Threshold Encryption with Silent Setup

Sanjam Garg, Dimitris Kolonelos, **Guru Vamsi Policharla**, Mingyuan Wang

Threshold Encryption

Goal: Encrypt a message to n parties, such that

Threshold Encryption

Goal: Encrypt a message to n parties, such that

- Can be decrypted by any t out of n parties

Threshold Encryption

Goal: Encrypt a message to n parties, such that

- Can be decrypted by any t out of n parties
- Semantic security holds against any subset of $< t$ parties

Threshold Encryption

Goal: Encrypt a message to n parties, such that

- Can be decrypted by any t out of n parties
- Semantic security holds against any subset of $< t$ parties
- $|\text{Ciphertext}| = O(1)$

Threshold Encryption

Goal: Encrypt a message to n parties, such that

- Can be decrypted by any t out of n parties
- Semantic security holds against any subset of $< t$ parties
- |Ciphertext| $= O(1)$
- Non-interactive decryption

Textbook Solution

Threshold ElGamal seems quite nice:

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$
- Short partial decryption: $g^{r[sk]}$

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$
- Short partial decryption: $g^{r[sk]}$

… but it's not perfect

- $O(n^2)$ DKG for setup with *each* committee

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$
- Short partial decryption: $g^{r[sk]}$

… but it's not perfect

- $O(n^2)$ DKG for setup with *each* committee
- DKG more expensive in asynchronous settings + $< n/3$ corruption

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$
- Short partial decryption: $g^{r[sk]}$

… but it's not perfect

- $O(n^2)$ DKG for setup with *each* committee
- DKG more expensive in asynchronous settings + $< n/3$ corruption
- Threshold fixed at the time of setup

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(g, h = g^{sk})$
- Short ciphertexts: $(g^r, h^r \cdot M)$
- Short partial decryption: $g^{r[sk]}$

Almost universal issue

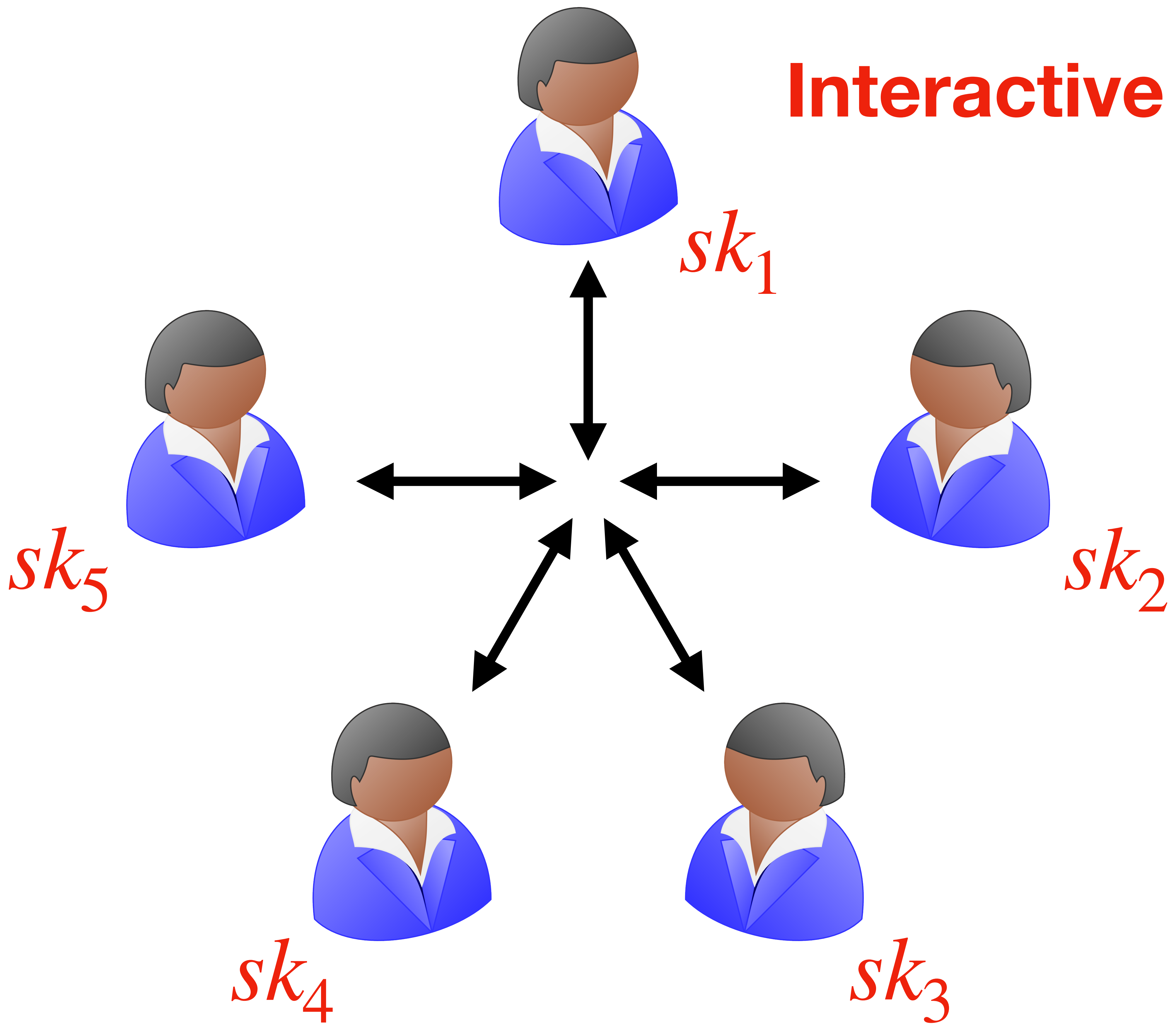
Today: **Silent** Threshold Encryption

… but it's not perfect

- $O(n^2)$ DKG for setup with *each* committee
- DKG more expensive in asynchronous settings + $< n/3$ corruption
- Threshold fixed at the time of setup

