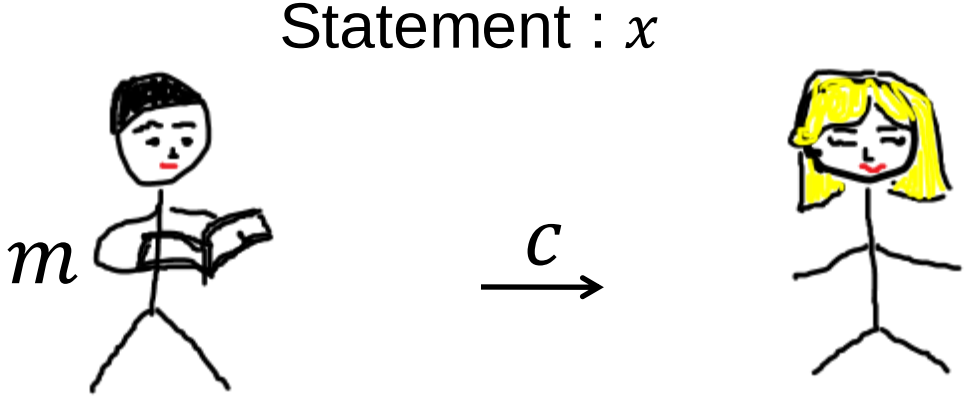


Witness Encryption: From Theory to Practice

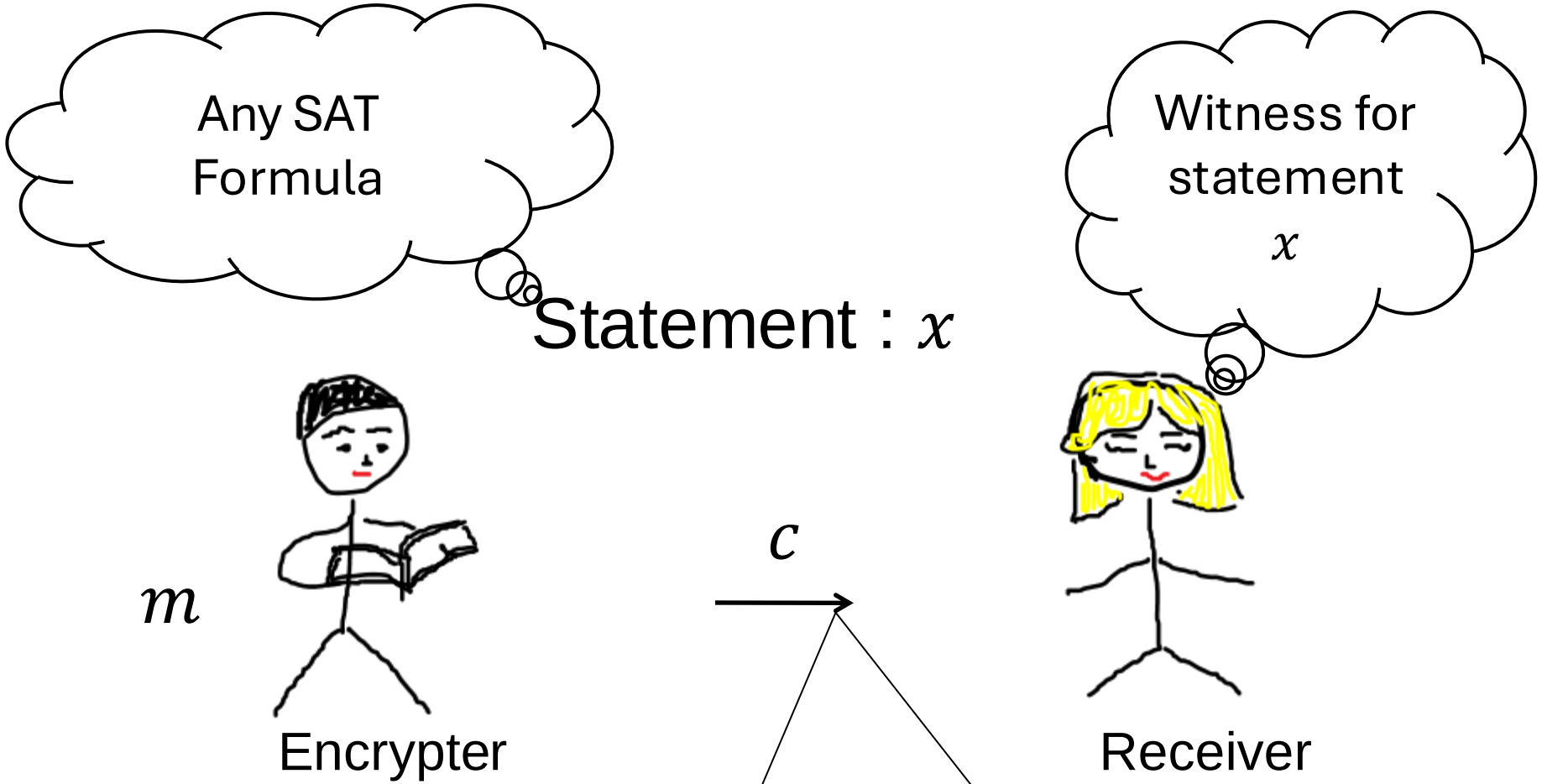
Sanjam Garg

University of California, Berkeley

Presented at NIST-STPPA7 on 2025-Jan-16

Witness Encryption (WE)

