

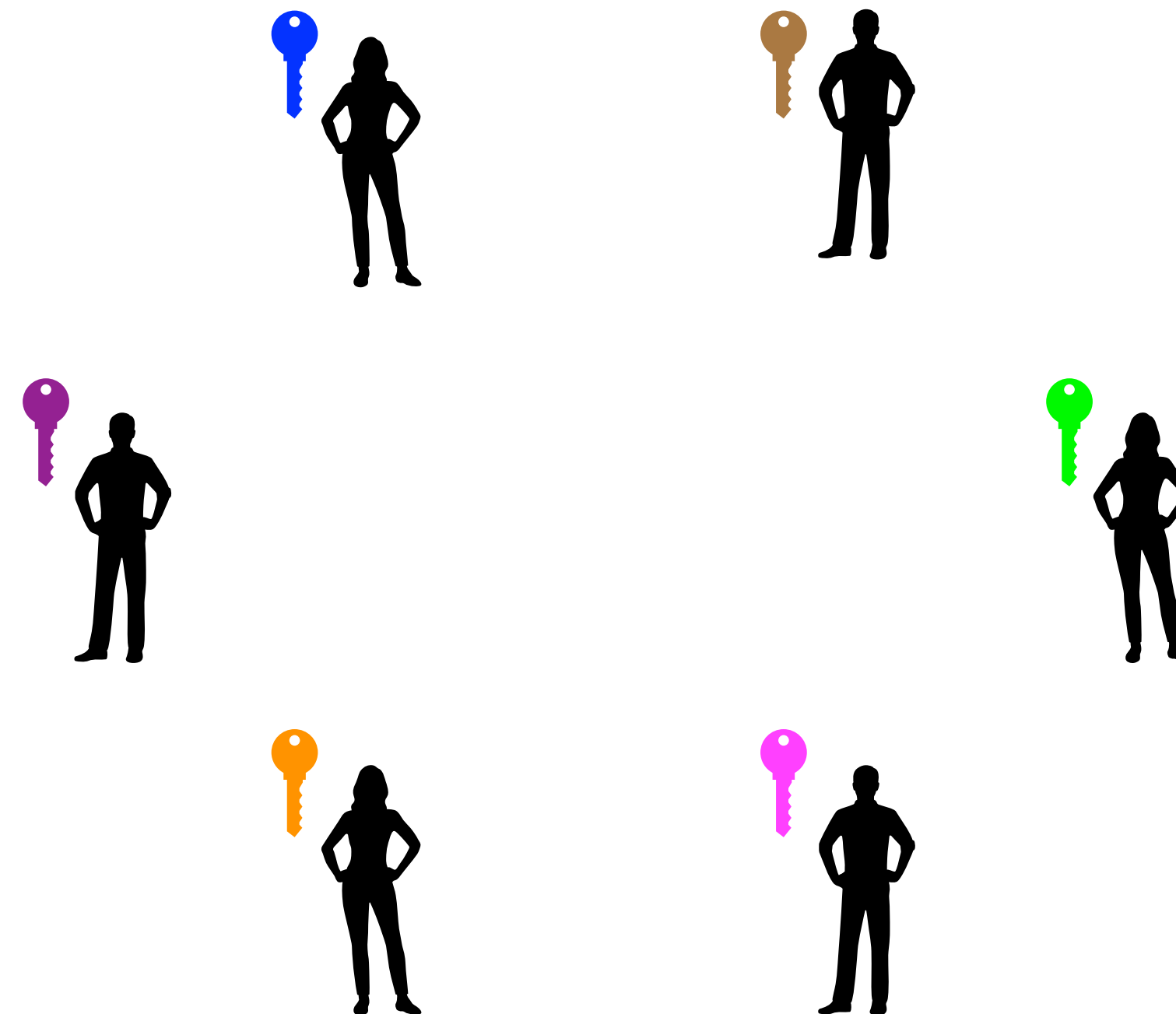
# Amber: Lattice-Based Threshold KEM with Active Security