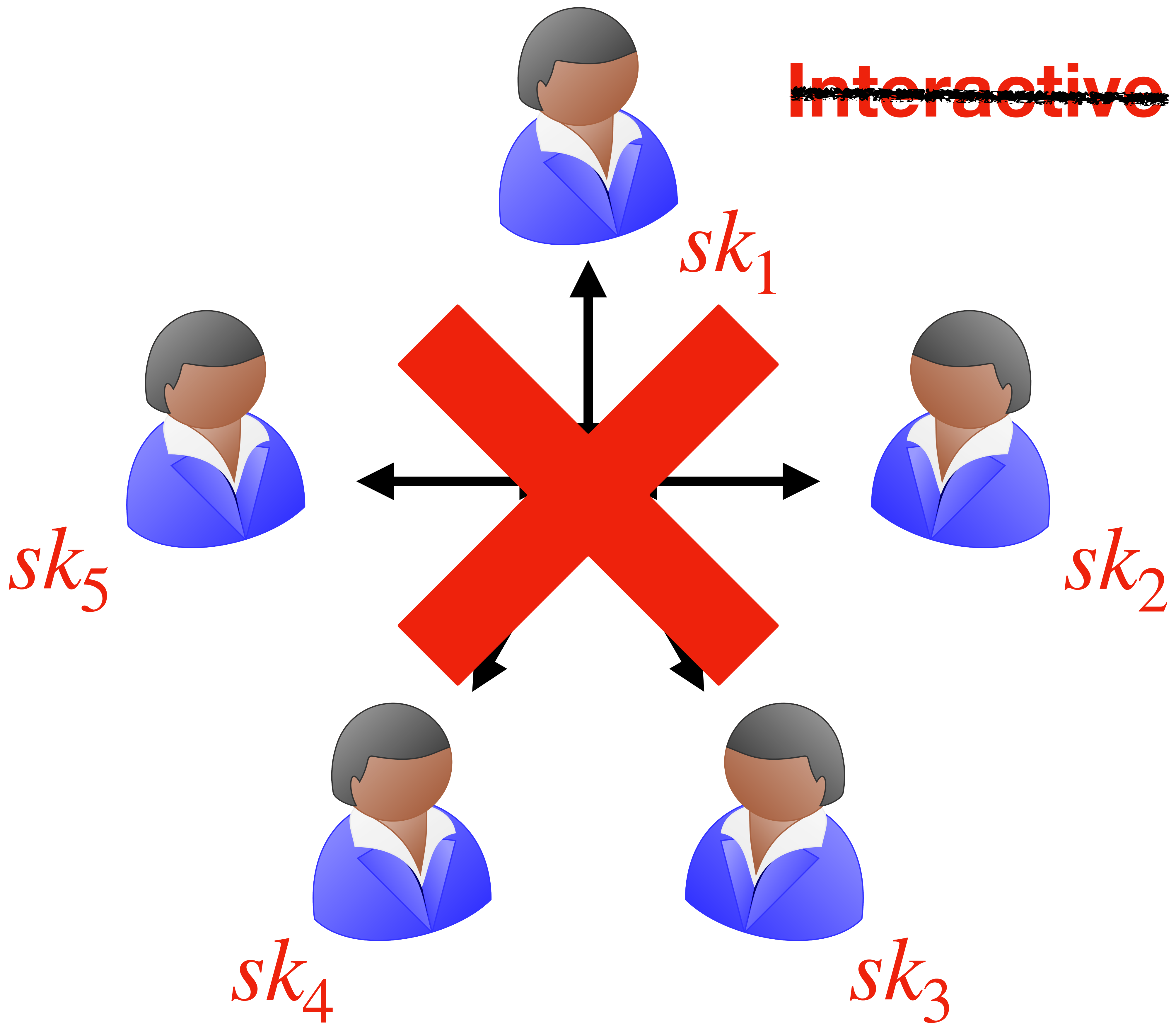
Threshold Encryption

Interactive Setup $\rightarrow pk$

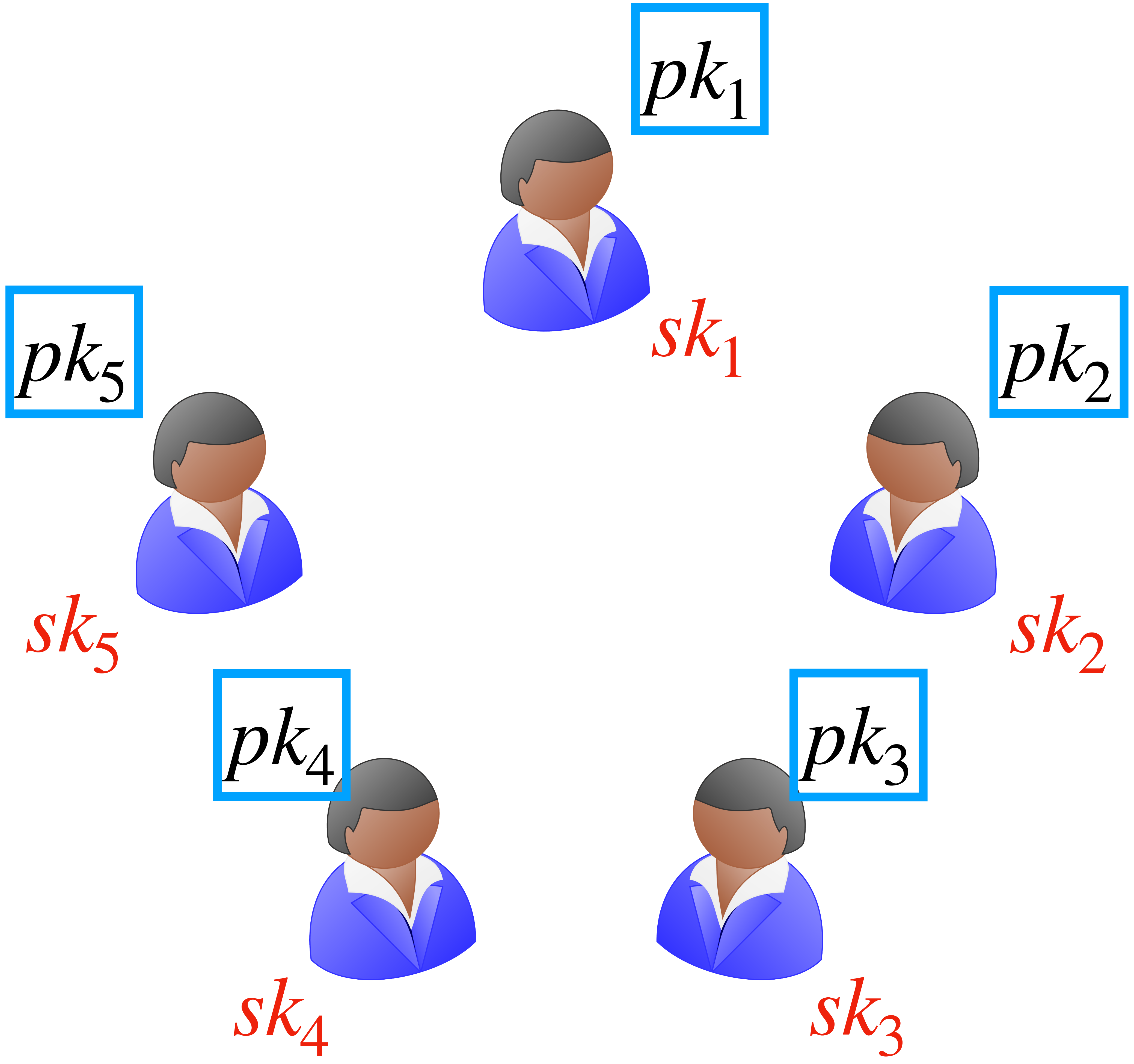


Silent Threshold Encryption

~~Interactive~~ Setup $\rightarrow pk$



Silent Threshold Encryption



Silent Threshold Encryption

$\boxed{pk_1}$

$\boxed{pk_5}$ sk_1 $\boxed{pk_2}$

sk_5 sk_2

$\boxed{pk_4}$ $\boxed{pk_3}$

sk_4 sk_3

Silent Threshold Encryption

$\boxed{pk_1}$

sk_1

$\boxed{pk_5}$ $\boxed{pk_2}$

sk_5 sk_2