[TW87, Rudich89, IOS97, IS91, KMV07, CS02, CCKV08, GOVW12 …]

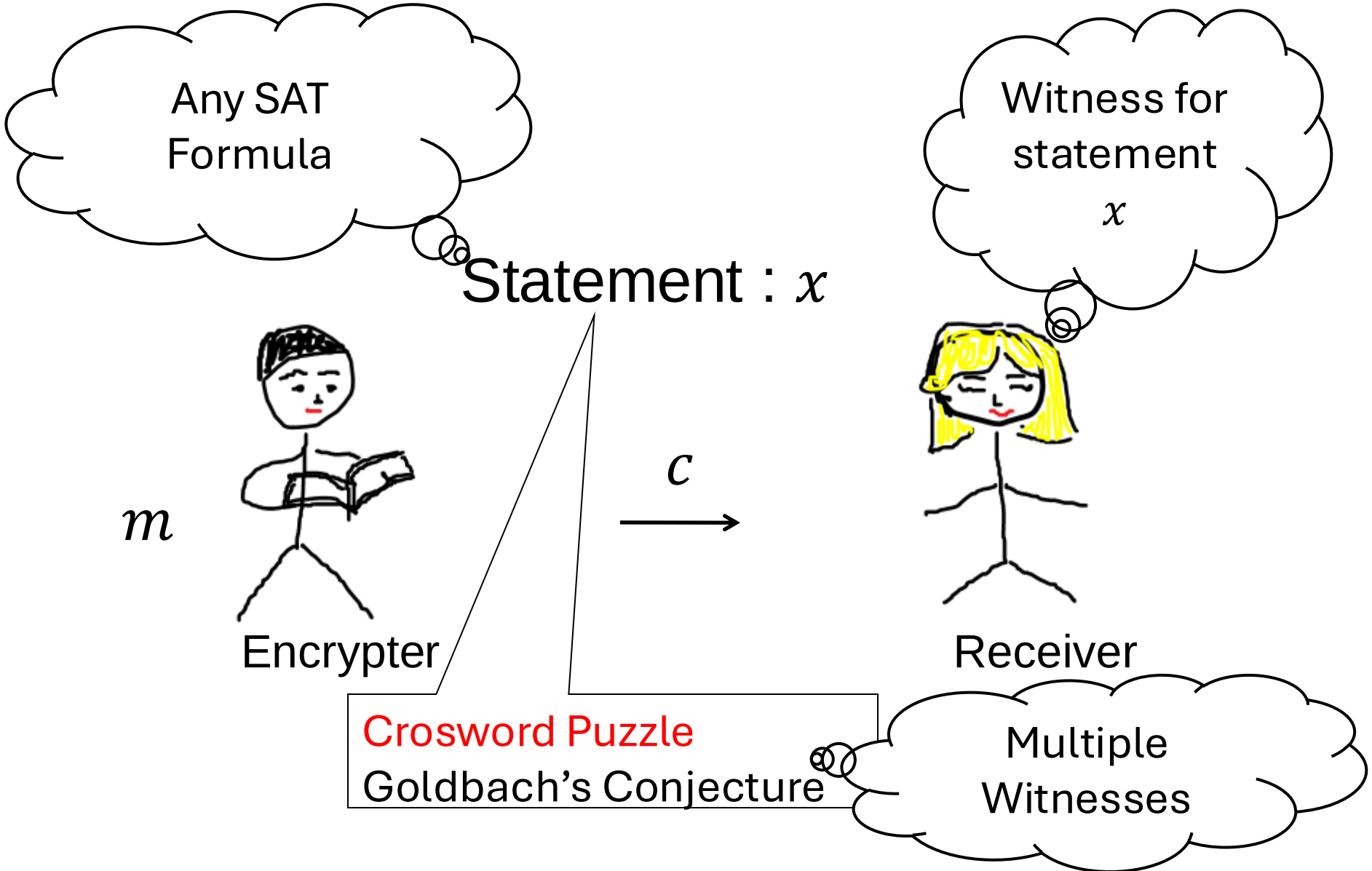


Soundness:

Statement is false $\Longrightarrow$ Semantic Security

Witness Encryption (WE) Examples

[TW87, Rudich89, IOS97, IS91, KMV07, CS02, CCKV08, GOVW12 …]



Feasibility

- We didn't know if the notion was realizable.
- In STOC 2013 [GGSW13] we showed a general purpose WE scheme.
  - Need strong assumption on multilinear maps [GGH13]

Early Applications of Witness Encryption

- PKE with super efficient KeyGen
  - PK is the output of a PRG and encryption is done so that message can be recovered using the seed
- IBE [BF01], ABE [SW05,GVW13] for circuits
- Resettable Statistical Zero-Knowledge [CGGM00,GOVW12] with efficient prover for all of NP
- ABE for TMs, and more [GKPVZ13]