Sasha Lapiha @ Silence Labs 19 Jan 2026

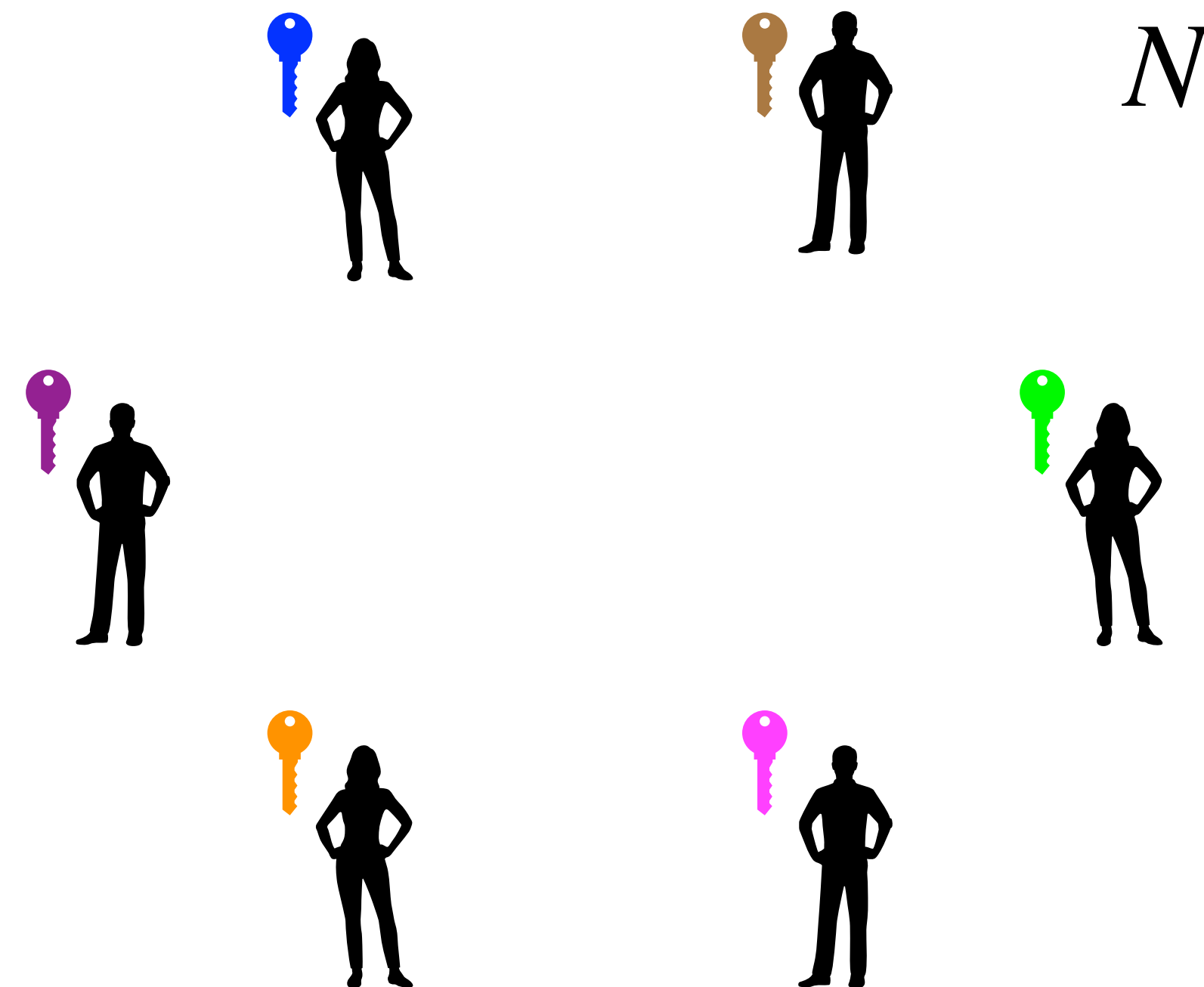


\*joint work with Katharina Boudgoust, Rafael del Pino and Thomas Prest

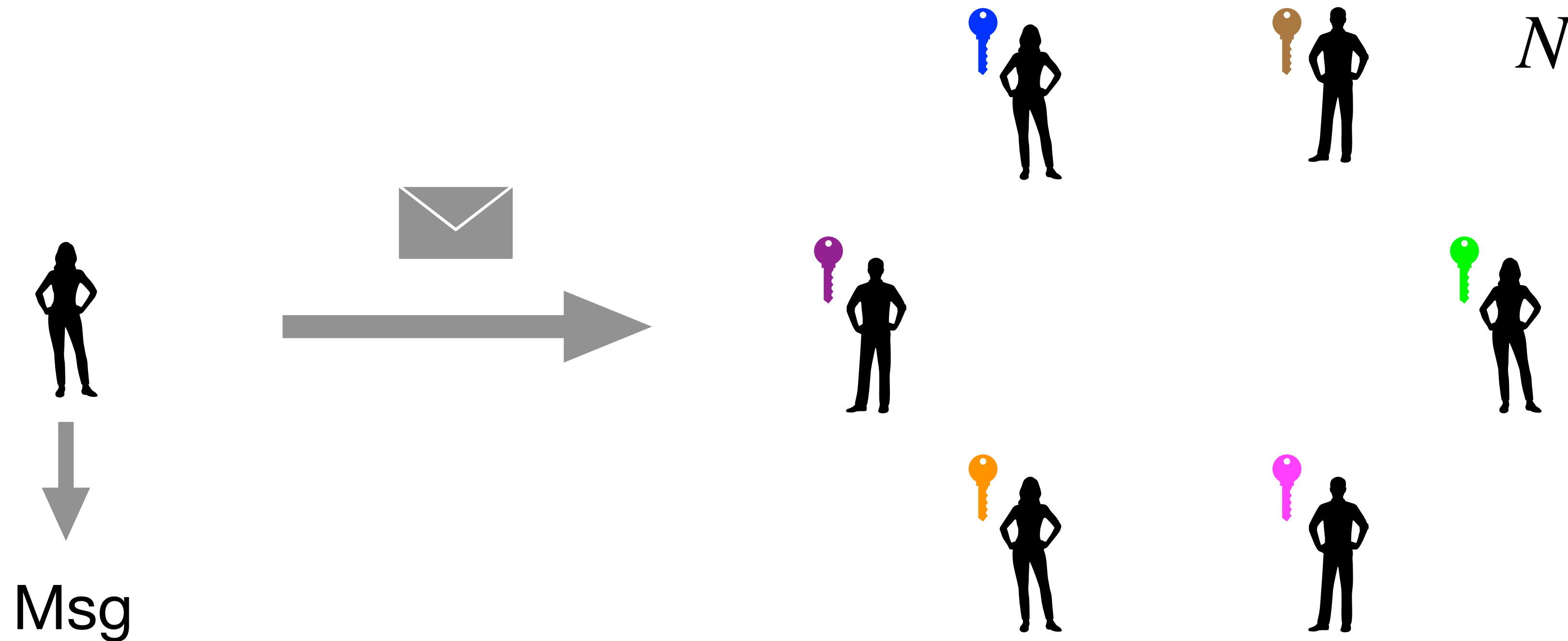
# Threshold Cryptography



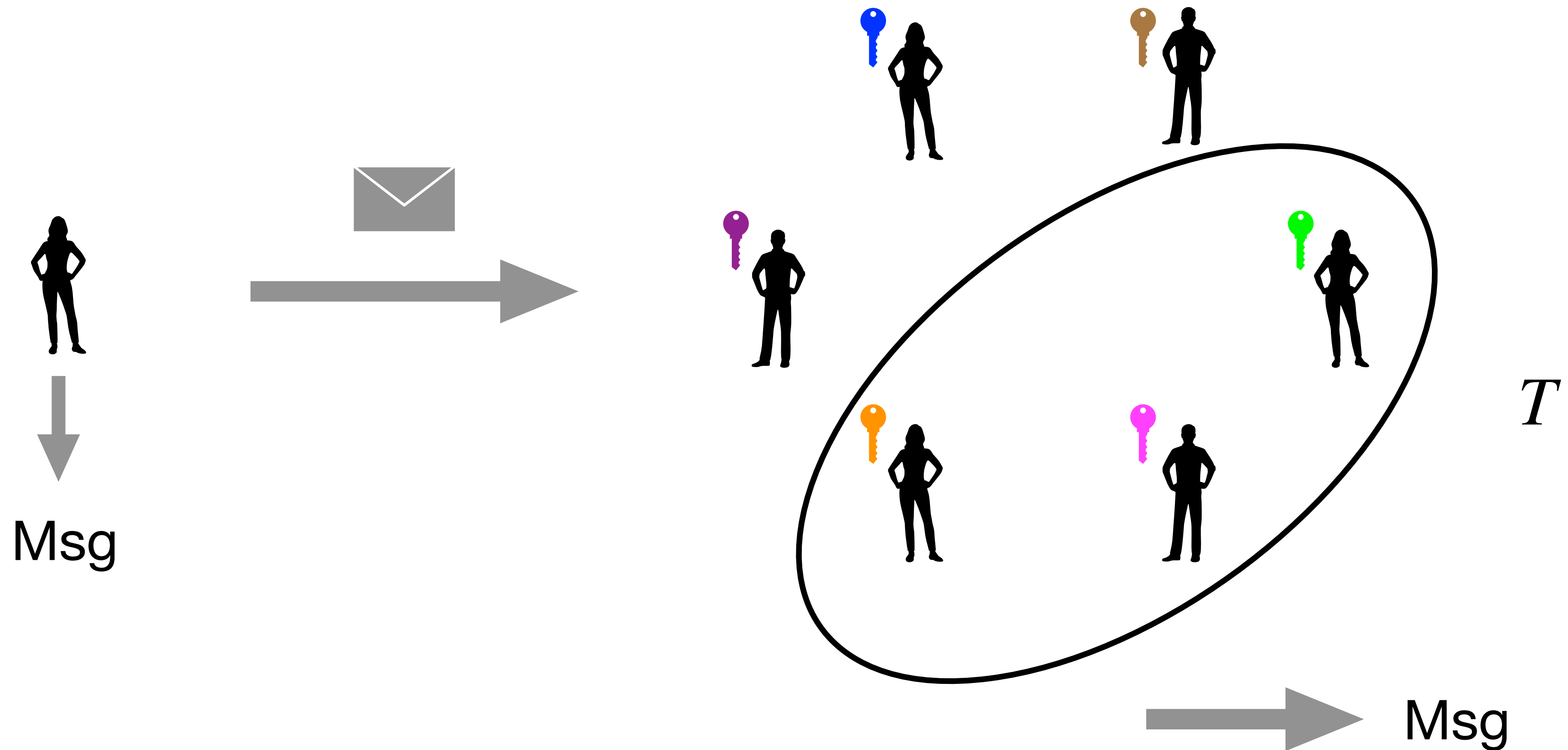
# Threshold Decryption



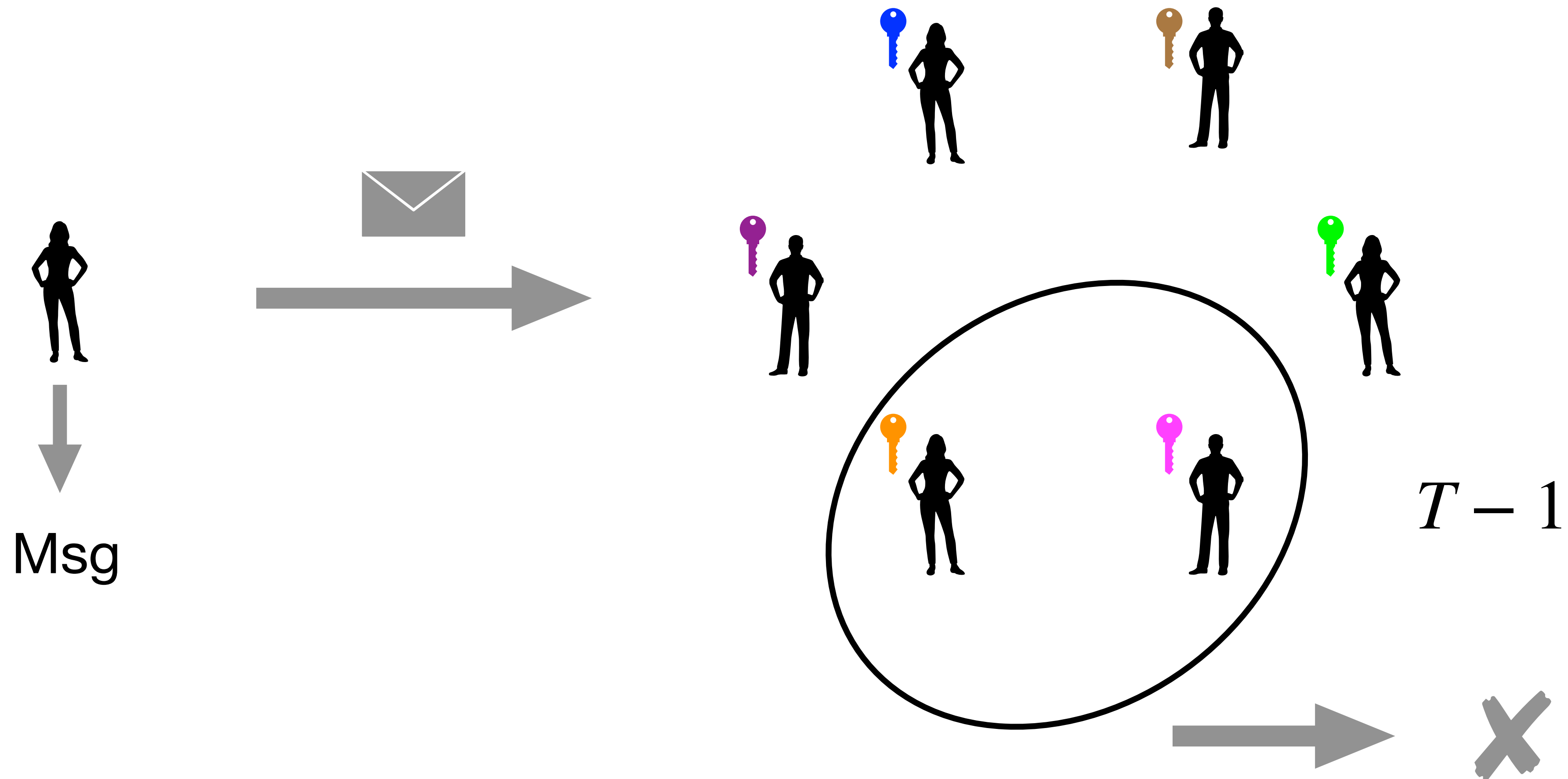
# Threshold Decryption



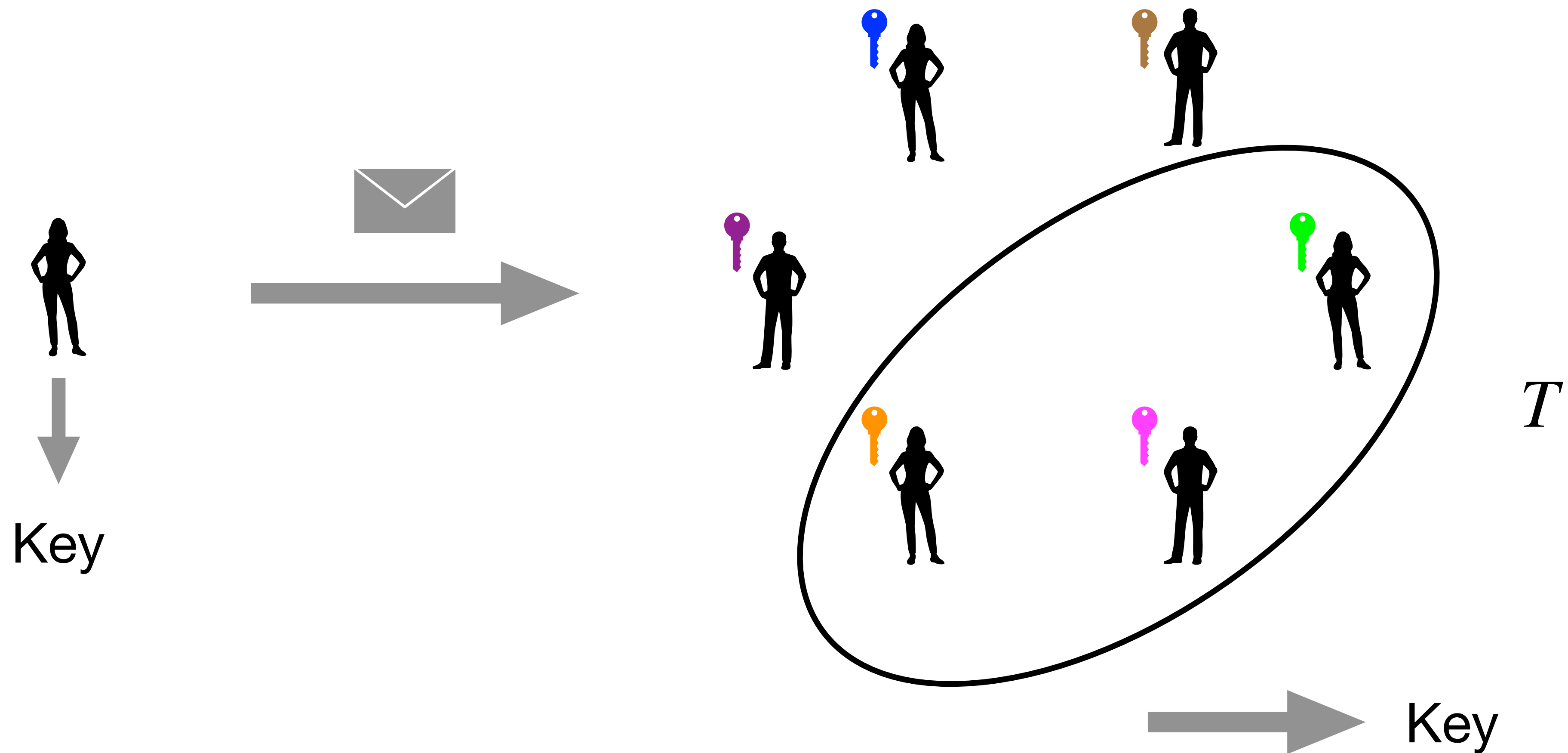
# Threshold Decryption



# Threshold Decryption

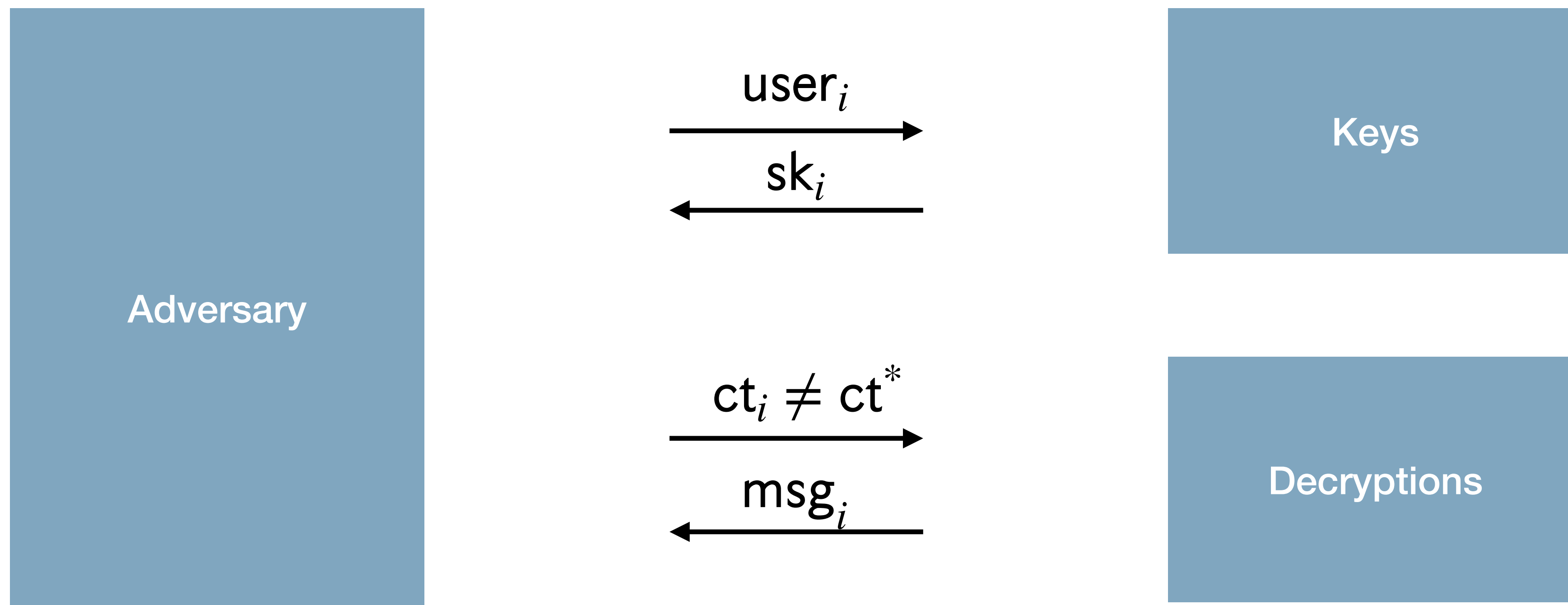


# Threshold Decapsulation (TKEM)



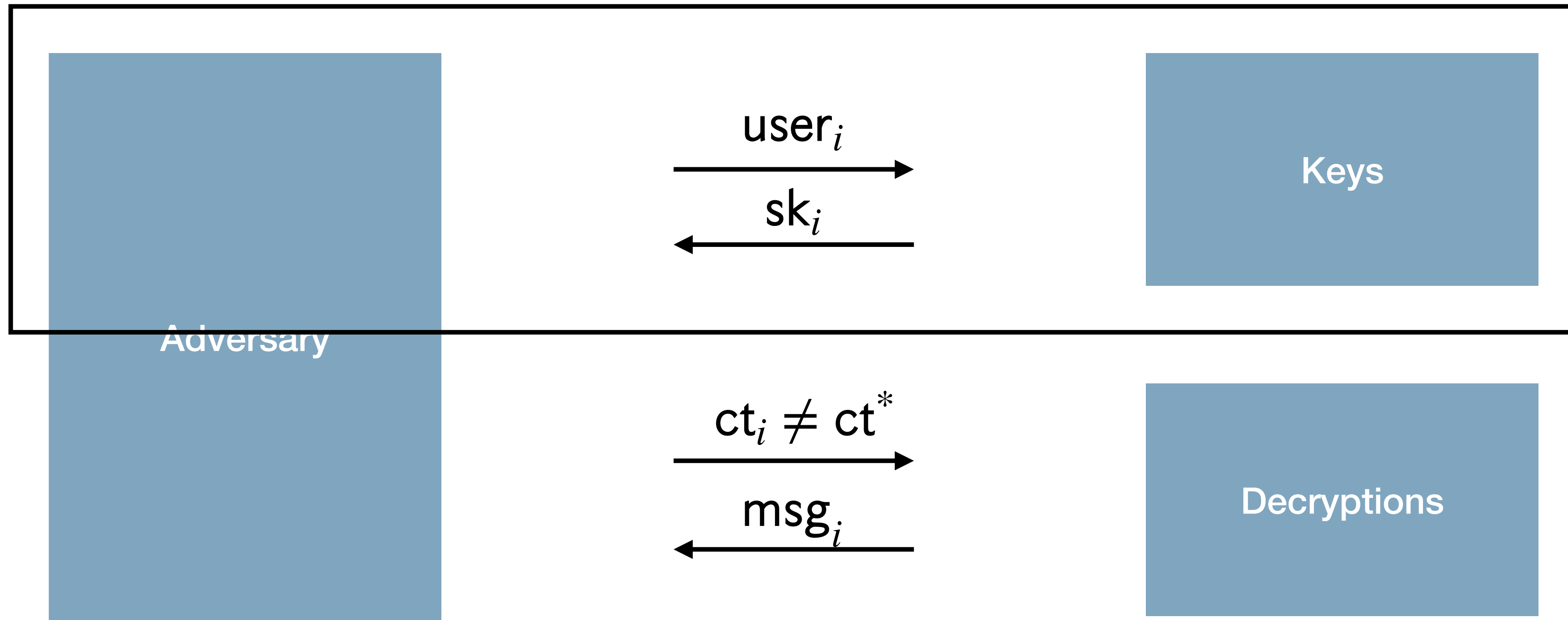
# Threshold Active Security

# Security Model

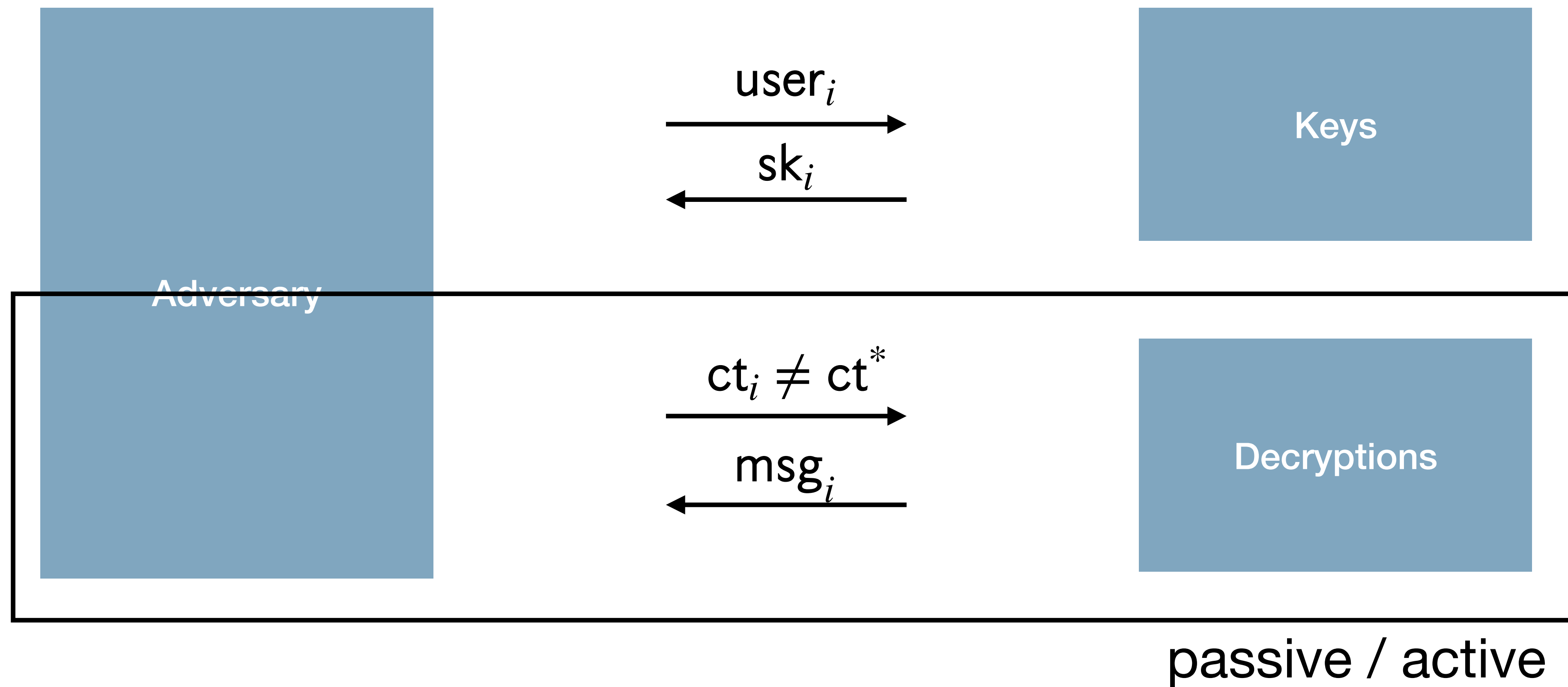


# Security Model

static / adaptive,  $\leq T$  users



# Security Model



# FO transform (non-threshold case)

$\text{msg} \xleftarrow{\$} \{0,1\}^{128}$   
 $\text{ct} = \text{Enc}(\text{msg}, \text{pk})$   