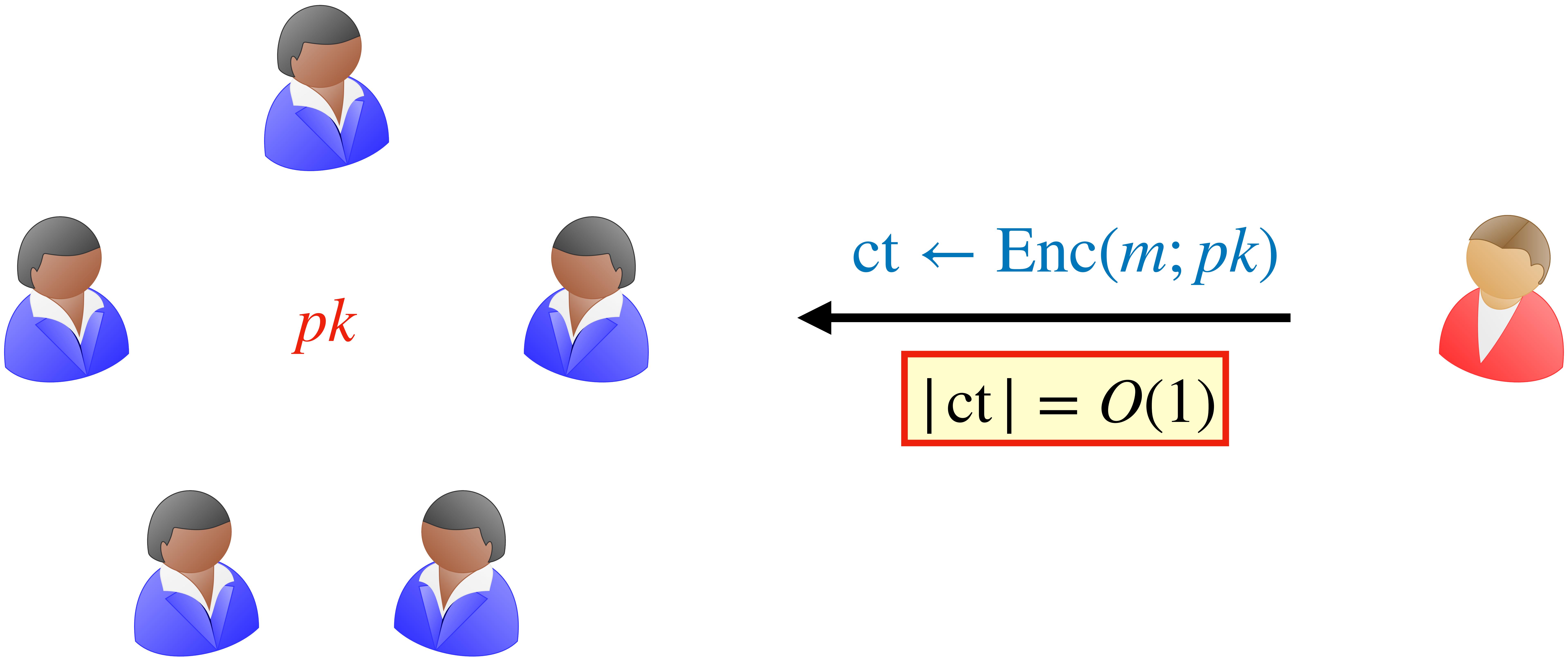
$\boxed{pk_4}$ $\boxed{pk_3}$

sk_4 sk_3

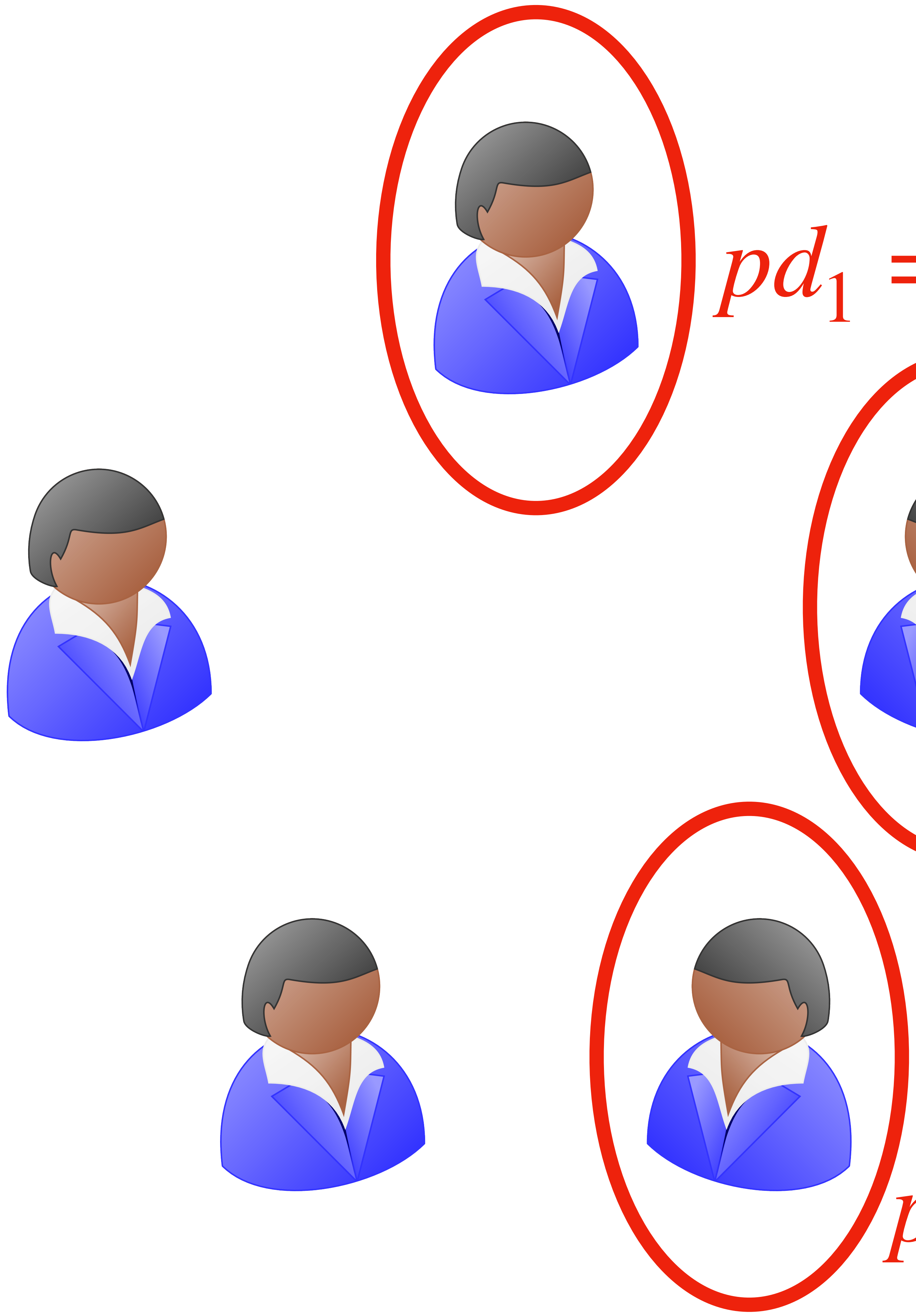
Deterministic Function:

$$\mathrm{Agg}(pk_1, pk_2, pk_3, pk_4, pk_5) \rightarrow pk$$

Silent Threshold Encryption



Silent Threshold Encryption



$pd_1 = \mathrm{Dec}(\mathrm{ct}, sk_1)$

$pd_2 = \mathrm{Dec}(\mathrm{ct}, sk_2)$

$pd_3 = \mathrm{Dec}(\mathrm{ct}, sk_3)$

Deterministic Function:

$\mathrm{Agg}(pd_1, pd_2, pd_3) \rightarrow m$

How do we compare?