Practicality

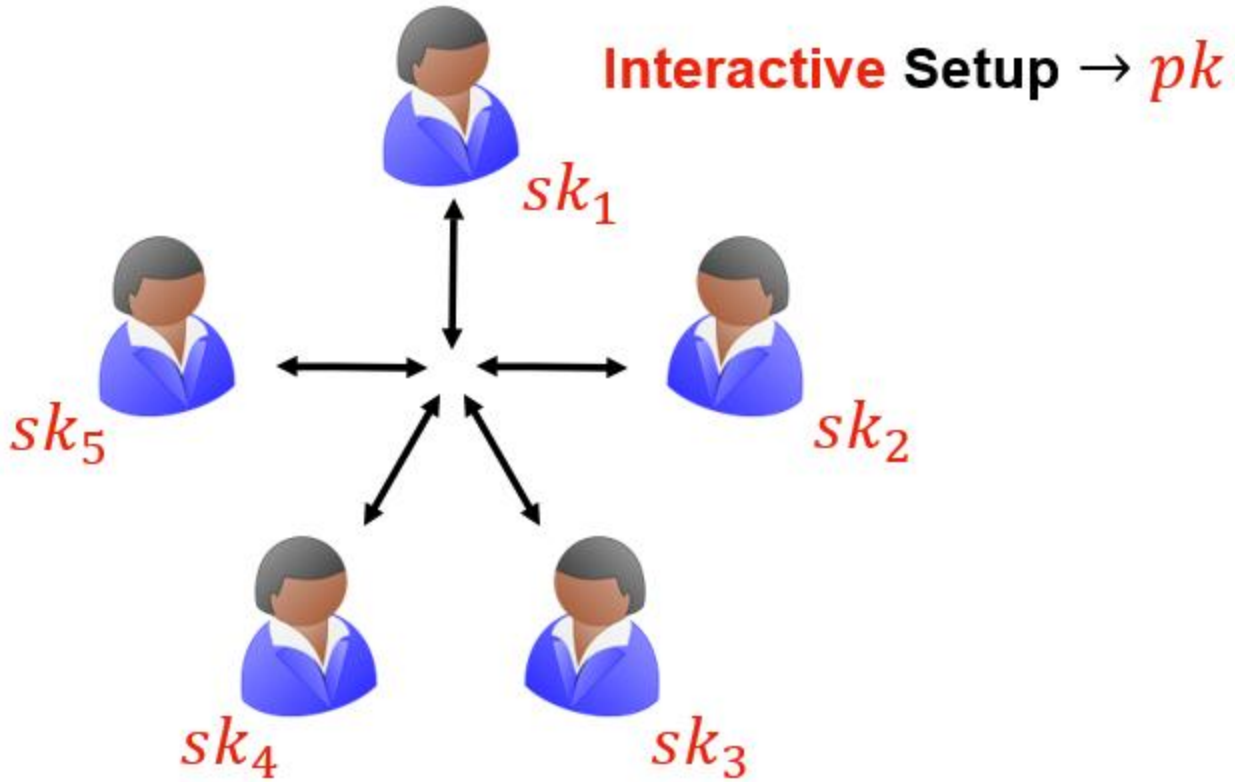
- Still more work remains towards realizing practical general-purpose WE

- Special purpose WE already quite useful
  - Hash proof systems [CS00]
  - Laconic OT[CDGGMP17], Hash Encryption/IBE [DG17]
  - Two round MPC [GGHR14, GS17, GIS18…]
  - Threshold Encryption [GKPW24] with applications to mempool privacy [CGPP24, CGPW24, BFOQ24, AFP24]

