

$$b = |\mathrm{QUAL}| - (t - |😈|) \geq n - 2t$$

- Each polynomial packs a secrets
- We get $a(n - 2t)$ random values for signing