Threshold ElGamal:

Silent Threshold:

How do we compare?

Threshold ElGamal:

- **Needs a DKG**
- Short public key: $O(1)$

Silent Threshold:

- Long individual public key: $O(n)$
- Short aggregated key: $O(1)$

How do we compare?

Threshold ElGamal:

- **Needs a DKG**
- Short public key: $O(1)$
- Short secret key: $O(1)$

Silent Threshold:

- Long individual public key: $O(n)$
- Short aggregated key: $O(1)$
- Short secret key: $O(1)$

How do we compare?

Threshold ElGamal:

- **Needs a DKG**
- Short public key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $1G + |\text{key}|$

Silent Threshold:

- Long individual public key: $O(n)$
- Short aggregated key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $2G_1 + 7G_2 + |\text{key}|$

How do we compare?

Threshold ElGamal:

- **Needs a DKG**
- Short public key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $1G + |\text{key}|$
- Short partial decryption: $1G$

Silent Threshold:

- Long individual public key: $O(n)$
- Short aggregated key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $2G_1 + 7G_2 + |\text{key}|$
- Short partial decryption: $1G_2$

How do we compare?

Threshold ElGamal:

- **Needs a DKG**
- Short public key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $1G + |\text{key}|$
- Short partial decryption: $1G$
- Reduces to DDH

Silent Threshold:

- Long individual public key: $O(n)$
- Short aggregated key: $O(1)$
- Short secret key: $O(1)$
- Short ciphertexts: $2G_1 + 7G_2 + |\text{key}|$
- Short partial decryption: $1G_2$
- Relies on GGM

Related Work

- (Flexible) Distributed Broadcast Encryption: FN94, WQZD10, BZ14, FWW23, KMW23, GLWW23

Related Work

- (Flexible) Distributed Broadcast Encryption: FN94, WQZD10, BZ14, FWW23, KMW23, GLWW23
- Adaptive Threshold: DP08, HLR10

Related Work

- (Flexible) Distributed Broadcast Encryption: FN94, WQZD10, BZ14, FWW23, KMW23, GLWW23
- Adaptive Threshold: DP08, HLR10
- Silent Threshold Encryption: RSY21 ($i\mathcal{O}$)

Related Work

- (Flexible) Distributed Broadcast Encryption: FN94, WQZD10, BZ14, FWW23, KMW23, GLWW23
- Adaptive Threshold: DP08, HLR10
- Silent Threshold Encryption: RSY21 ($i\mathcal{O}$)
- Silent Threshold Signatures: DCX+23, GJM+24

A Witness Encryption Solution

Can we build Silent Threshold Encryption given WE for NP?

(Extractable) Witness Encryption: Encrypt a message to a statement x.

Can decrypt iff you know w such that $R_L(x, w) = 1$

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i \in [n]}$ on tag"

● = witness ● = statement

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i\in[n]}$ on tag"

● = witness ● = statement

Compile WE → Silent Threshold Encryption:

- **Setup:** Each party sample's pk_i independently

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i\in[n]}$ on tag"

● = witness ● = statement

Compile WE → Silent Threshold Encryption:

- **Setup:** Each party sample's pk_i independently
- **Encrypt:** WE.Encrypt msg with a random string tag

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i \in [n]}$ on tag"

● = witness ● = statement

Compile WE → Silent Threshold Encryption:

- **Setup:** Each party sample's pk_i independently
- **Encrypt:** WE.Encrypt msg with a random string tag
- **Decrypt:** Partial decryptions are signatures on tag.

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i \in [n]}$ on tag"

● = witness ● = statement

Compile WE → Silent Threshold Encryption:

- **Setup:** Each party sample's pk_i independently
- **Encrypt:** WE.Encrypt msg with a random string tag
- **Decrypt:** Partial decryptions are signatures on tag.
- **Aggregate:** Run WE.Decrypt, to recover msg

A Witness Encryption Solution

Suppose we have a WE for the relation:

"I know t signatures from *some* subset of $\{pk_i\}_{i \in [n]}$ on tag"

● = witness ● = statement

BLS signatures

We build a <u>concretely efficient</u> WE for the relation above

What class of relations support <u>efficient</u> WE?

Relations with “Linear” verifiers

Relations with “Linear” verifiers

Express the verification circuit for $R_L(x, w) = 1$ as a set of PPEs.

Relations with “Linear” verifiers

Express the verification circuit for $R_L(x, w) = 1$ as a set of PPEs.

$$\prod e(x_i, x_j) \cdot \prod e(x_i, w_j) \cdot \prod e(w_i, x_j) \cdot \prod e(w_i, w_j) = c_T$$

Relations with "Linear" verifiers

Express the verification circuit for $R_L(x, w) = 1$ as a set of PPEs.

$$\prod e(x_i, x_j) \cdot \prod e(x_i, w_j) \cdot \prod e(w_i, x_j) \cdot \cancel{\prod e(w_i, w_j)} = c_T$$

Relations with “Linear” verifiers

Express the verification circuit for $R_L(x, w) = 1$ as a set of PPEs.

$$\prod e(x_i, x_j) \cdot \prod e(x_i, w_j) \cdot \prod e(w_i, x_j) \cdot \prod e(\cancel{w_i, w_j}) = c_T$$

Compiler [BC16]: Linear PPE → WE

WE for Silent Threshold Encryption

Problem reduced to building linear verifier for:

"I know t signatures from *some* subset of $\{pk_i\}$ on tag"

$\Longrightarrow$ **Silent Threshold Encryption!**

Rest of the talk: building a linear verifier

Our Construction

Recall: BLS Signature

Recall: BLS Signature

- Public Key: $g^{sk} \in \mathbb{G}_1$

Recall: BLS Signature

- Public Key: $g^{sk} \in \mathbb{G}_1$
- Signature: $\sigma = H(m)^{sk} \in \mathbb{G}_2$

Recall: BLS Signature

- Public Key: $g^{sk} \in \mathbb{G}_1$
- Signature: $\sigma = H(m)^{sk} \in \mathbb{G}_2$
- Verification: $e(g, \sigma) = e(pk, H(m))$

Recall: BLS Signature

- Public Key: $g^{sk} \in \mathbb{G}_1$
- Signature: $\sigma = H(m)^{sk} \in \mathbb{G}_2$
- Verification: $e(g, \sigma) = e(pk, H(m))$

Cool, but what's so special?