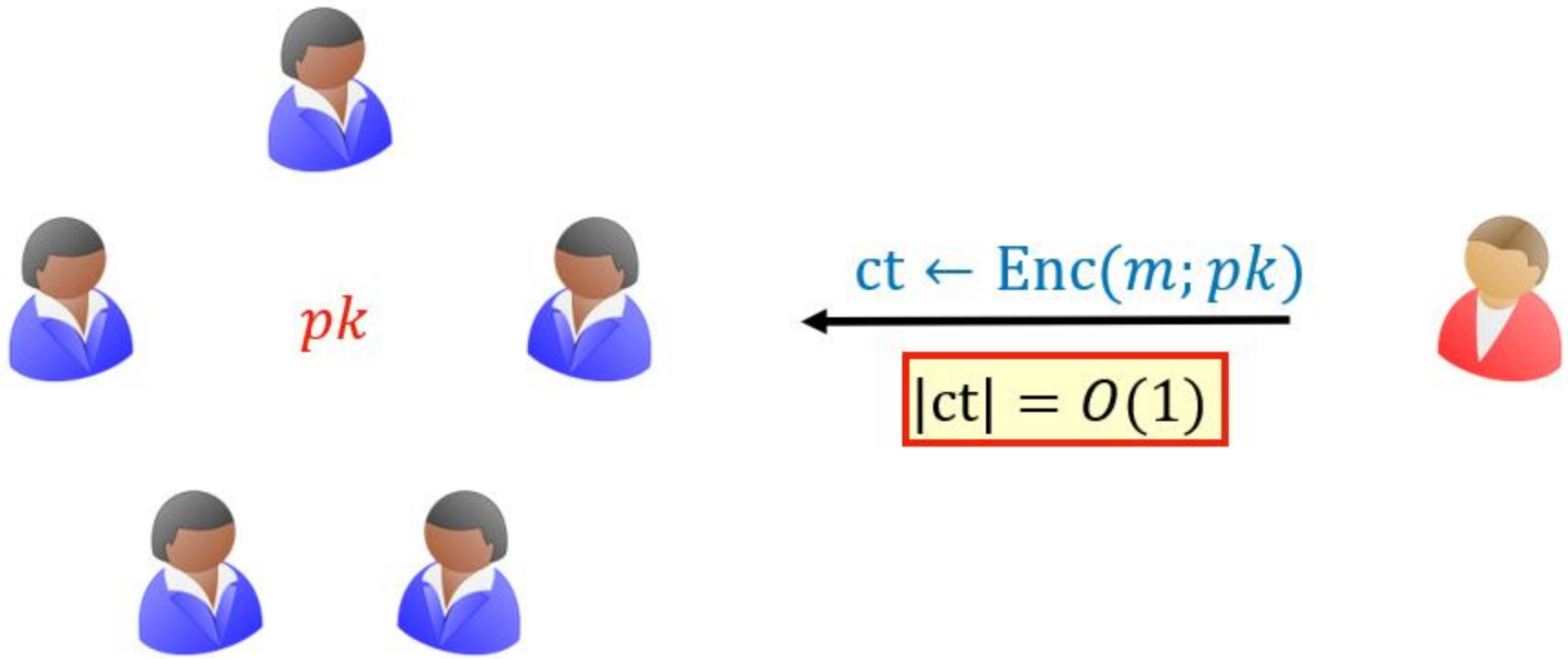
Threshold Encryption

- **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 < *t* of parties
- Ciphertext Size does not grow with *t, n*
- Non-interactive decryption