and  
 $\text{msg} = \text{Dec}(\text{ct}, \text{sk})$

# FO transform (non-threshold case)

$\text{msg} \xleftarrow{\$} \{0,1\}^{128}$   
 $\text{ct} = \text{Enc}(\text{msg}, \text{pk})$   
and  
 $\text{msg} = \text{Dec}(\text{ct}, \text{sk})$



*Enforce honest encryption*

- Generate encryption randomness via PRG
- Encrypt the PRG seed

# FO transform (non-threshold case)

$\text{msg} \xleftarrow{\$} \{0,1\}^{128}$   
 $\text{ct} = \text{Enc}(\text{msg}, \text{pk})$   
and  
 $\text{msg} = \text{Dec}(\text{ct}, \text{sk})$



*Enforce honest encryption*

- Generate encryption randomness via PRG
- Encrypt the PRG seed
- Recover the seed from Dec()
- Re-encrypt with the same seed and compare ciphertexts

# FO transform (non-threshold case)

$\text{msg} \xleftarrow{\$} \{0,1\}^{128}$   
 $\text{ct} = \text{Enc}(\text{msg}, \text{pk})$   
and  
 $\text{msg} = \text{Dec}(\text{ct}, \text{sk})$



$\text{msg} \xleftarrow{\$} \{0,1\}^{128}$   
 $\text{rand} = \text{PRG}(\text{msg})$   
 $\text{ct} = \text{Enc}(\text{msg}, \text{pk}; \text{rand})$   
and  
 $\text{msg} = \text{Dec}(\text{ct}, \text{sk})$   
 $\text{rand} = \text{PRG}(\text{msg})$   
 $\text{Enc}(\text{msg}, \text{pk}; \text{rand}) == \text{ct}?$

# Obtaining active security

MPC

FHE

ZK Proofs

Generic ?

# BCHK Transform

# BCHK Transform



# Identity-based Encryption

$(sk, pp) \leftarrow \text{ibe-KeyGen}()$   
 $ct = \text{ibe-Enc}(msg, pp, id)$

and

$sk_{id} = \text{ibe-Extract}(sk, id)$   
 $msg = \text{ibe-Dec}(ct, sk_{id})$

# BCHK Transform

$ct = \text{Enc}(msg, pk)$

and

$msg = \text{Dec}(ct, sk)$



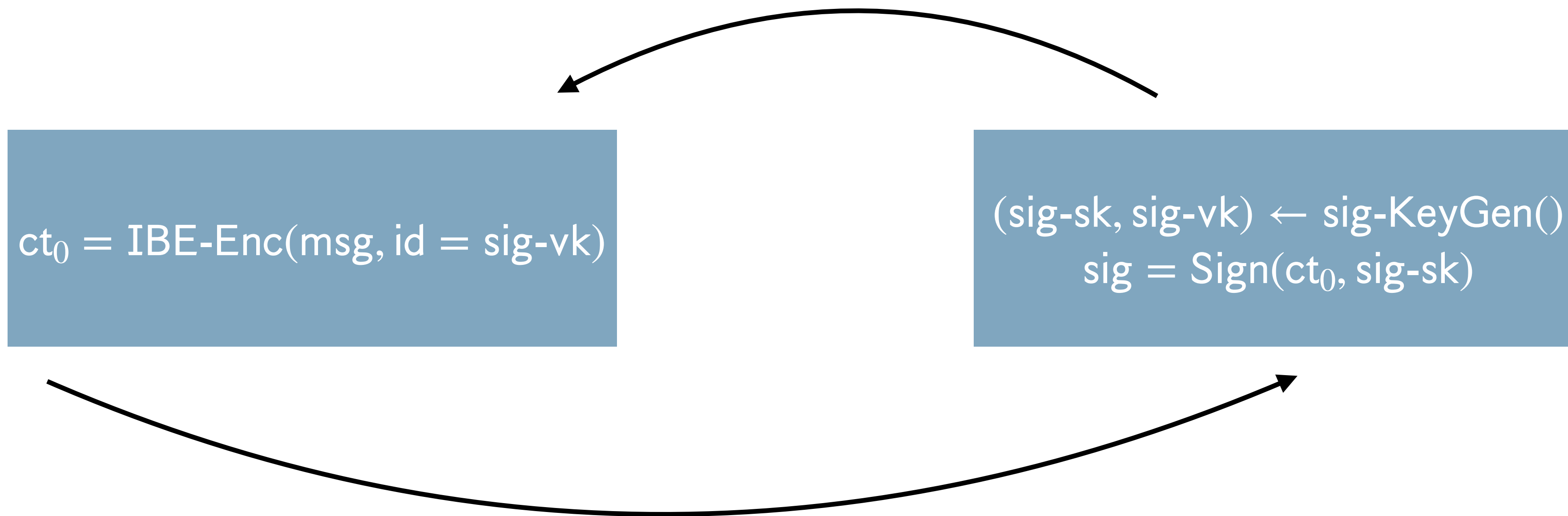
$(sig-sk, sig-vk) \leftarrow \text{sig-KeyGen}()$

$ct_0 = \text{ibe-Enc}(msg, id = sig-vk)$

$sig = \text{Sign}(ct_0, sig-sk)$

$ct = (ct_0, sig, sig-vk)$

# BCHK Transform



# BCHK Transform

$ct = \text{Enc}(msg, pk)$

and

$msg = \text{Dec}(ct, sk)$



$(sig-sk, sig-vk) \leftarrow sig\text{-KeyGen}()$

$ct_0 = \text{ibe-Enc}(msg, id = sig-vk)$

$sig = \text{Sign}(ct_0, sig-sk)$

$ct = (ct_0, sig, sig-vk)$

and

$\text{Verify}(ct_0, sig, sig-vk) == 1?$

$sk_{id} = \text{Extract}(sk, id = sig-vk)$

$msg = \text{Dec}(ct_0, sk_{id})$

# Threshold BCHK

$ct = \text{Enc}(msg, pk)$

and

$msg = \text{Dec}(ct, sk)$



$(sig-sk, sig-vk) \leftarrow sig\text{-KeyGen}()$

$ct_0 = \text{ibe-Enc}(msg, id = sig-vk)$

$sig = \text{Sign}(ct_0, sig-sk)$

$ct = (ct_0, sig, sig-vk)$

and

$\text{Verify}(ct_0, sig, sig-vk) == 1?$

$sk_{id} = \text{Extract}(sk, id = sig-vk)$

$msg = \text{Dec}(ct_0, sk_{id})$

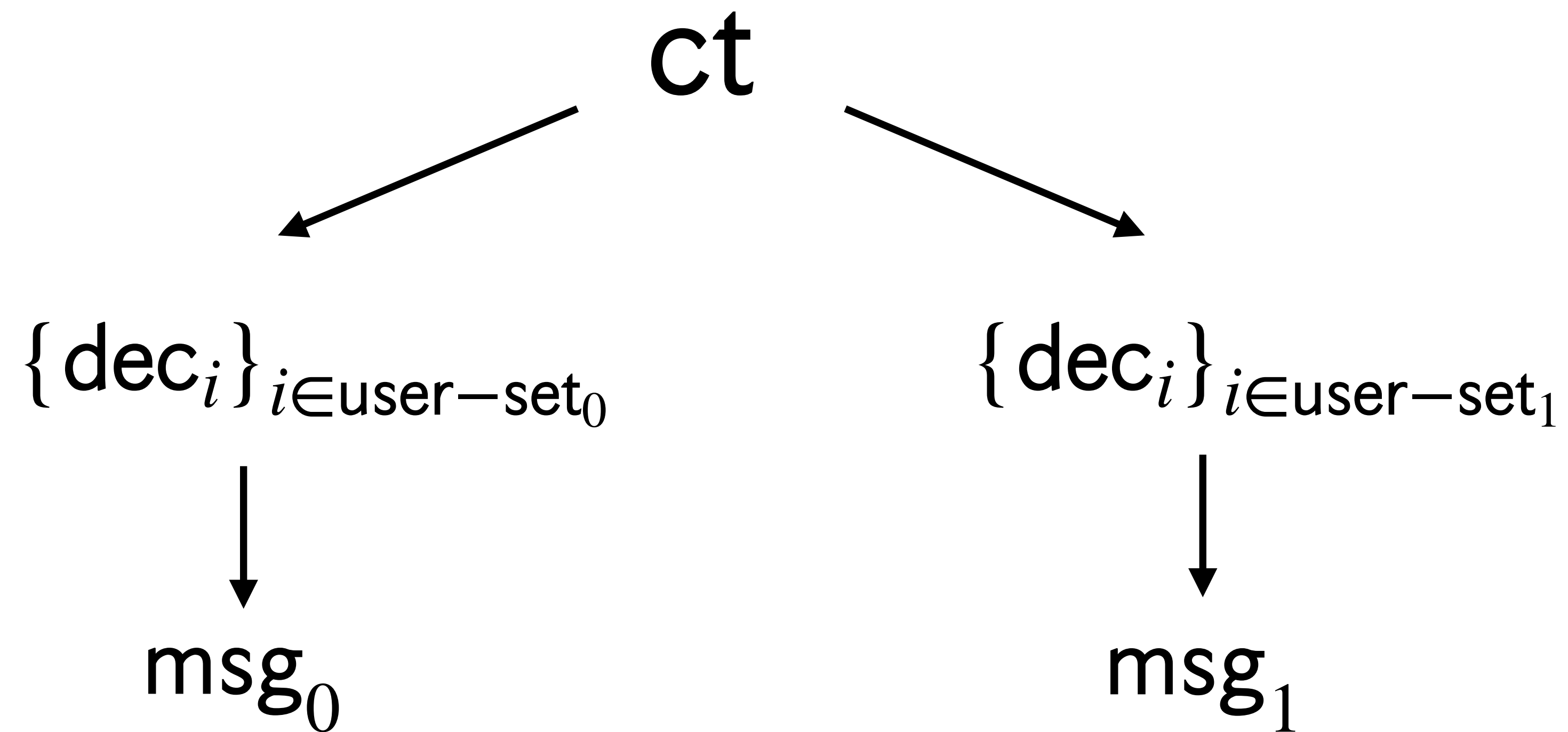
# Threshold BCHK



# BCHK<sub>+</sub> = BCHK + FO

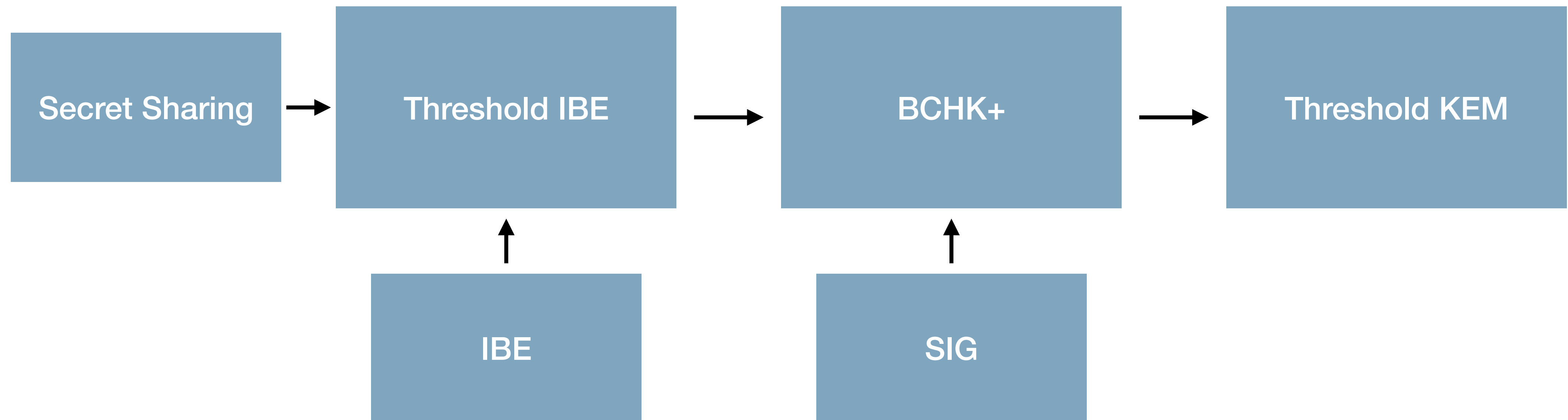


# Decryption Consistency



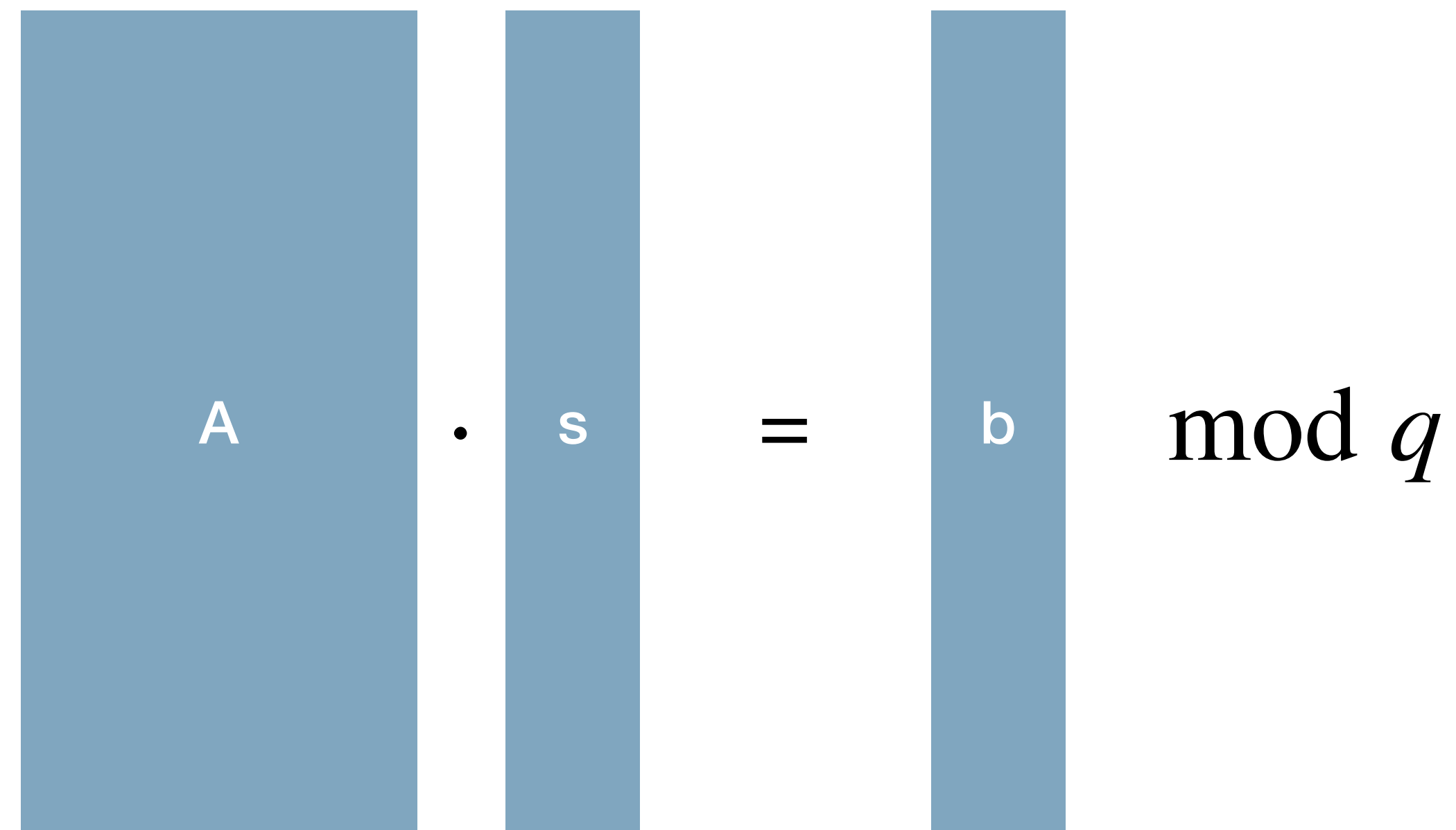
# Amber: The Lattice Construction

# The Blueprint



# Learning with Errors Problem

Easy problem: find unique  $\mathbf{s}$  such that

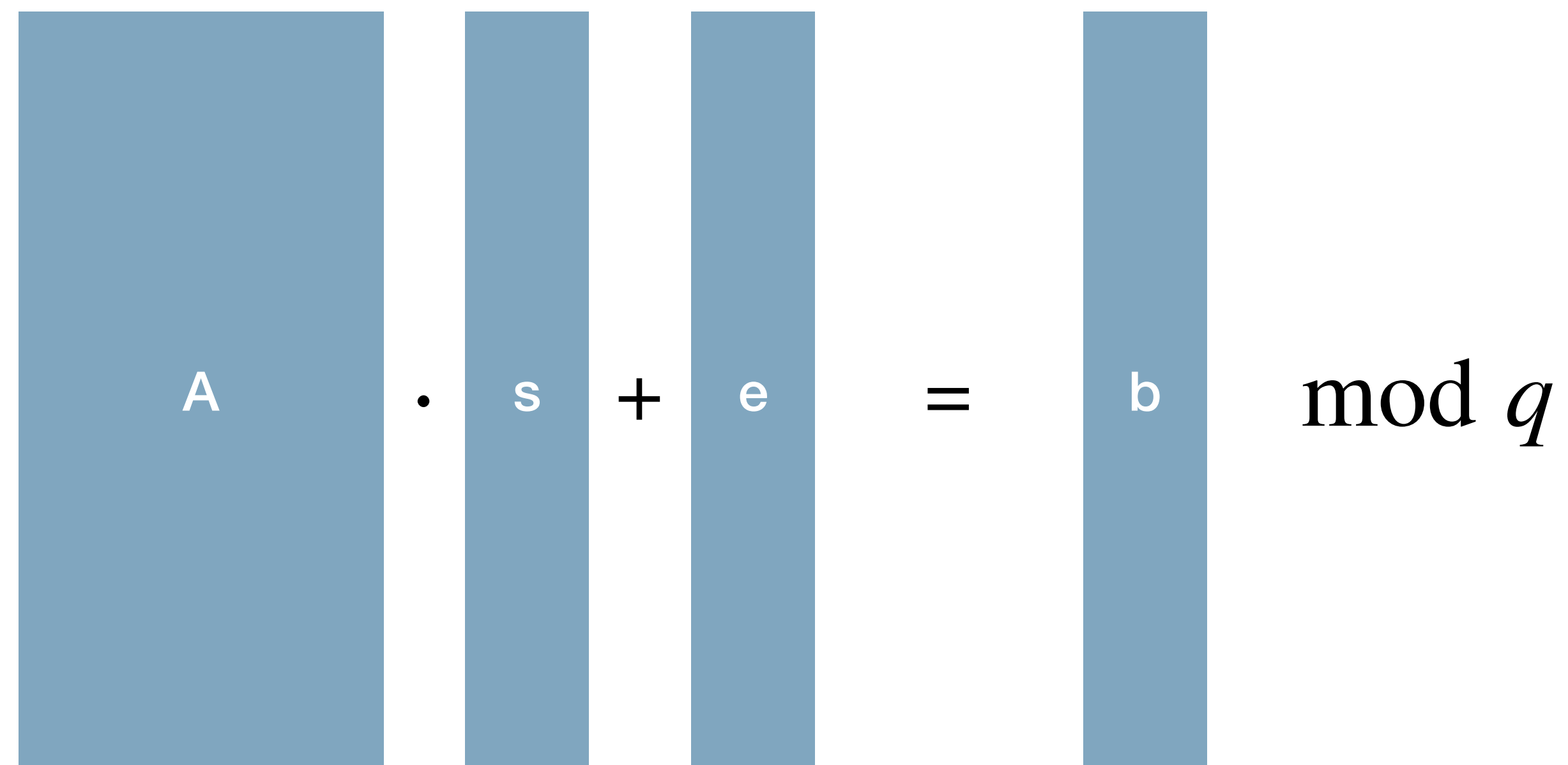


The diagram illustrates the equation  $A \cdot s = b \pmod q$ . It features three blue rectangular blocks: a tall block on the left labeled 'A', a shorter block in the middle labeled 's', and another tall block on the right labeled 'b'. These blocks are arranged horizontally, separated by a dot, an equals sign, and the text 'mod q'.

$$A \cdot s = b \pmod q$$

# Learning with Errors Problem

Hard problem: find  $\mathbf{s}$  and a *short noise*  $\mathbf{e}$  such that



The diagram illustrates the Learning with Errors (LWE) problem. It features four blue rectangular blocks of different widths. The first block is the widest and contains the letter 'A'. To its right is a small dot, followed by a narrower block containing the letter 's'. This is followed by a plus sign, then another narrow block containing the letter 'e'. To the right of this is an equals sign, followed by a narrow block containing the letter 'b'. Finally, to the right of the 'b' block is the text 'mod q'. The blocks for 's', 'e', and 'b' are all the same width, while the block for 'A' is significantly wider.

$$\mathbf{A} \cdot \mathbf{s} + \mathbf{e} = \mathbf{b} \pmod{q}$$

# Learning with Errors Problem

The set  $\{\mathbf{y} \mid \mathbf{y} = \mathbf{A} \cdot \mathbf{s} \bmod q\}$  is a lattice.

Finding  $\mathbf{e}$  from  $\mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$  is a *decoding problem*.

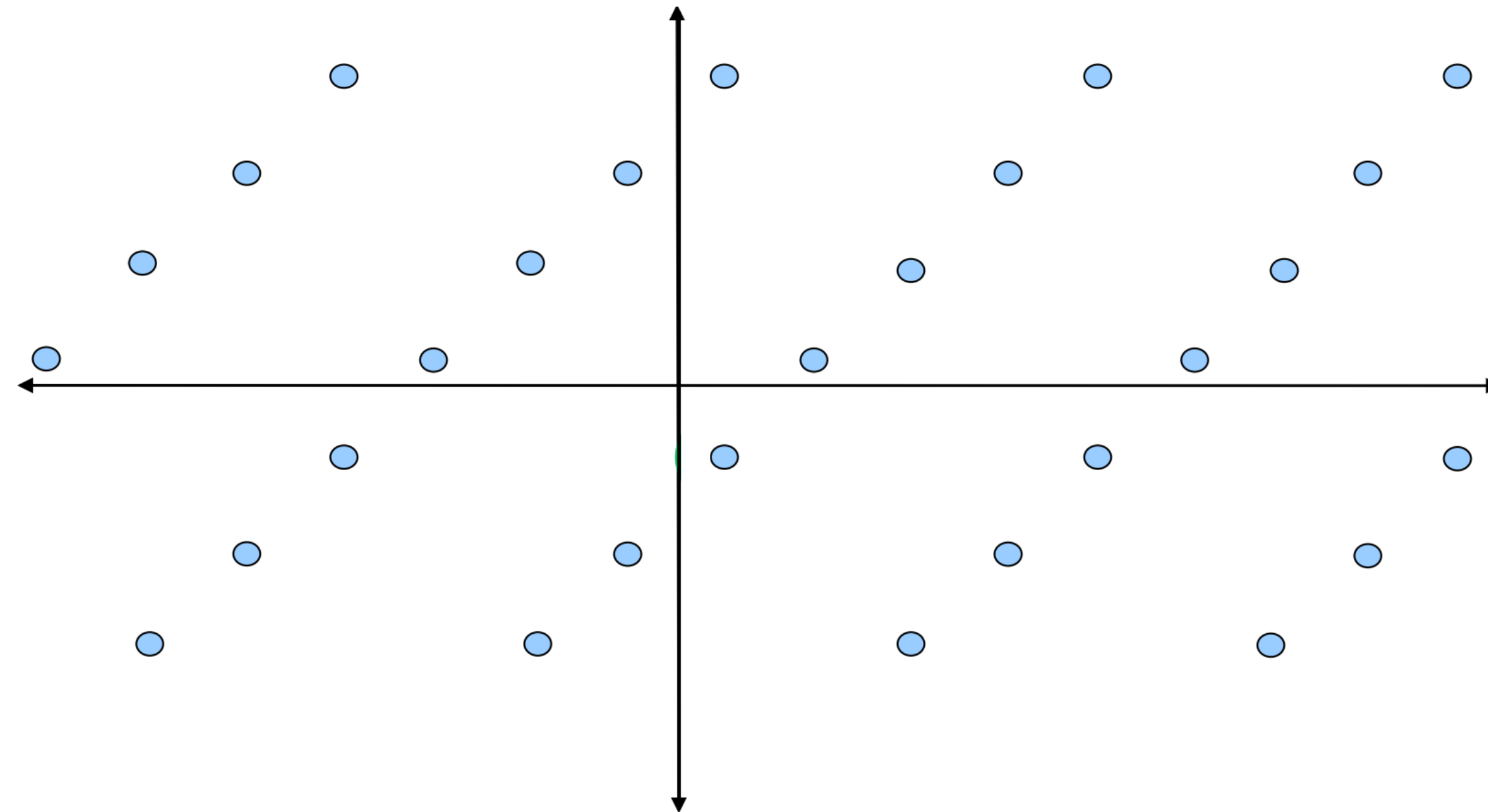


Image from V. Lyubashevsky, CRYSTALS-Dilithium Talk, 2018

# Learning with Errors Problem

The set  $\{\mathbf{y} \mid \mathbf{y} = \mathbf{A} \cdot \mathbf{s} \bmod q\}$  is a lattice.

Finding  $\mathbf{e}$  from  $\mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$  is a *decoding problem*.

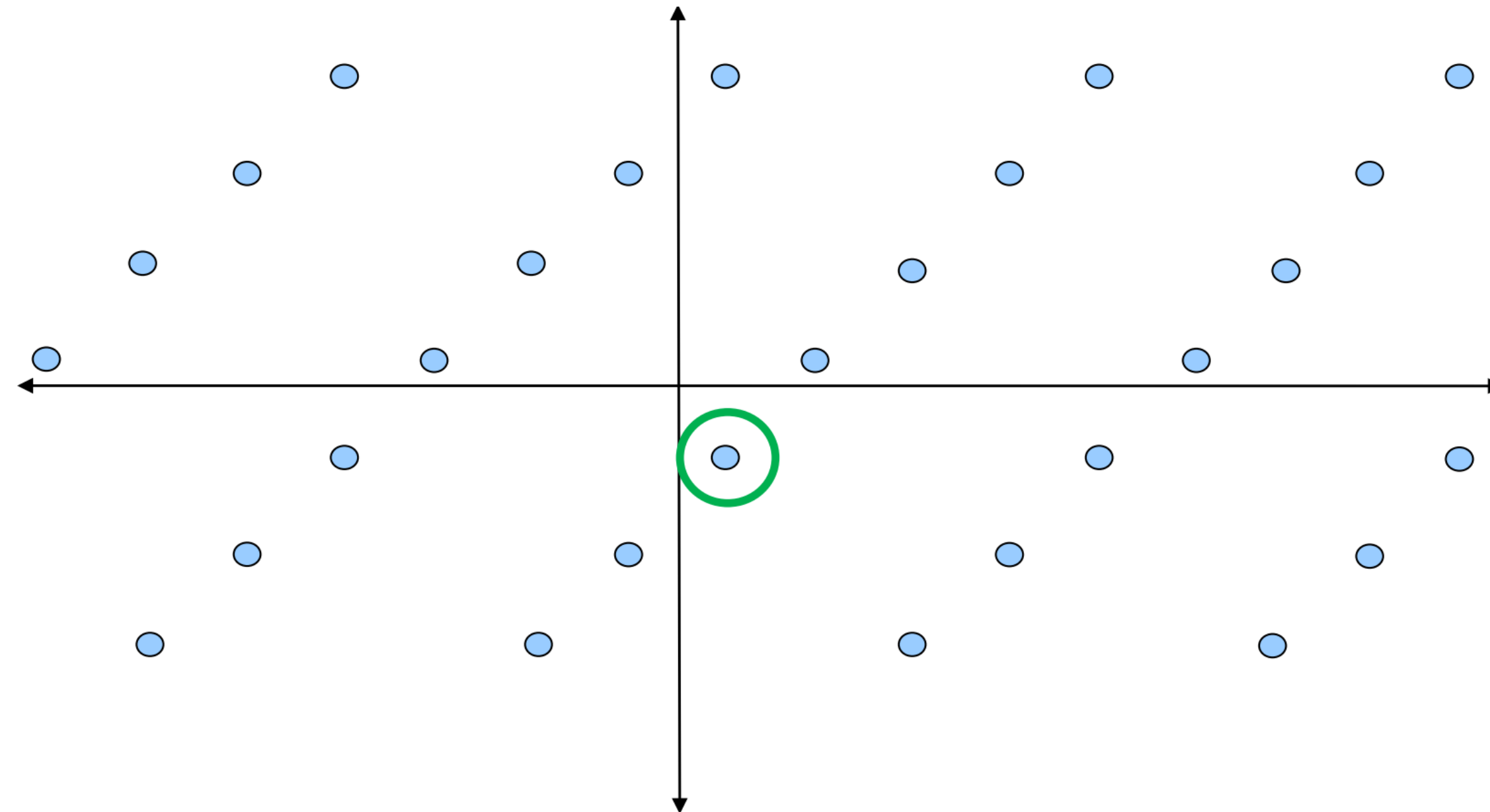


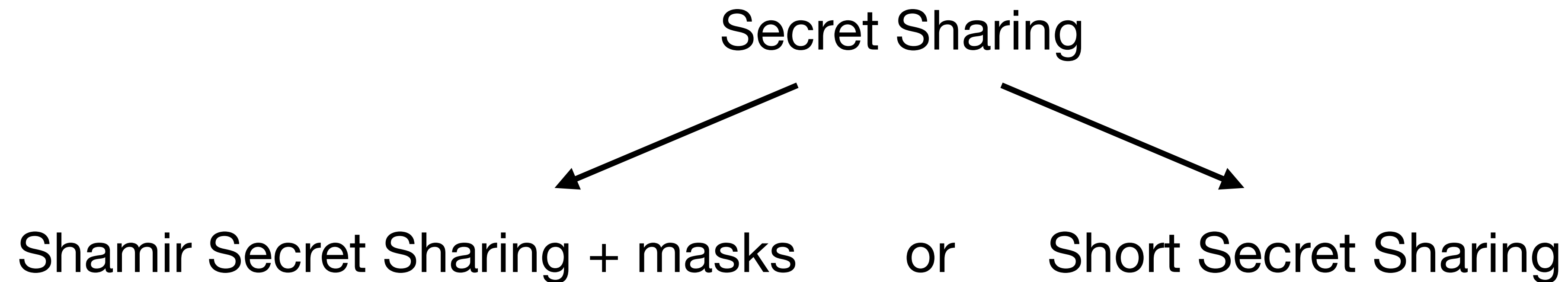
Image from V. Lyubashevsky, CRYSTALS-Dilithium Talk, 2018

# Secret Sharing

Secret keys: short vectors **s** and **e**

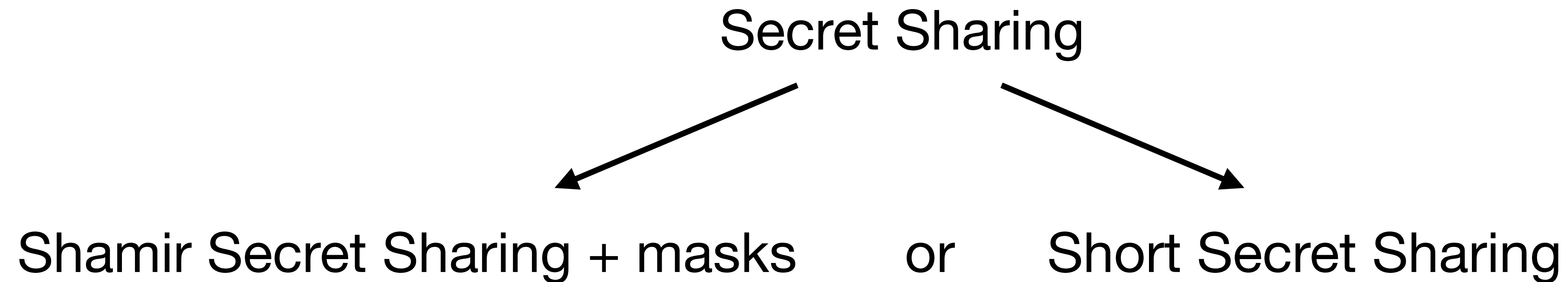
# Secret Sharing

Secret keys: short vectors **s** and **e**



# Secret Sharing

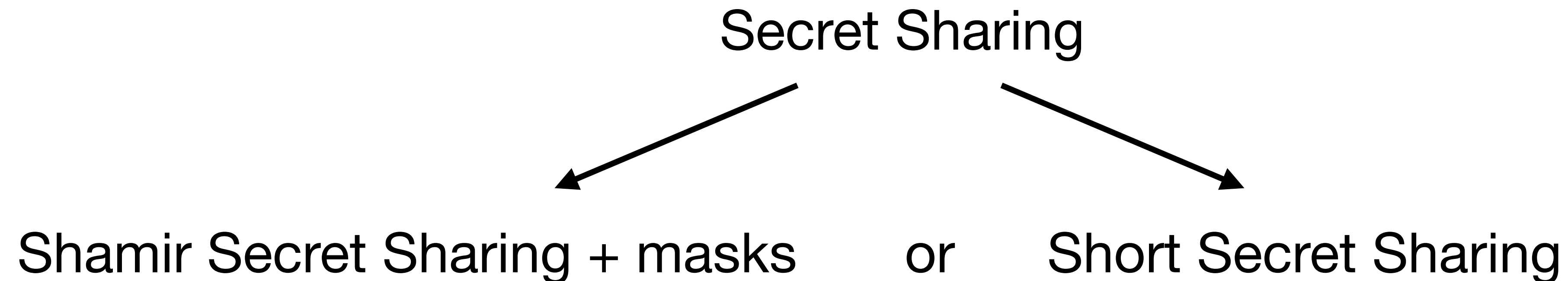
Secret keys: short vectors **s** and **e**



$$\mathbf{s} = \sum \lambda_i \mathbf{s}_i + \mathbf{m}_i$$
$$\sum \mathbf{m}_i = 0$$

# Secret Sharing

Secret keys: short vectors **s** and **e**



$$\mathbf{s} = \sum \lambda_i \mathbf{s}_i + \mathbf{m}_i$$
$$\sum \mathbf{m}_i = 0$$

$$\mathbf{s} = \sum \mathbf{s}_i$$

# Security Implications

| Variant   | Corruptions | Robustness | Group Size     |
|-----------|-------------|------------|----------------|
| Shamir SS | Adaptive    | No         | Large (< 1024) |
| Short SS  | Static      | Yes        | Medium (< 64)  |

# The parameters

| Variant   | Rounds | Decryption Queries | Pk size (bytes) | Ct size (bytes) |
|-----------|--------|--------------------|-----------------|-----------------|
| Shamir SS | 3      | $2^{46}$           | 6 688           | 28 544          |
| Short SS  | 3      | $2^{25}$           | 7 456           | 29 056          |

The number are stated for  $T = 32$  and security level of 128 bits.

# Future work

- ◆ NIST Threshold Call and Implementation
- ◆ Feedback on the security features
- ◆ Further reducing the sizes
- ◆ Kyber compatibility

# Summary

**“Amber: Lattice-Based Threshold KEM with Active Security” | [ia.cr/2025/1958](https://ia.cr/2025/1958), [ia.cr/2026/021](https://ia.cr/2026/021)**

- ◆ Active security definitions
- ◆ BCHK and FO transforms
- ◆ Amber blueprints
- ◆ Lattice-based secret sharing

# Summary

**“Amber: Lattice-Based Threshold KEM with Active Security” | [ia.cr/2025/1958](https://ia.cr/2025/1958), [ia.cr/2026/021](https://ia.cr/2026/021)**

- ◆ Active security definitions
- ◆ BCHK and FO transforms
- ◆ Amber blueprints
- ◆ Lattice-based secret sharing

# Thank you!

# BCHK+ = BCHK + FO

TKEM.Keygen( $1^\kappa$ )

$(\{dk_i\}_{i \in [N]}, ek) \leftarrow \text{TIBE.Setup}(1^\kappa)$   
**return**  $(\{dk_i\}_{i \in [N]}, ek)$

TKEM.Encaps(ek)

$msg \leftarrow \{0, 1\}^d$   
 $rand = G_{fo}(msg)$   
 $(sk, vk) \leftarrow \text{SIG.Keygen}(1^\kappa)$   
 $ct = \text{TIBE.Encrypt}(ek, id = vk, msg; rand)$   
 $sig \leftarrow \text{SIG.Sign}(sk, ct)$   
 $K = H_{fo}(msg || ct)$   
**return**  $(ct, vk, sig), K$

TKEM.Decaps( $dk_i, (ct, vk, sig)$ )

**assert**  $\text{SIG.Verify}(vk, ct, sig) = 1$   
 $contrib_i = \text{TIBE.Extract}(dk_i, id = vk)$   
**return**  $contrib_i$

TKEM.Combine( $\{contrib_i\}_{i \in \text{act}}, (ct, vk, sig)$ )

**assert**  $\text{SIG.Verify}(vk, ct, sig) = 1$   
 $msg = \text{TIBE.Combine}(ct, vk, \text{act}, \{contrib_i\}_i)$   
 $rand = G_{fo}(msg)$   
**assert**  $ct = \text{TIBE.Encrypt}(ek, vk, msg; rand)$   
 $K = H_{fo}(msg || ct)$   
**return**  $K$

# Dual-Regev Encryption Scheme

- Public key  $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{r} \in \mathbb{Z}_q^n)$
- Secret key: short  $\mathbf{T}_A \in \mathbb{Z}^{m \times m}$  s.t.  $\mathbf{T}_A \cdot \mathbf{A} = \mathbf{0} \bmod q$
- Ciphertext:  $\mathbf{u} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ ,  $v = \mathbf{r}^T \mathbf{s} + \mathbf{e}' + \text{msg}$
- Decryption key: short  $\mathbf{x}$  s.t.  $\mathbf{x}^T \cdot \mathbf{A} = \mathbf{r} \bmod q$

$$\begin{aligned} v - \mathbf{x}^T \cdot \mathbf{u} &= \mathbf{r}^T \mathbf{s} + \mathbf{e}' + \text{msg} - \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{s} - \mathbf{x}^T \cdot \mathbf{e} \\ &= \text{msg} + \mathbf{e}' - \mathbf{x}^T \cdot \mathbf{e} = \text{msg} + \text{noise} \end{aligned}$$

# Lattice-Based IBE

- Public key  $(\mathbf{A} \in \mathbb{Z}_q^{m \times n}, \mathbf{r} \in \mathbb{Z}_q^n)$
- Master Secret key: short  $\mathbf{T}_A \in \mathbb{Z}^{m \times m}$  s.t.  $\mathbf{T}_A \cdot \mathbf{A} = \mathbf{0} \bmod q$
- Ciphertext:  $\mathbf{u} = \begin{bmatrix} \mathbf{A} \\ \mathbf{H}(\text{id}) \end{bmatrix} \cdot \mathbf{s} + \mathbf{e}, v = \mathbf{r}^T \mathbf{s} + \mathbf{e}' + \text{msg}$
- Identity key: short  $\mathbf{x}$  s.t.  $\mathbf{x}^T \cdot \begin{bmatrix} \mathbf{A} \\ \mathbf{H}(\text{id}) \end{bmatrix} = \mathbf{r}^T \bmod q$