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Verification:

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

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Verification:

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

Linear verifier for $n = 1$: "I know a signature σ under pk on m"

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Linear Verification: $e(g, \sigma) = e(pk, H(m))$

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Linear Verification: $e(g, \sigma) = e(pk, H(m))$
- Aggregate Verification: Succinctly verify that m signed by **all** $\{pk_i\}_{i \in [n]}$

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Linear Verification: $e(g, \sigma) = e(pk, H(m))$
- Aggregate Verification: Succinctly verify that m signed by **all** $\{pk_i\}_{i \in [n]}$

$$e(g, \prod_{i \in [n]} \sigma_i) = e(\prod_{i \in [n]} pk_i, H(m))$$

Recall: BLS Signature

- Public Key: g^{sk}
- Signature: $\sigma = H(m)^{sk}$
- Linear Verification: $e(g, \sigma) = e(pk, H(m))$
- Aggregate Verification: Succinctly verify that m signed by **all** $\{pk_i\}_{i \in [n]}$

$$e(g, \prod_{i \in [n]} \sigma_i) = e(\prod_{i \in [n]} pk_i, H(m))$$

Linear verifier for $t = n$: "I know a signatures $\{\sigma_i\}$ from **all** $\{pk_i\}_{i \in [n]}$ on m"

n-out-of-n $\rightarrow$ t-out-of-n

n-out-of-n $\rightarrow$ t-out-of-n

"I know t signatures from *some* subset of $\{pk_i\}$ on tag"

High level plan:

n-out-of-n $\rightarrow$ t-out-of-n

"I know t signatures from *some* subset of $\{pk_i\}$ on tag"

High level plan:

1. Aggregate the *subset* of public keys that signed

$$pk_S = \prod_{i \in [n]} pk_i^{b_i}, \text{ where } b_i \in \{0,1\}$$

n-out-of-n → t-out-of-n

"I know t signatures from *some* subset of $\{pk_i\}$ on tag"

High level plan:

1. Aggregate the *subset* of public keys that signed

$$pk_S = \prod_{i \in [n]} pk_i^{b_i}, \text{ where } b_i \in \{0,1\}$$

2. Verify signature under aggregate public key

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

n-out-of-n → t-out-of-n

"I know t signatures from *some* subset of $\{pk_i\}$ on tag"

High level plan:

1. Aggregate the *subset* of public keys that signed

$$pk_S = \prod_{i \in [n]} pk_i^{b_i}, \text{ where } b_i \in \{0,1\}$$

2. Verify signature under aggregate public key

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

Linear

Sumcheck [BCR+19, CNR+22]

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sufficient to show there exist polynomials $Q_1(x)$ and $Q_2(x)$:

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sufficient to show there exist polynomials $Q_1(x)$ and $Q_2(x)$:

$$f(x) \cdot h(x) = s/|H| + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sufficient to show there exist polynomials $Q_1(x)$ and $Q_2(x)$:

$$f(x) \cdot h(x) = s/|H| + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

$$\deg(Q_1(x)) < |H| - 1$$

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sufficient to show there exist polynomials $Q_1(x)$ and $Q_2(x)$:

$$f(x) \cdot h(x) = s/|H| + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

$$\deg(Q_1(x)) < |H| - 1$$

$$\deg(Q_2(x)) < \deg(f(x) \cdot h(x)) - |H|$$

Sumcheck [BCR+19, CNR+22]

Given two polynomials $f(x)$, $h(x)$, prove that $\sum_{i \in H} f(i) \cdot h(i) = s$.

Sufficient to show there exist polynomials $Q_1(x)$ and $Q_2(x)$:

$$\boxed{e(g^{f(\tau)}, g^{h(\tau)}) = e(g^{s/|H|}, g) \cdot e(g^{Q_1(\tau)}, g^{\tau}) \cdot e(g^{Q_2(\tau)}, g^{Z_H(\tau)})}$$

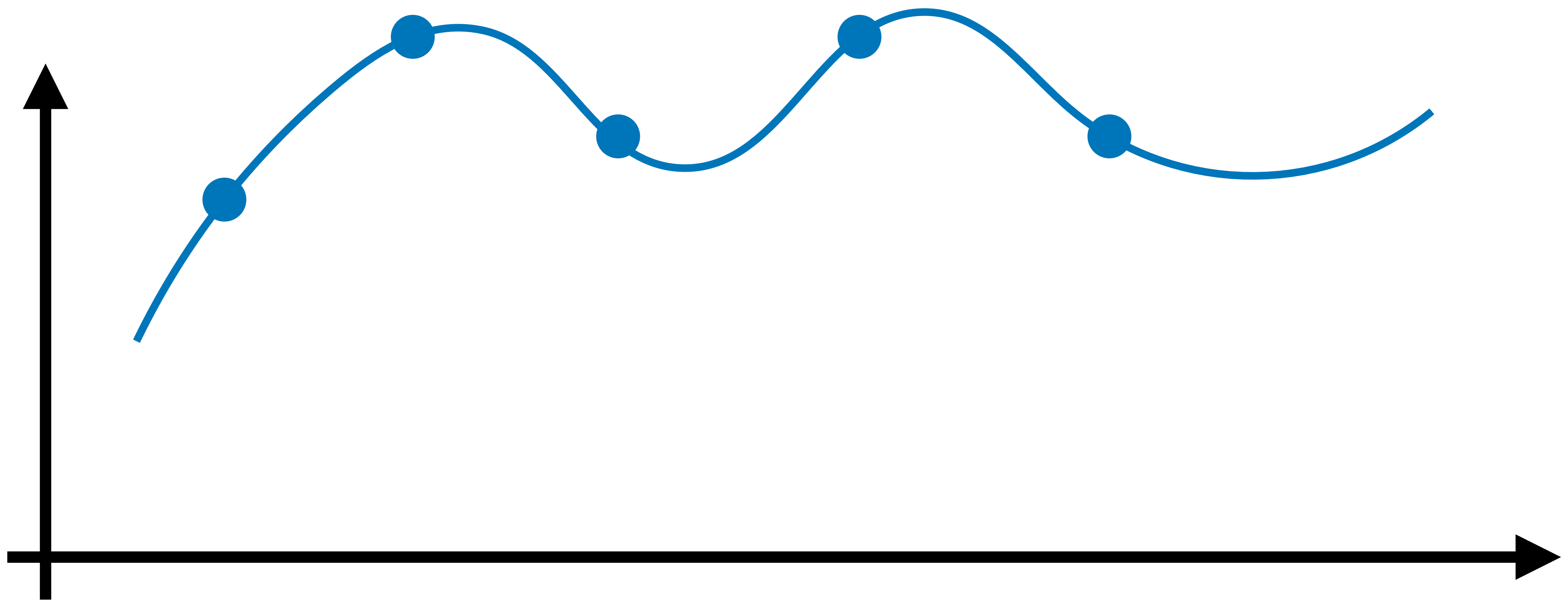
$$\deg(Q_1(x)) < |H| - 1$$

Linear verification given a powers of tau CRS!