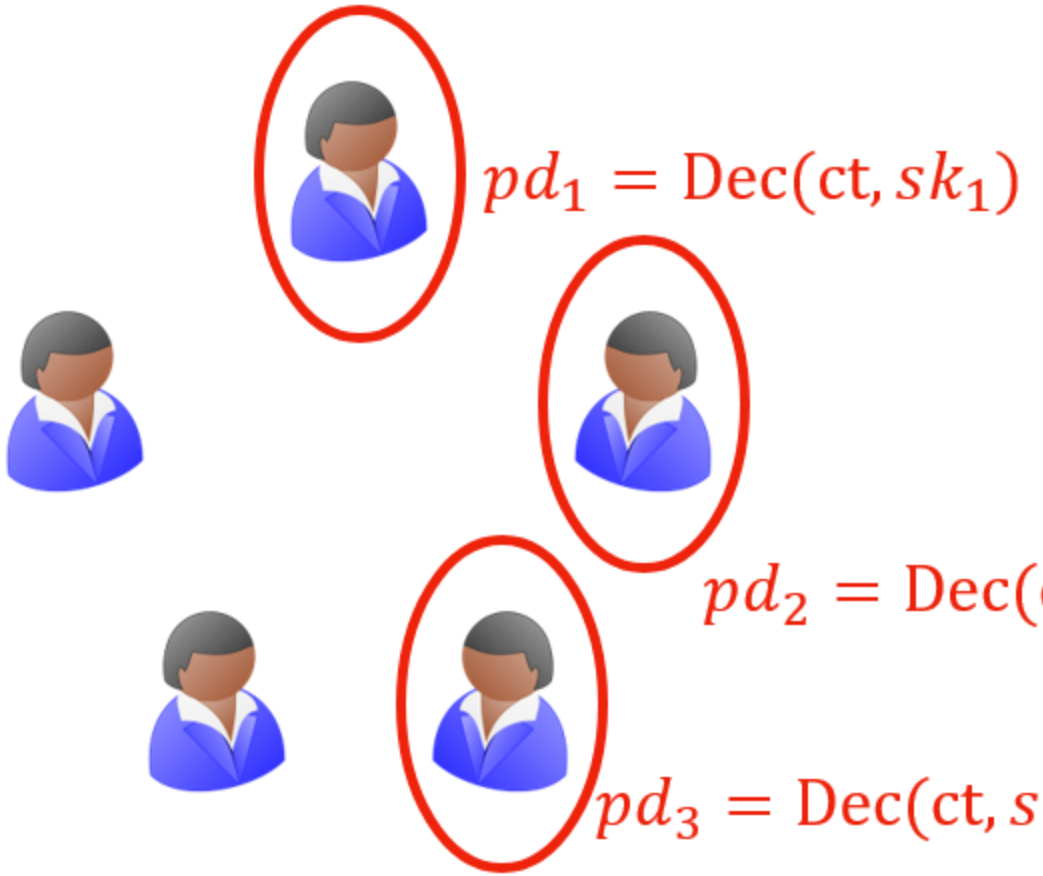
Threshold Encryption - DKG



Threshold Encryption - Encryption



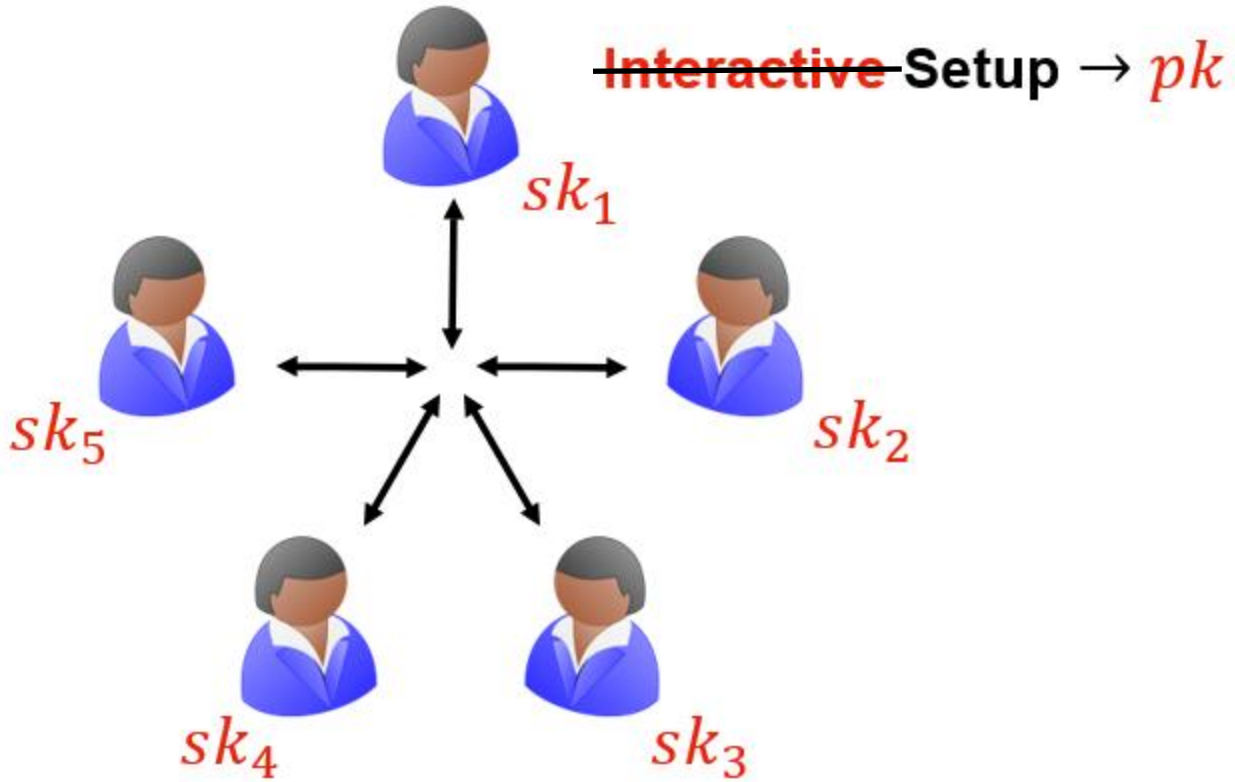
Threshold Encryption - Decryption



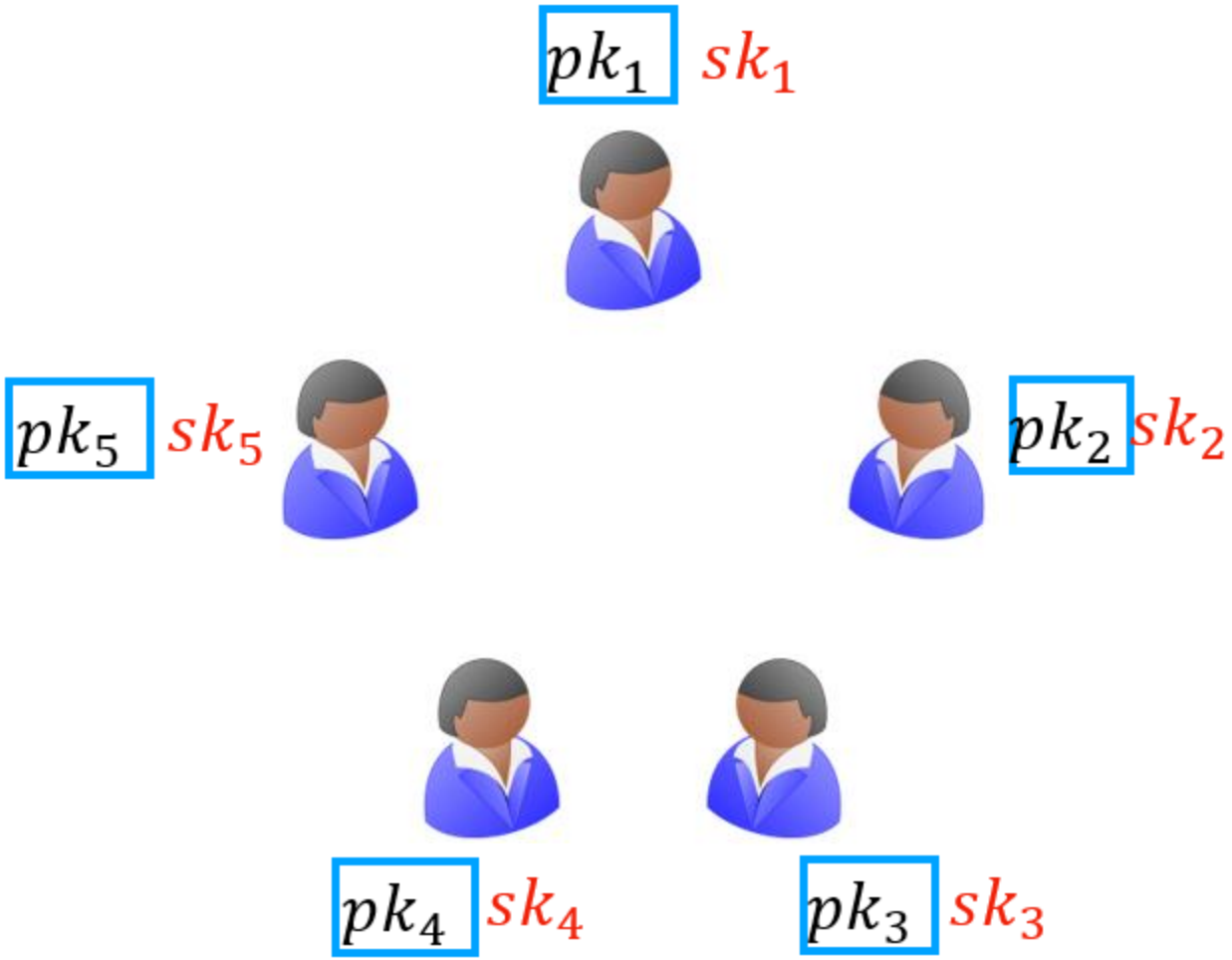