$$\deg(Q_2(x)) < \deg(f(x) \cdot h(x)) - |H|$$

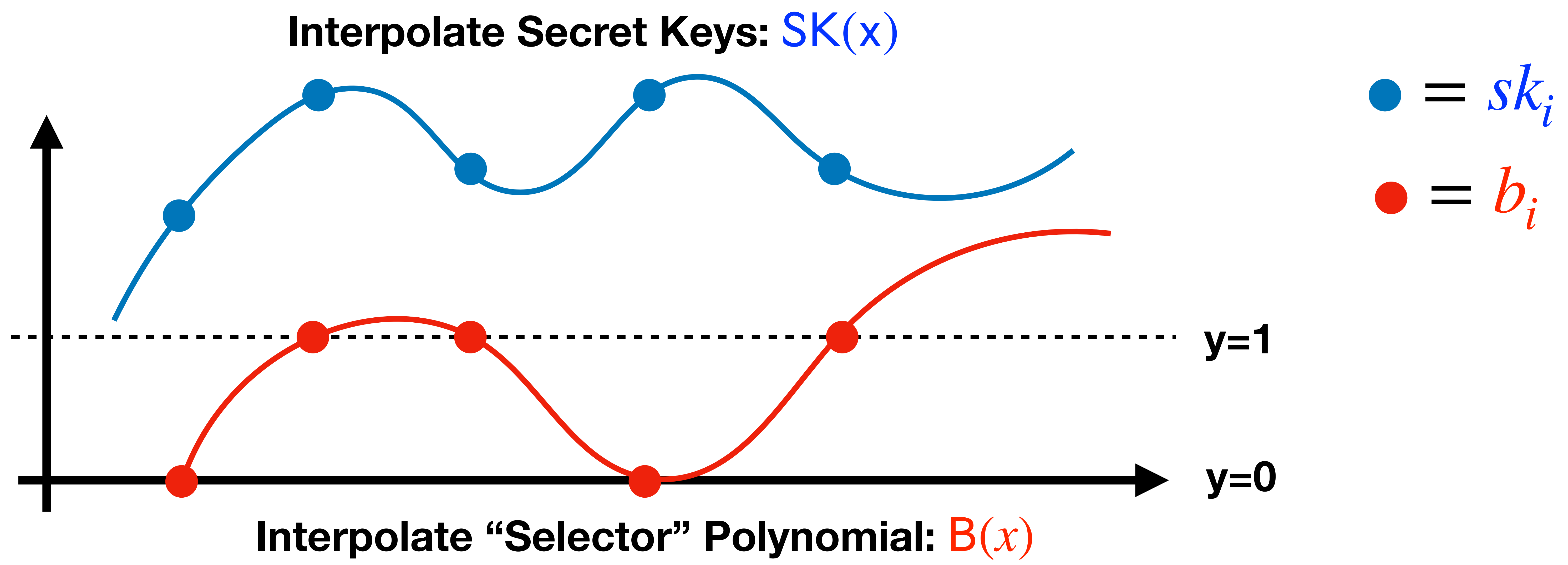
Aggregate via Sumcheck [DCX+23, GJM+24]

Aggregate via Sumcheck [DCX+23, GJM+24]

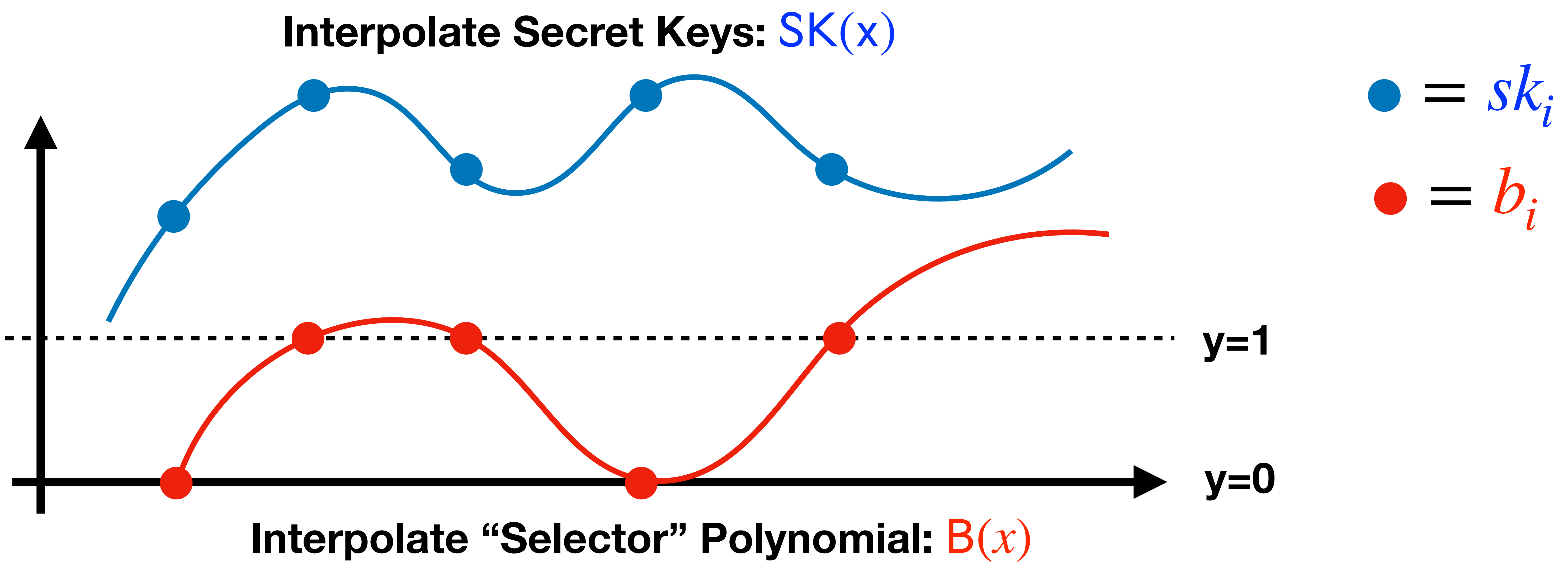


● = sk_i

Aggregate via Sumcheck [DCX+23, GJM+24]