Deterministic Function:

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

Today: Silent Threshold Encryption



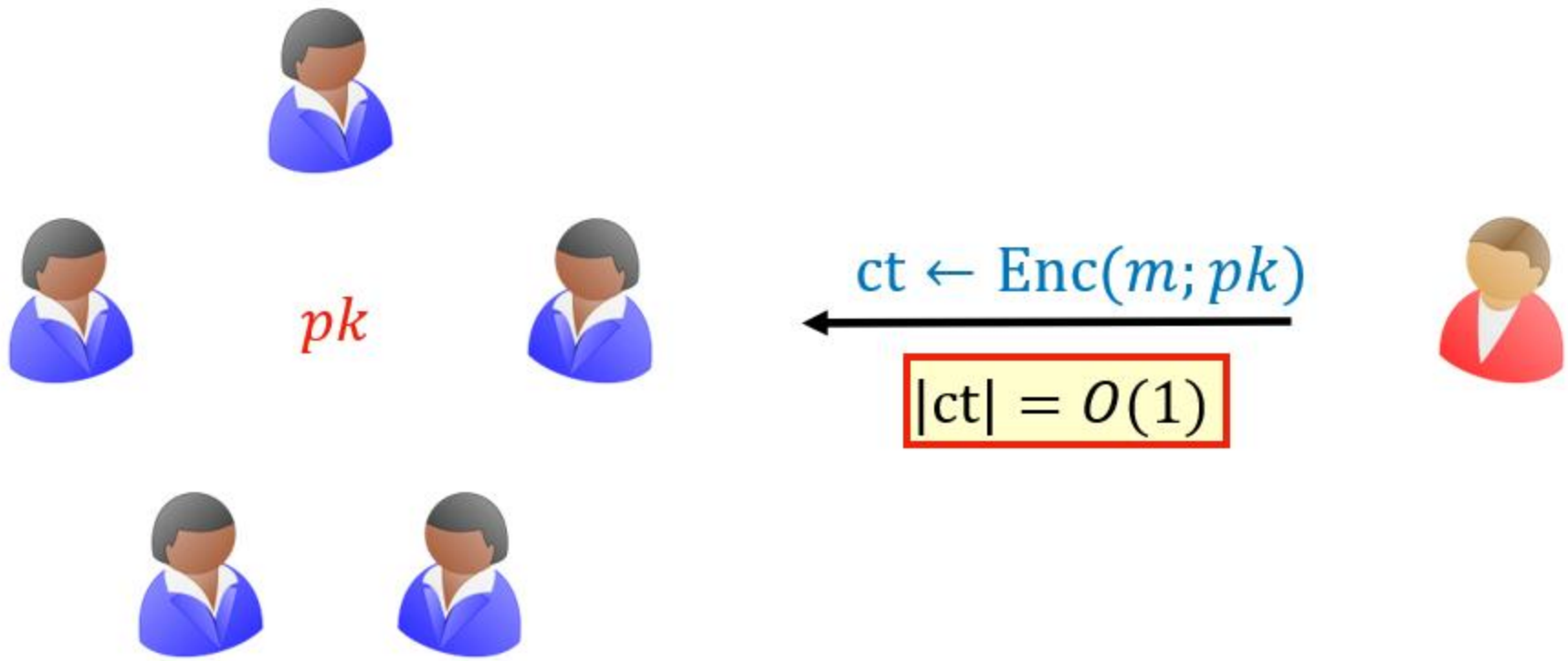
Silent Threshold Encryption – Silent Setup



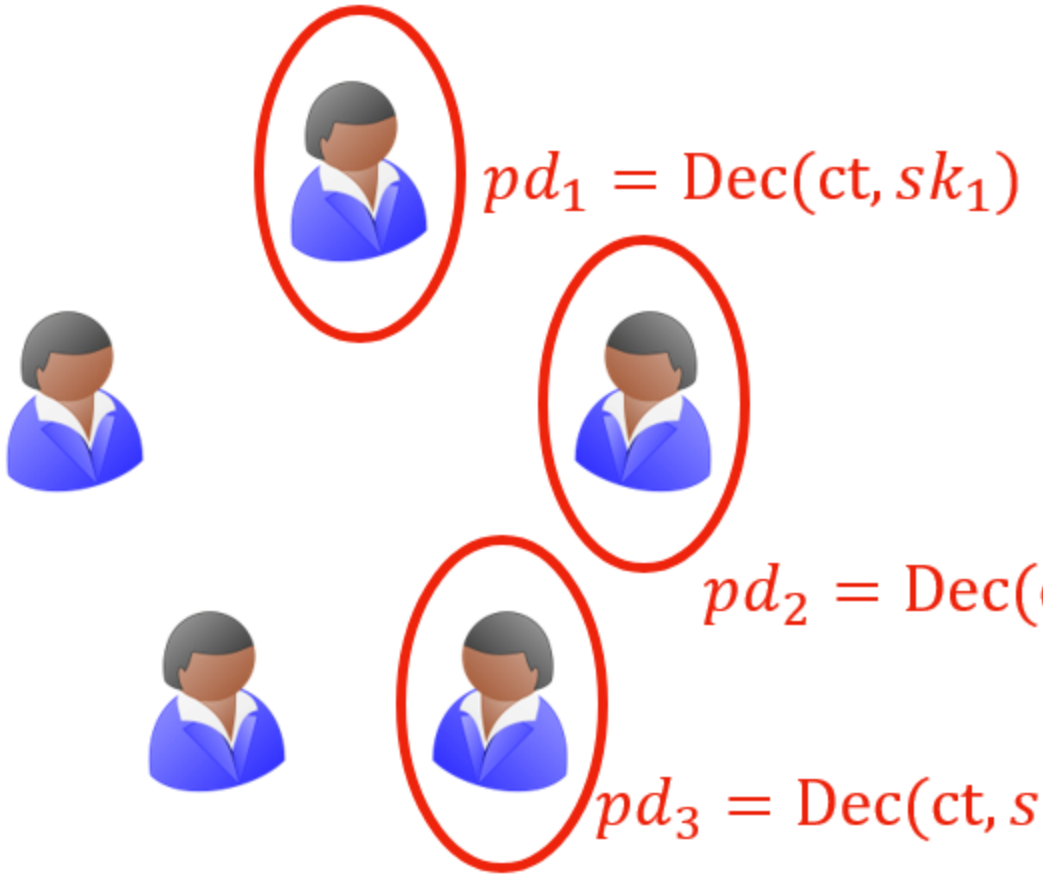
Deterministic Function:

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

Silent Threshold Encryption - Encryption



Silent Threshold Encryption - Decryption



Deterministic Function:

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

Textbook Solution

Threshold ElGamal seems quite nice:

- Short public key: $(\mathrm{g},\ \mathrm{h} = \mathrm{g}^{x})$
- Short secret key: $[\mathrm{x}] = (\mathrm{x}_1, \ldots x_n)$
- Short ciphertexts: $(\mathrm{g}^{r}, h^{r} \cdot M)$
- Short partial decryption: $\mathrm{g}^{r \cdot x_i}$

… but

Silent Setup important in applications where the committee changes frequently

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

Comparsion

Threshold ElGamal:

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

Silent Threshold:

- Long 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

Prior 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: DCXNBR23, GJMSWZ24

Application in Mempool Privacy

- Blockchains: Cleartext Transactions submitted to Mempool
  - Delay before they make it to chain
- Allows for attacks such as a front running
- Alternatively: Submit encrypted transactions
- Decrypted by a committee that changes with each block

How do we do it?

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"

● = *statement*
● = *witness*

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

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"

● = *statement*

● = *witness*

BLS signatures

We build a concretely efficient WE for the relation above

Relations with *Linear* Verifiers

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

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

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

Recall: BLS Signatures

- 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?

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

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

Recall: Aggregate BLS

- 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

"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

Sum-check [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 $Q_2(x)$:

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

$$\deg(Q_1(x)) < |H| - 1$$
$$\deg(Q_2(x)) < \deg(f(x) \cdot h(x)) - |H|$$

Sum-check [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 $Q_2(x)$:

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

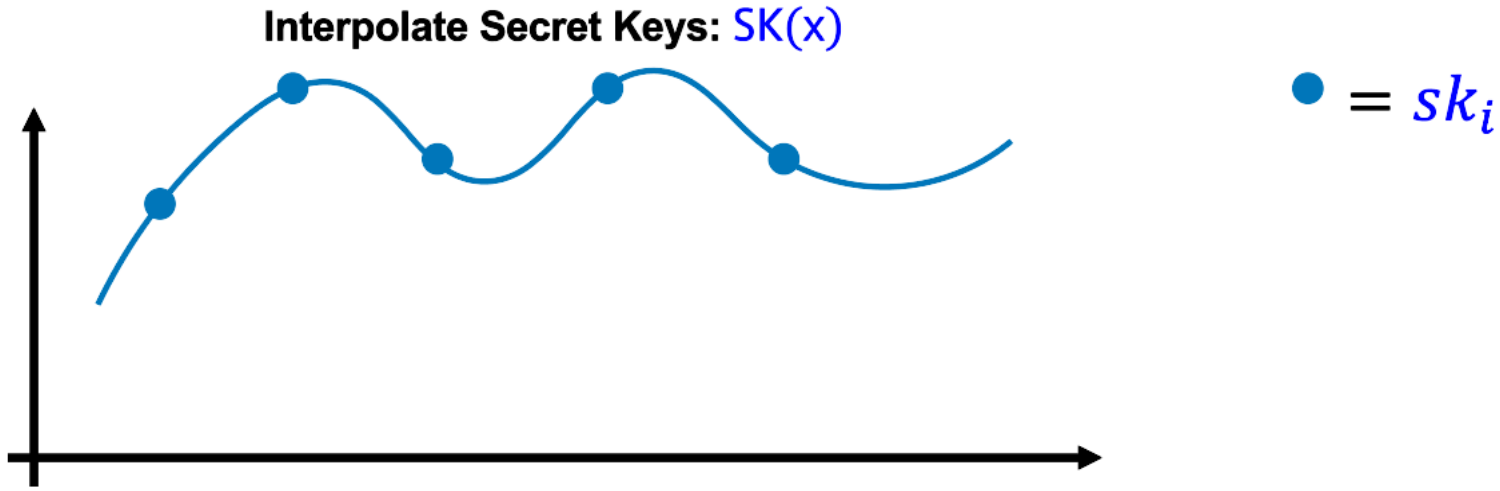
$$\deg(Q_1(x)) < |H| - 1$$
$$\deg(Q_2(x)) < \deg(f(x) \cdot h(x)) - |H|$$

Linear

Linear verification given a powers of tau setup.

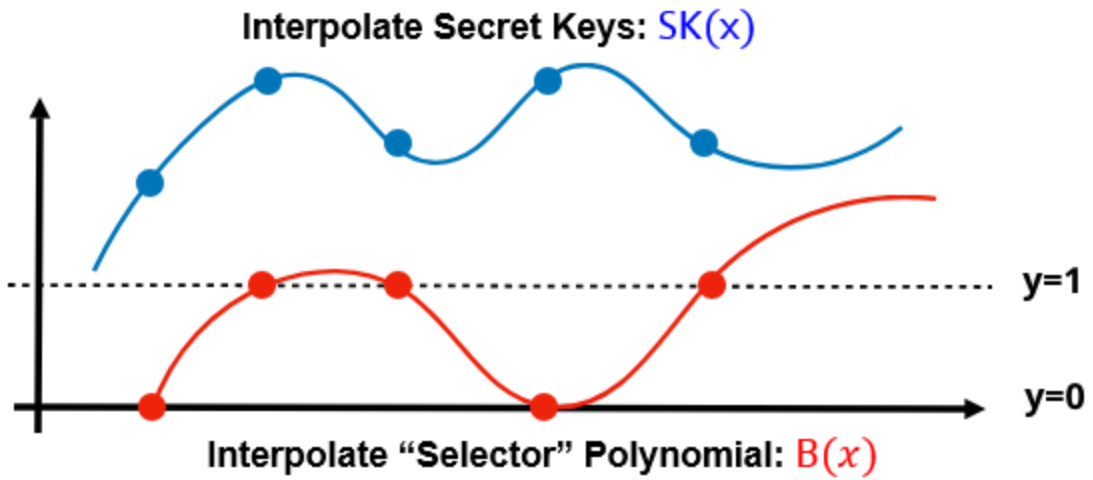
Our Scheme

- CRS = Powers of tau
- *i-th* party publishes its public key as $(g^{sk_i}, g^{sk_i \cdot \tau}, g^{sk_i \cdot \tau^2}, \ldots, g^{sk_i \cdot \tau^n})$



- Public key of group is $g^{SK(\tau)}$

Aggregating a subset of the keys



$\bullet = sk_i$

$\bullet = b_i$

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

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

Check via pairings!

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

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

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!