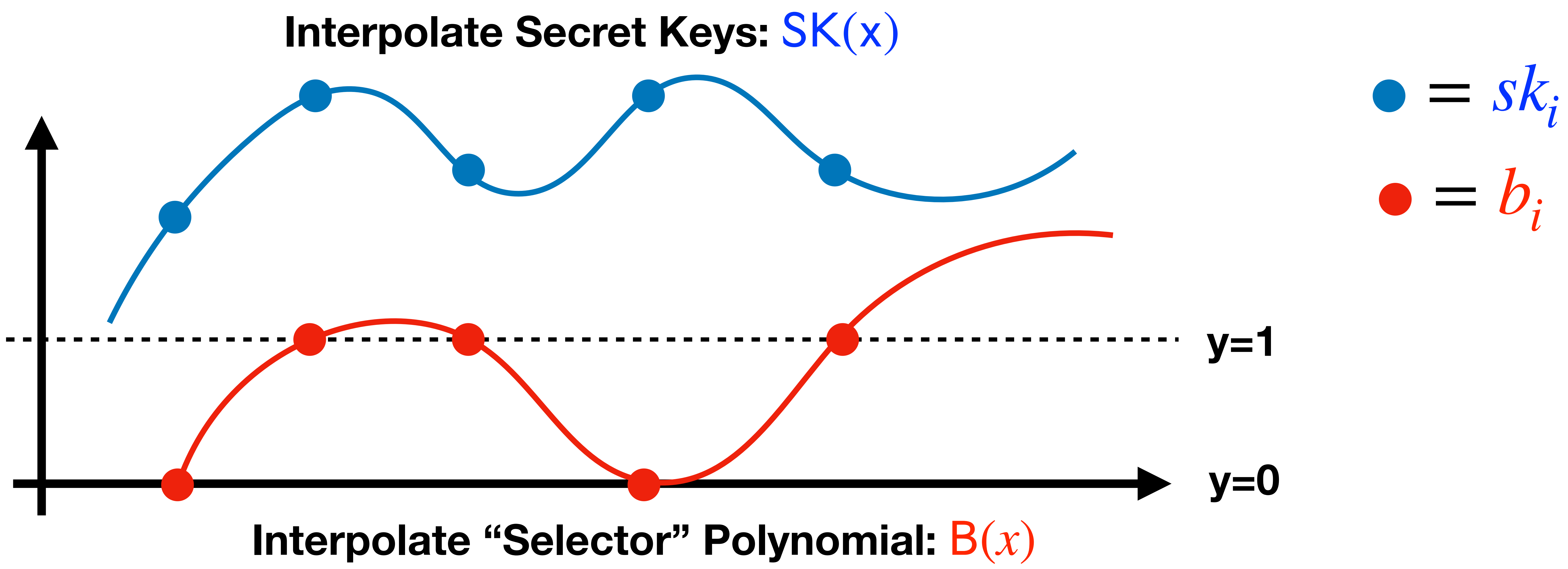


Aggregate via Sumcheck [DCX+23, GJM+24]



Run Sumcheck on SK(x) · B(x)!

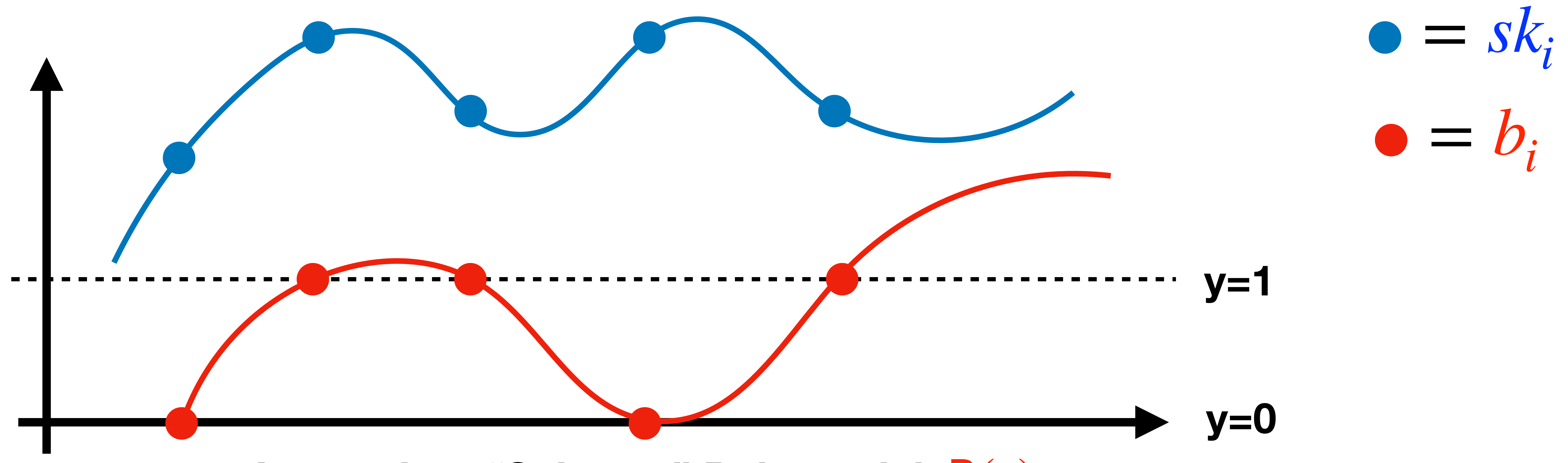
Aggregate via Sumcheck [DCX+23, GJM+24]



Run Sumcheck on $\mathsf{SK}(x) \cdot \mathsf{B}(x)$!

$$\mathsf{SK}(x) \cdot \mathsf{B}(x) = aSK + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

Aggregate via Sumcheck [DCX+23, GJM+24]



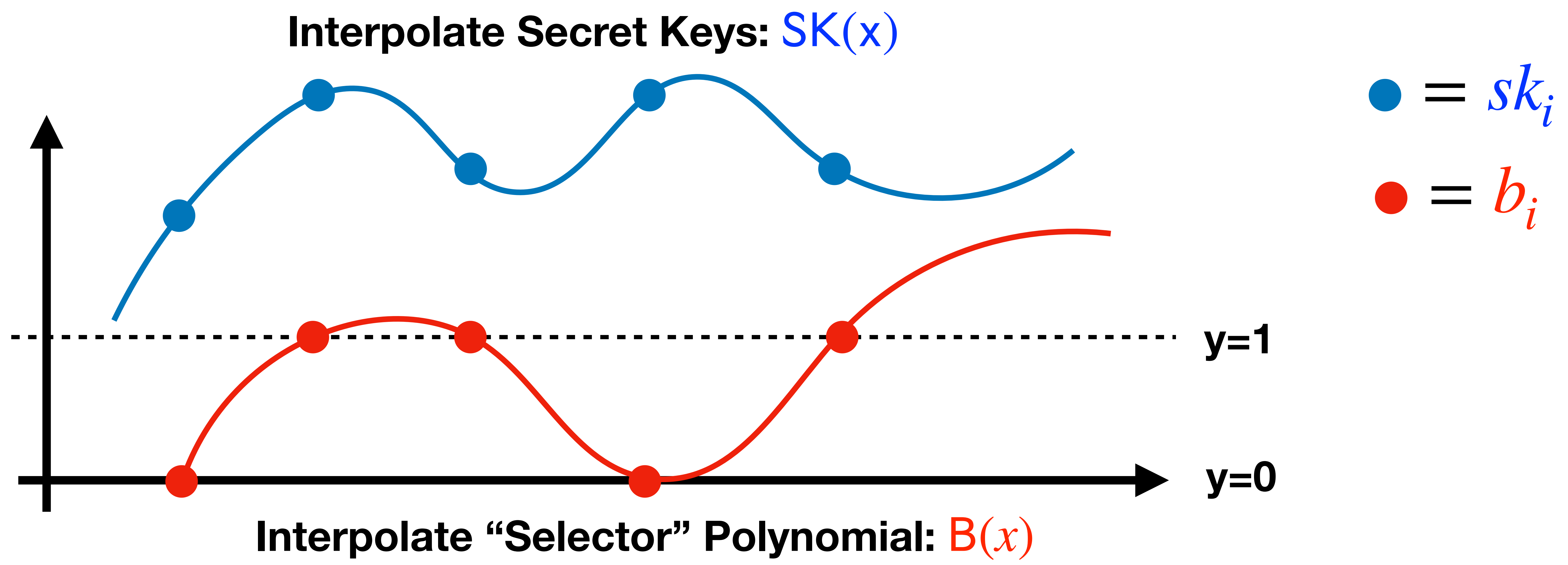
Run Sumcheck on $\mathsf{SK}(x) \cdot \mathsf{B}(x)$!

$$\mathsf{SK}(x) \cdot \mathsf{B}(x) = aSK + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

Check via pairings!

$$e(g^{\mathsf{SK}(\tau)}, g^{\mathsf{B}(\tau)}) = e(PK_S, g) \cdot e(g^{Q_1(\tau)}, g^{\tau}) \cdot e(g^{Q_2(\tau)}, g^{Z_H(\tau)})$$

Aggregate via Sumcheck [DCX+23, GJM+24]



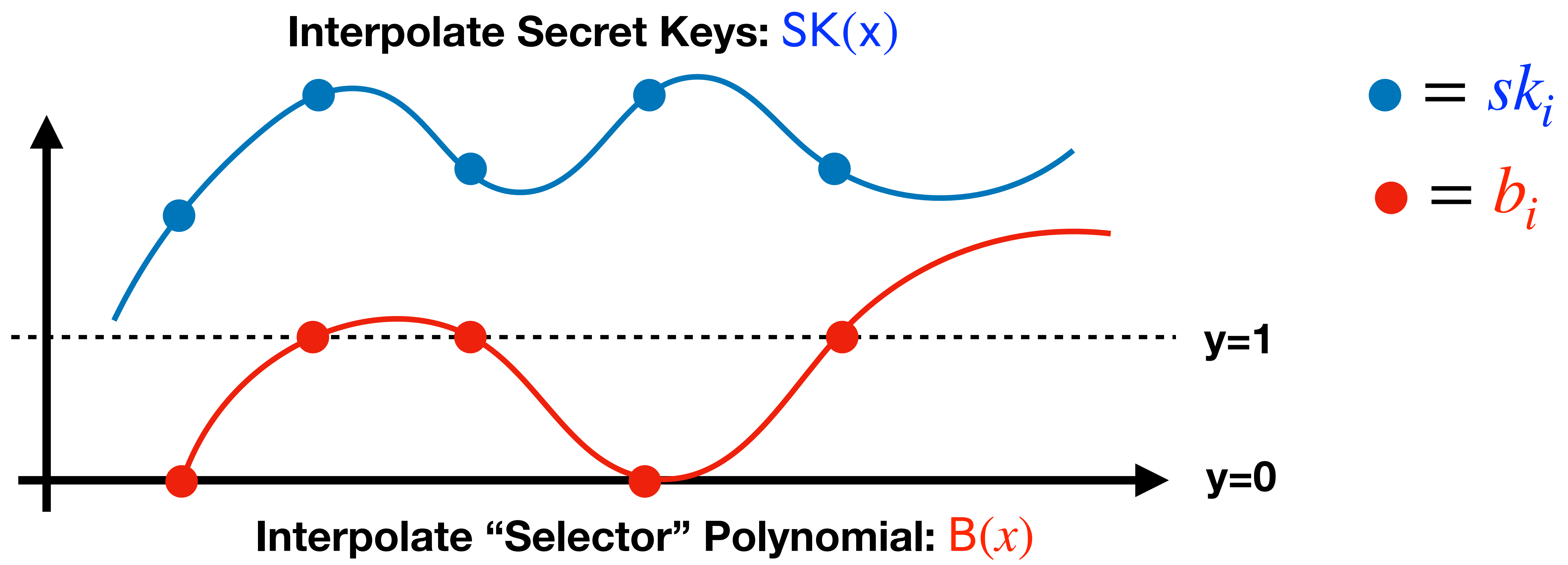
Run Sumcheck on $\mathsf{SK}(x) \cdot \mathsf{B}(x)$!

$$\mathsf{SK}(x) \cdot \mathsf{B}(x) = aSK + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

Check via pairings!

$$e(g^{\mathsf{SK}(\tau)}, g^{\mathsf{B}(\tau)}) = e(\boxed{PK_S}, g) \cdot e(g^{Q_1(\tau)}, g^{\tau}) \cdot e(g^{Q_2(\tau)}, g^{Z_H(\tau)})$$

Aggregate via Sumcheck [DCX+23, GJM+24]



Run Sumcheck on $\mathsf{SK}(x) \cdot \mathsf{B}(x)$!

$$\mathsf{SK}(x) \cdot \mathsf{B}(x) = aSK + Q_1(x) \cdot x + Q_2(x) \cdot Z_H(x)$$

Check via pairings!

$$e(g^{\mathsf{SK}(\tau)}, g^{\mathsf{B}(\tau)}) = e(\boxed{PK_S}, g) \cdot e(g^{Q_1(\tau)}, g^{\tau}) \cdot e(g^{Q_2(\tau)}, g^{Z_H(\tau)})$$

Note: Doing this in the "exponent" need an additional $O(n)$ terms $(g^{sk_i}, g^{sk_i \cdot \tau}, g^{sk_i \cdot \tau^2}, \ldots, g^{sk_i \cdot \tau^n})$

n-out-of-n → t-out-of-n

"I know t signatures from any subset of n parties on tag"

$$\boxed{pk_S = \prod_{i \in [n]} pk_i^{b_i}} \longrightarrow \text{Univariate Sumcheck [DCX+23, GJM+24]}$$

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

n-out-of-n → t-out-of-n

"I know t signatures from any subset of n parties on tag"

$$pk_S = \prod_{i \in [n]} pk_i^{b_i}$$

→ Univariate Sumcheck [DCX+23, GJM+24]

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

Linear

n-out-of-n → t-out-of-n

"I know t signatures from any subset of n parties on tag"

$$\boxed{pk_S = \prod_{i \in [n]} pk_i^{b_i}}$$

→ Univariate Sumcheck [DCX+23, GJM+24]

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

Linear

$$|\{b_i \mid b_i \neq 0\}| \geq t$$

n-out-of-n → t-out-of-n

"I know t signatures from any subset of n parties on tag"

$$pk_S = \prod_{i \in [n]} pk_i^{b_i}$$ → Univariate Sumcheck [DCX+23, GJM+24]

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

Linear

$$|\{b_i \mid b_i \neq 0\}| \geq t$$ → Can be reduced to a degree check (linear 😃)

n-out-of-n → t-out-of-n

"I know t signatures from any subset of n parties on tag"

$$\boxed{pk_S = \prod_{i \in [n]} pk_i^{b_i}}$$ → Univariate Sumcheck [DCX+23, GJM+24]

$$e(g, \sigma_S) = e(pk_S, H(tag))$$

Linear

$$\boxed{|\{b_i \mid b_i \neq 0\}| \geq t}$$ → Can be reduced to a degree check (linear 😃)

Compile to a witness encryption 😃🍾

Evaluation

Evaluation (512 Parties)

KeyGen [One-Time]: 125 ms (24 KB in size)

Encrypt: 7 ms, 768 bytes.

Partial Decrypt: 2.5 ms, 96 bytes

Reconstruct: 130 ms

https://github.com/guruvamsi-policharla/silent-threshold-encryption

Thank you!

Paper: ia.cr/2024/263

Blogpost: