Resolving the Efficiency-Utility Dilemma of Threshold Linearly Homomorphic Encryption via Message-Space Adapter

Yijia Chang

Rongmao Chen*

Chao Lin

Songze Li

Xinyi Huang*

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

国防科技大学 NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY

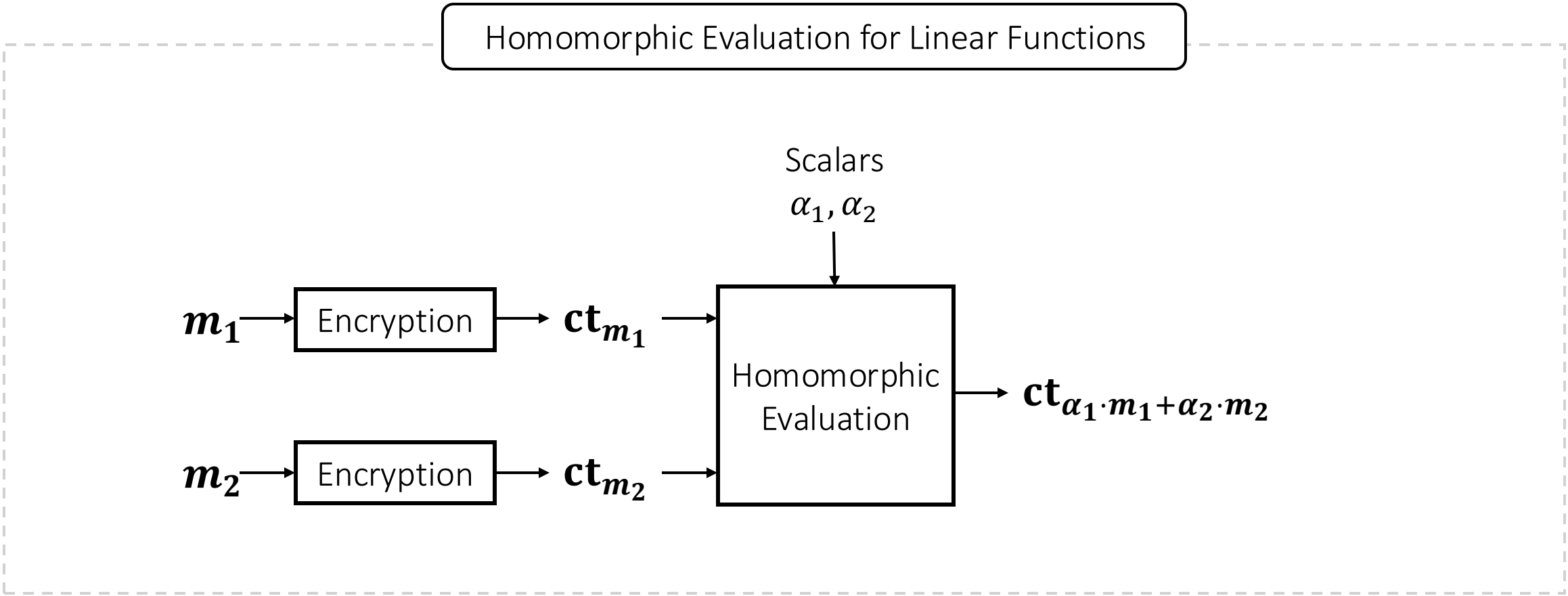
福建師範大學 FUJIAN NORMAL UNIVERSITY

東南大學 SOUTHEAST UNIVERSITY

暨南大學 JINAN UNIVERSITY

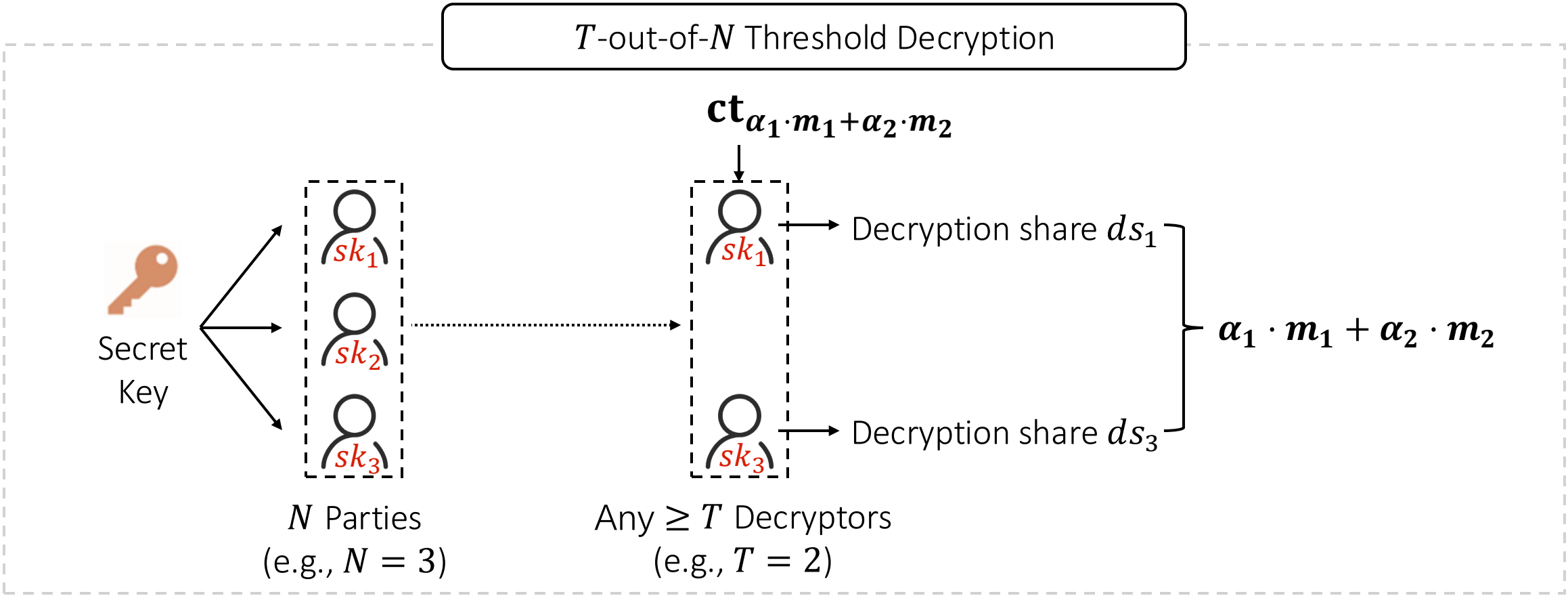
Threshold Linearly Homomorphic Encryption (ThLHE)

ThLHE supports homomorphic evaluation and threshold decryption



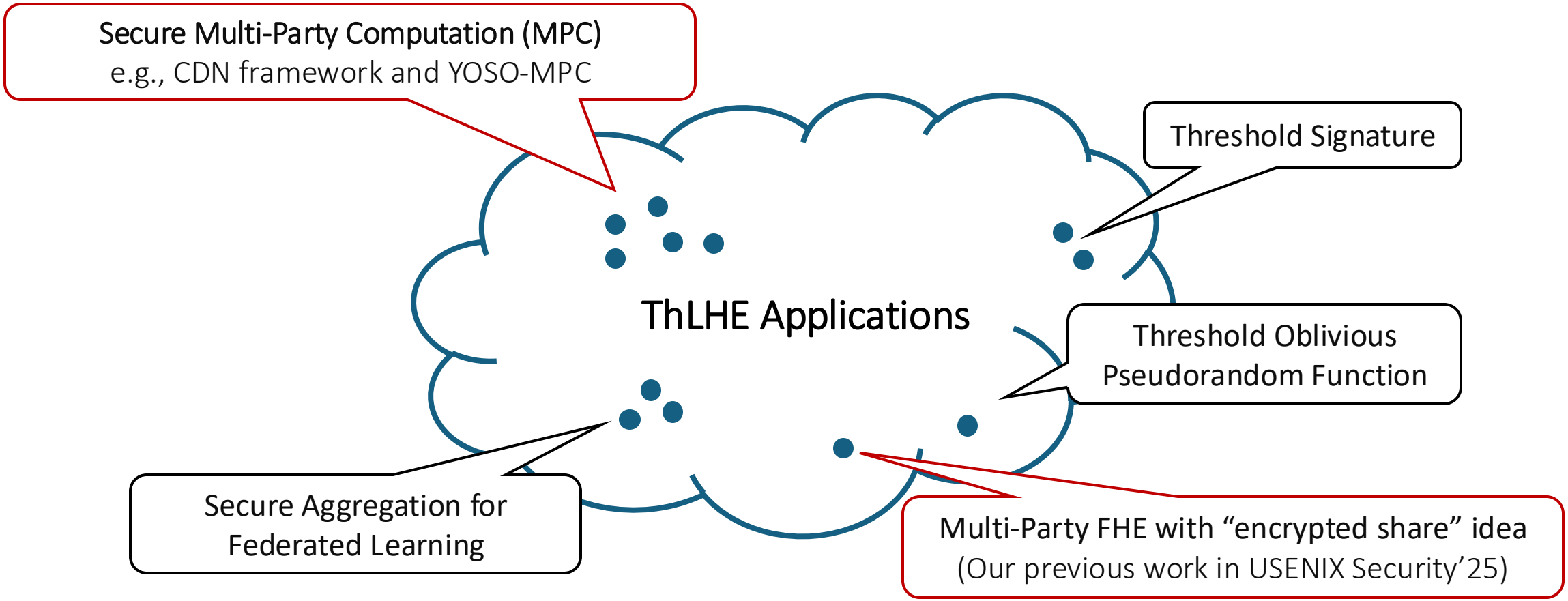
Threshold Linearly Homomorphic Encryption (ThLHE)

ThLHE supports homomorphic evaluation and threshold decryption



Why We Study ThLHE: Reason 1

ThLHE has many applications in multi-party setting



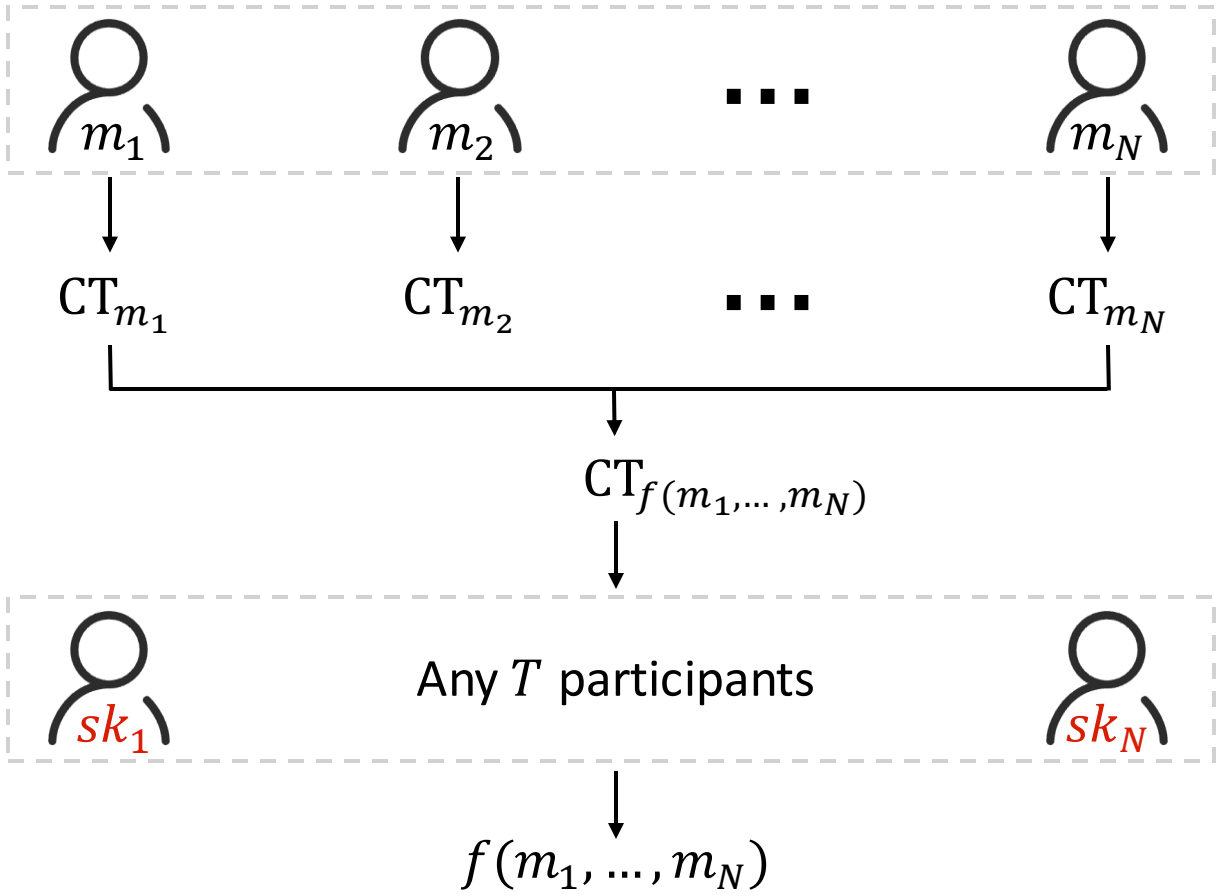
Why We Study ThLHE: Reason 2

ThLHE has low communication cost with $O(N)$ complexity due to less interactions

1. Encryption

2. Evaluation

3. Threshold Decryption



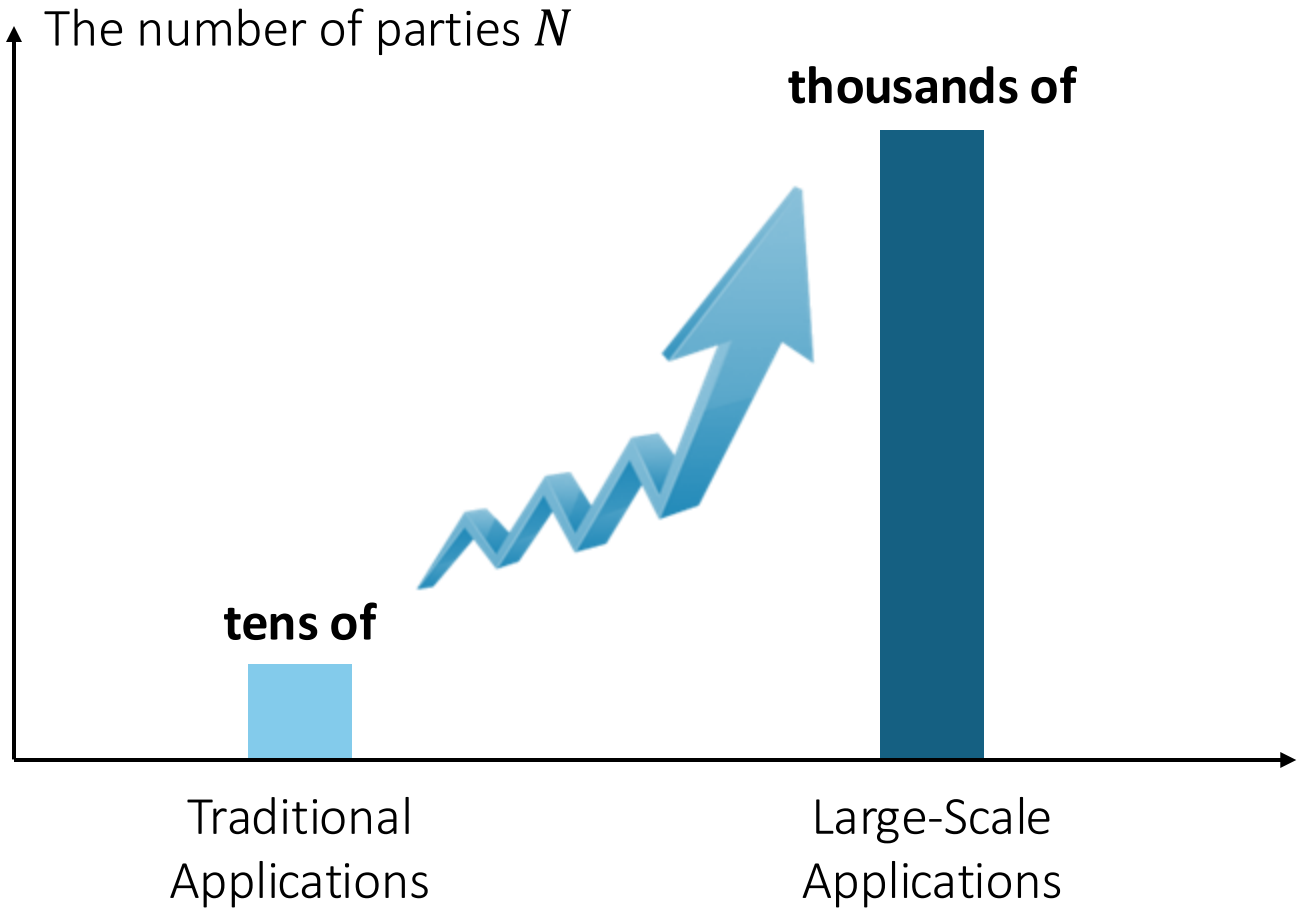
➡ no interactions

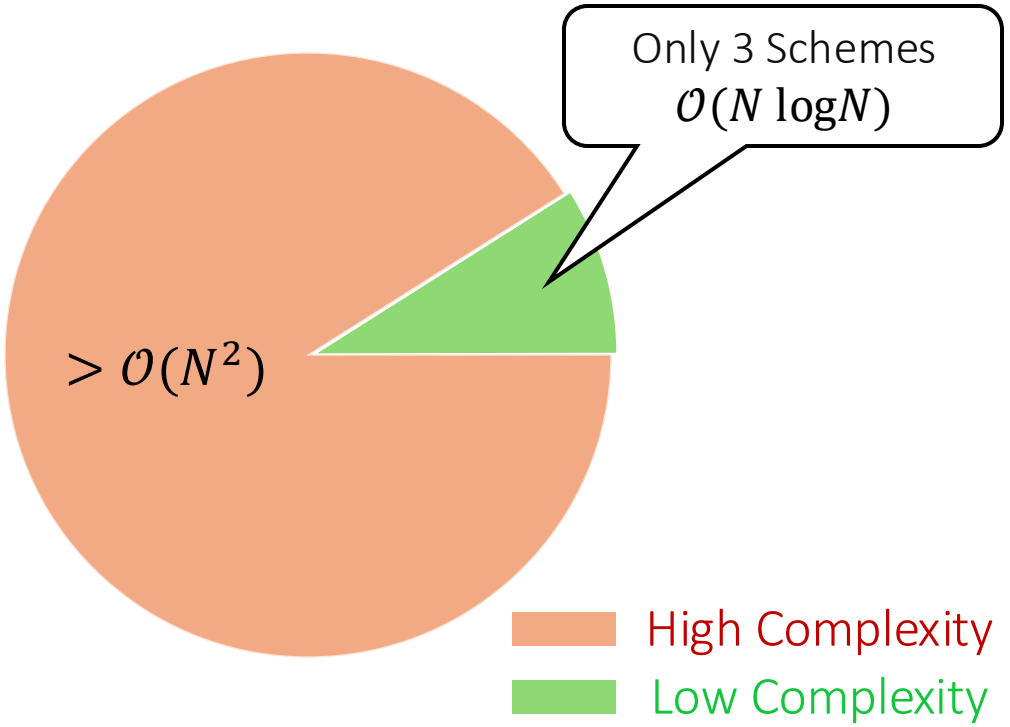
➡ broadcast only one ciphertext

➡ one-shot P2P interaction

ThLHE Is Currently Less Efficient Than Expected

Most of schemes are inefficient due to high computation complexity of threshold decryption





Efficient ThLHE Schemes Suffer From Utility Restrictions

Existing schemes with low complexity has either small message-space or static decryptor-set

DDH-Based Exponential ElGamal

Messages must be from a small, known set ⟶ ✖ ⟶ Applications usually have large message set

Encryption Decryption

$$m \longrightarrow g^m \longrightarrow \mathrm{CT}_m \longrightarrow g^m \longrightarrow m$$

This step is practical only if m is small

$m \in \mathbb{Z}_p$
p is large (e.g.,124-bit)

CDN Framework for Secure MPC

Integer Message
e.g., $m = 1010231$
↑
Floating Parameter
e.g., 1.010231

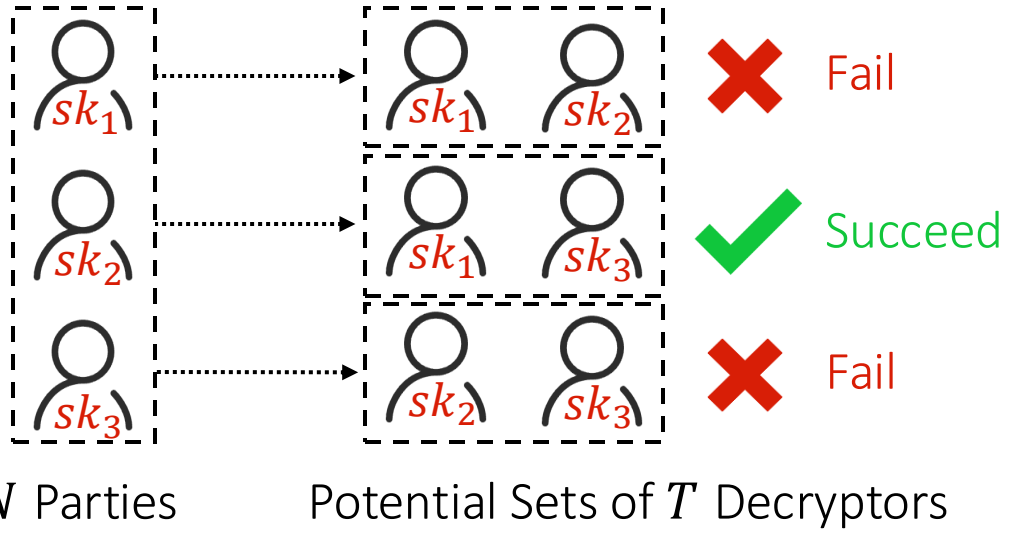
AI Model Aggregation

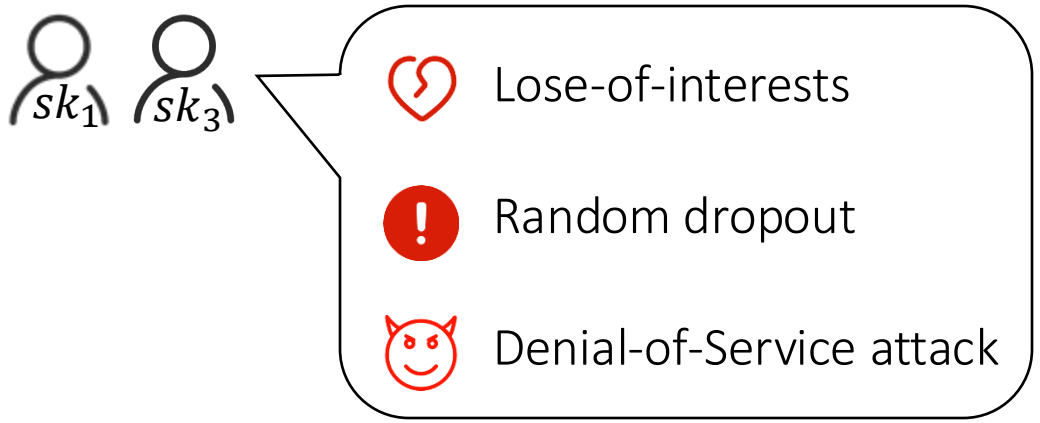
Efficient ThLHE Schemes Suffer From Utility Restrictions

Existing schemes with low complexity has either small message-space or static decryptor-set

Lattice-Based Threshold Fully HE

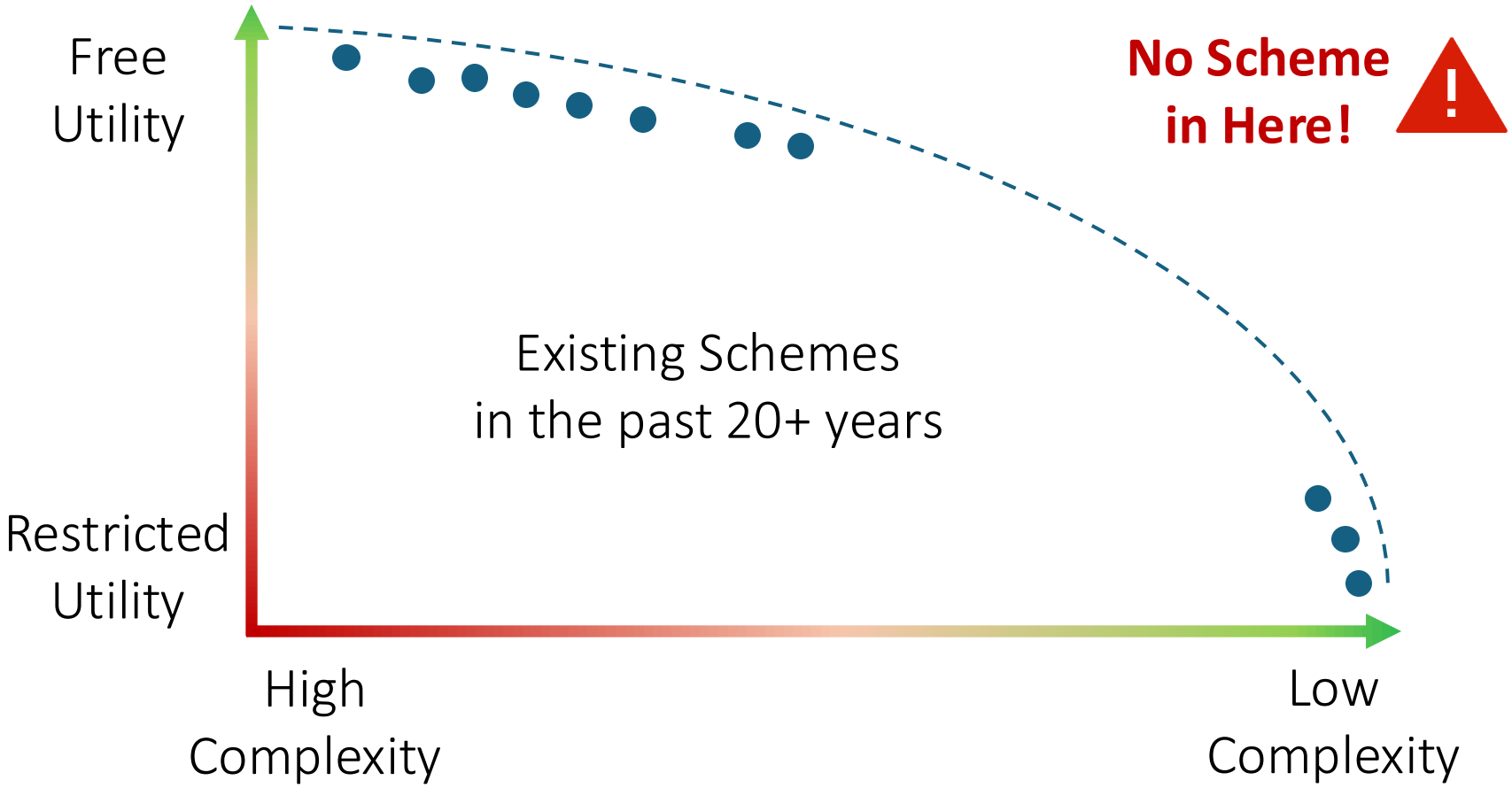
Successful decryption assumes a pre-determined set of decryptors will participate, which may not hold

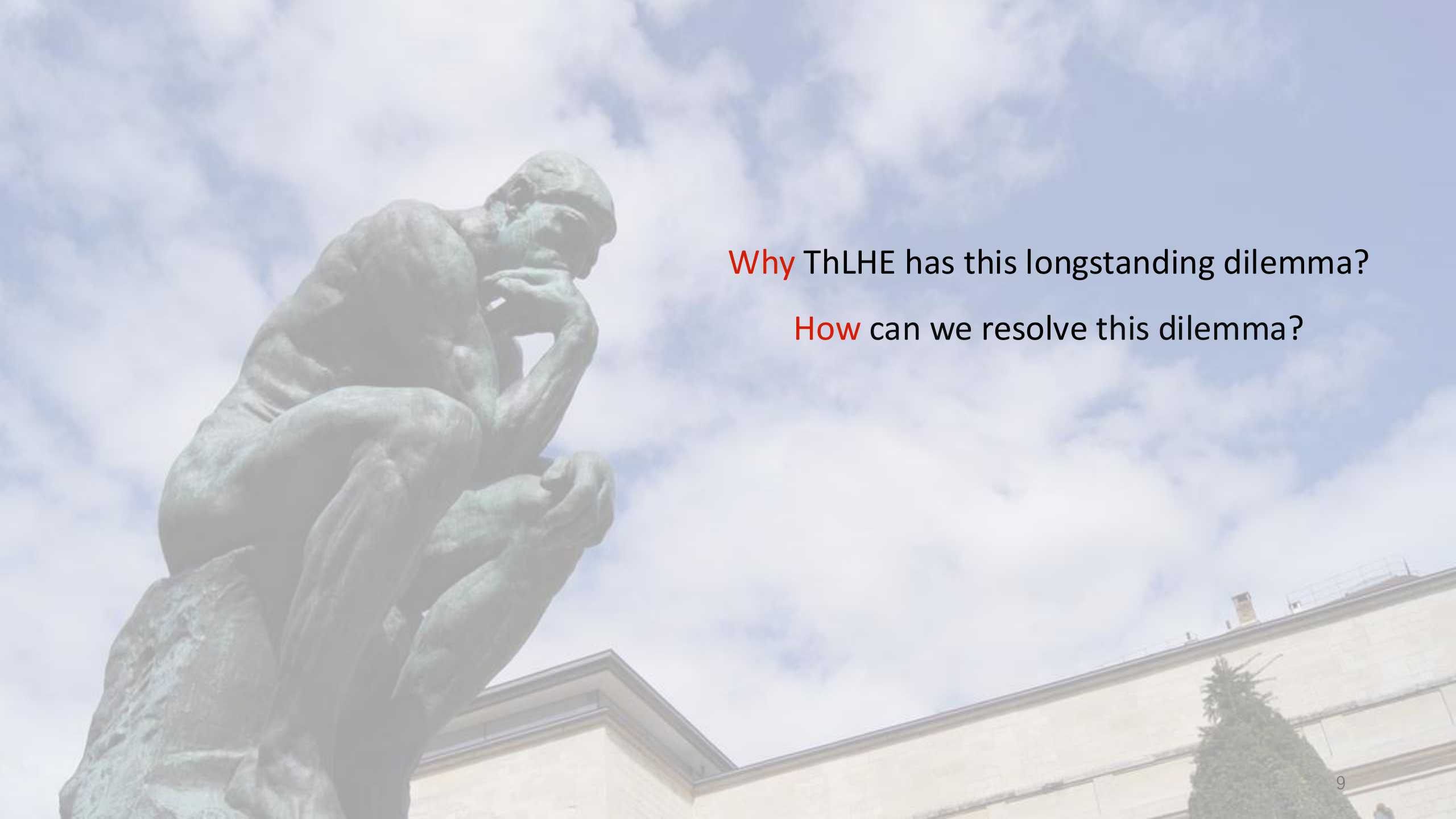




Efficiency-Utility Dilemma of ThLHE

In the past 20+ years, existing ThLHE schemes have either high complexity or utility restrictions



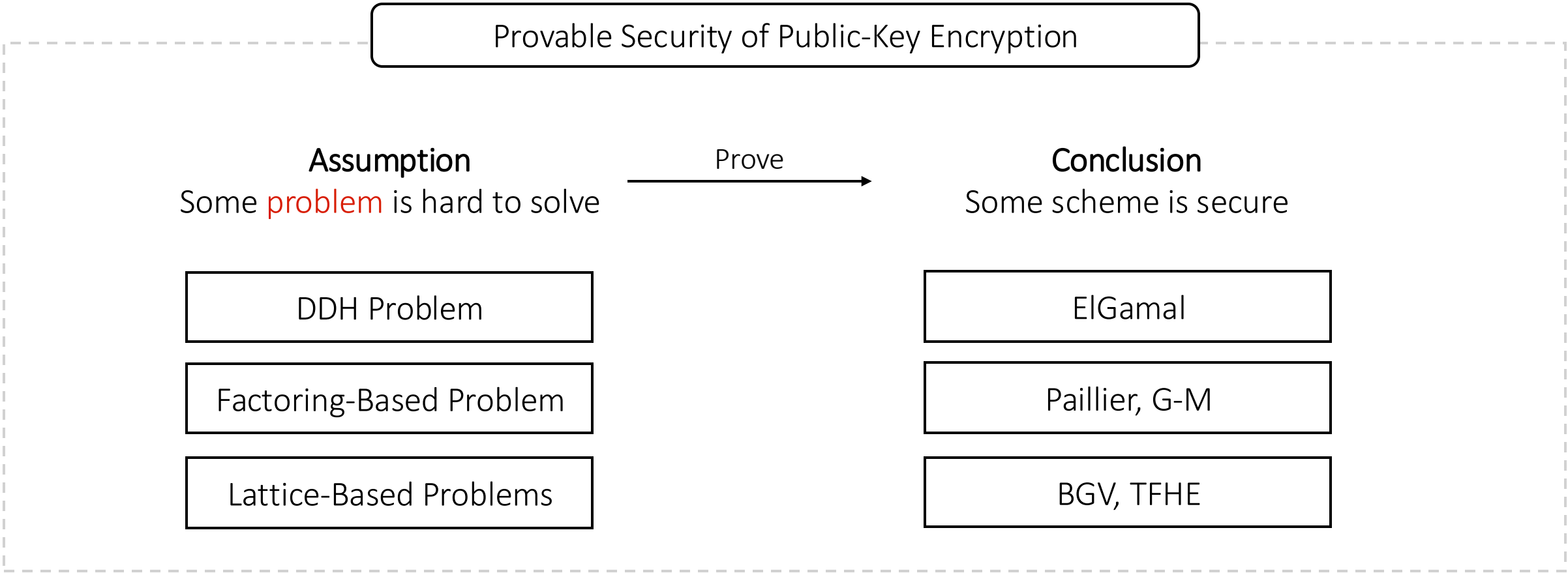


Why ThLHE has this longstanding dilemma?

How can we resolve this dilemma?

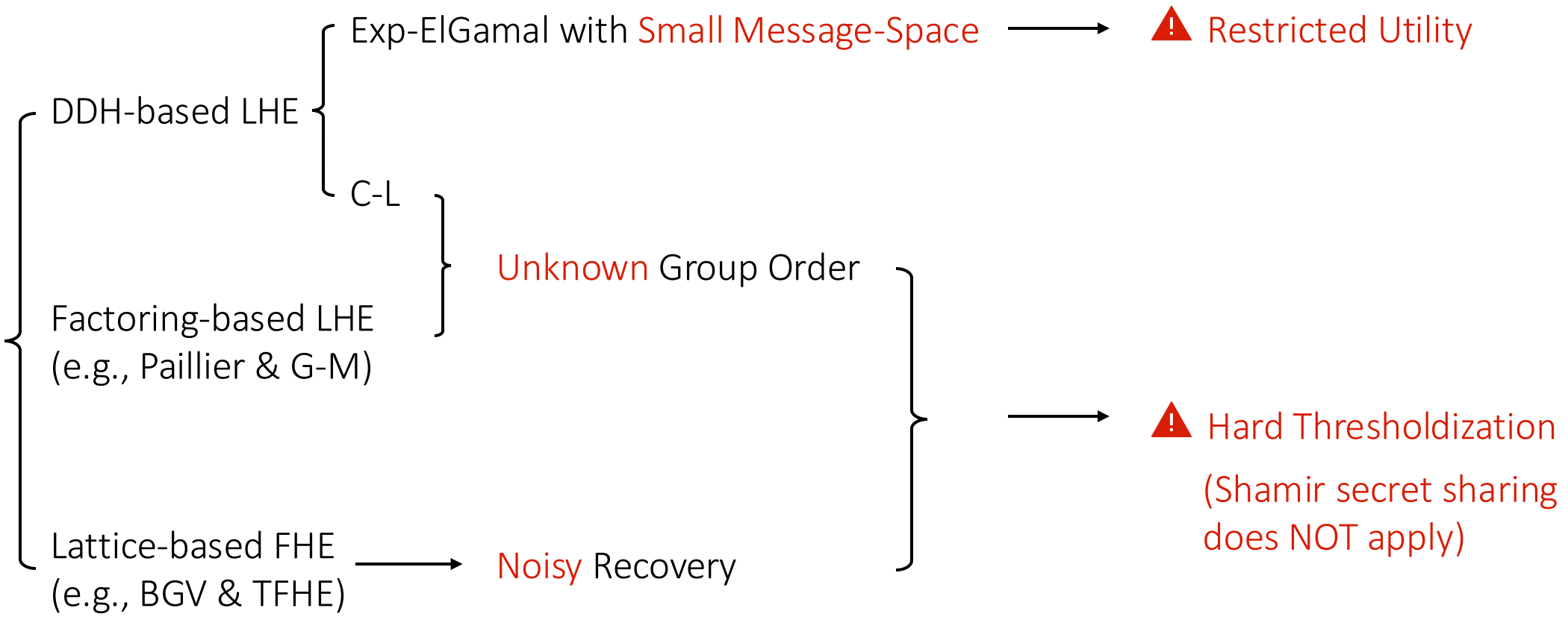
Quick Answer: Why This Dilemma Exists

The security assumptions make HE either restricted or very challenging to be thresholdized



Quick Answer: Why This Dilemma Exists

The security assumptions make HE either restricted or very challenging to be thresholdized



Quick Answer: Resolving This Dilemma for Threshold LHE

Instead of improving SS, we propose the first LHE with easy thresholdization and flexible utility

Quick Answer: Resolving This Dilemma for Threshold LHE

Instead of improving SS, we propose the first LHE with easy thresholdization and flexible utility

Traditional Perspective

Let's devise a stronger hammer

Quick Answer: Resolving This Dilemma for Threshold LHE

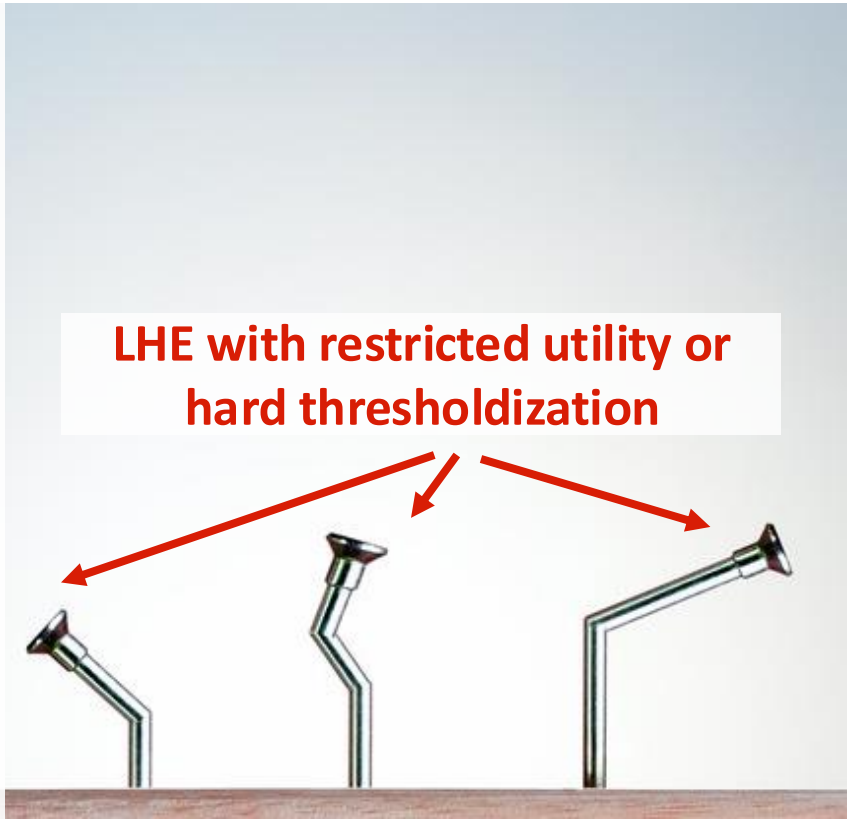
Instead of improving SS, we propose the first LHE with easy thresholdization and flexible utility

Traditional Perspective

Let's devise a stronger hammer

Secret Sharing (SS)

LHE with restricted utility or hard thresholdization

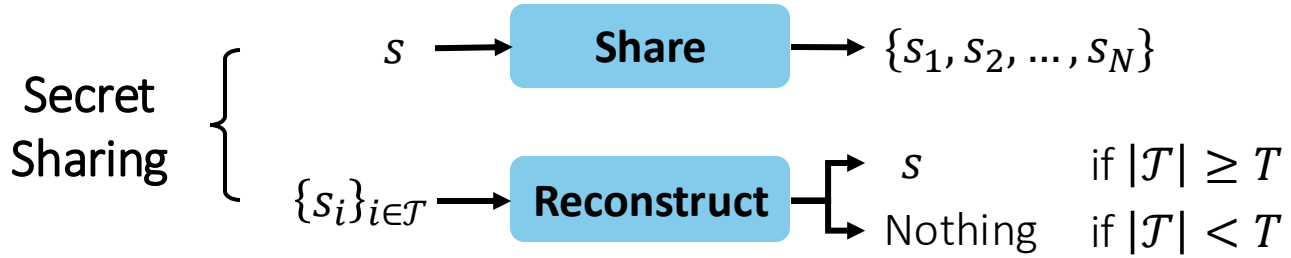


Our New Perspective

Let's try easy nails

Potential Tool for ThLHE Construction

We can employ computationally-efficient secret sharing (SS) to thresholdize LHE



Shamir SS with $1 \leq T \leq N$

$$s = \sum_{i\in\mathcal{T}} L_{\mathcal{T},i} \cdot s_i \pmod{P} \qquad L_{\mathcal{T},i} = \prod_{j\in\mathcal{T}, j\neq i} \frac{x_j}{x_j - x_i} \pmod{P}$$

for any set $\mathcal{T}$ of $\geq T$ parties

✓ Size of s = Size of s_i ✓ $O(N \log N)$ Computations

Group order P must be public information

Problems for Thresholdizing Factoring-Based ThLHE

Due to the unknown group order, Shamir secret sharing (SS) cannot be used

Must keep secret to avoid the leakage of p and q

Large Primes $\boldsymbol{p, q}$ ⇄ (Easy / Hard) RSA Integer $\boldsymbol{N = p \cdot q}$

Group Order $\boldsymbol{P = \phi(N) = (p-1)\cdot(q-1)}$

Typical Solution: "clearing-out-denominator" technique

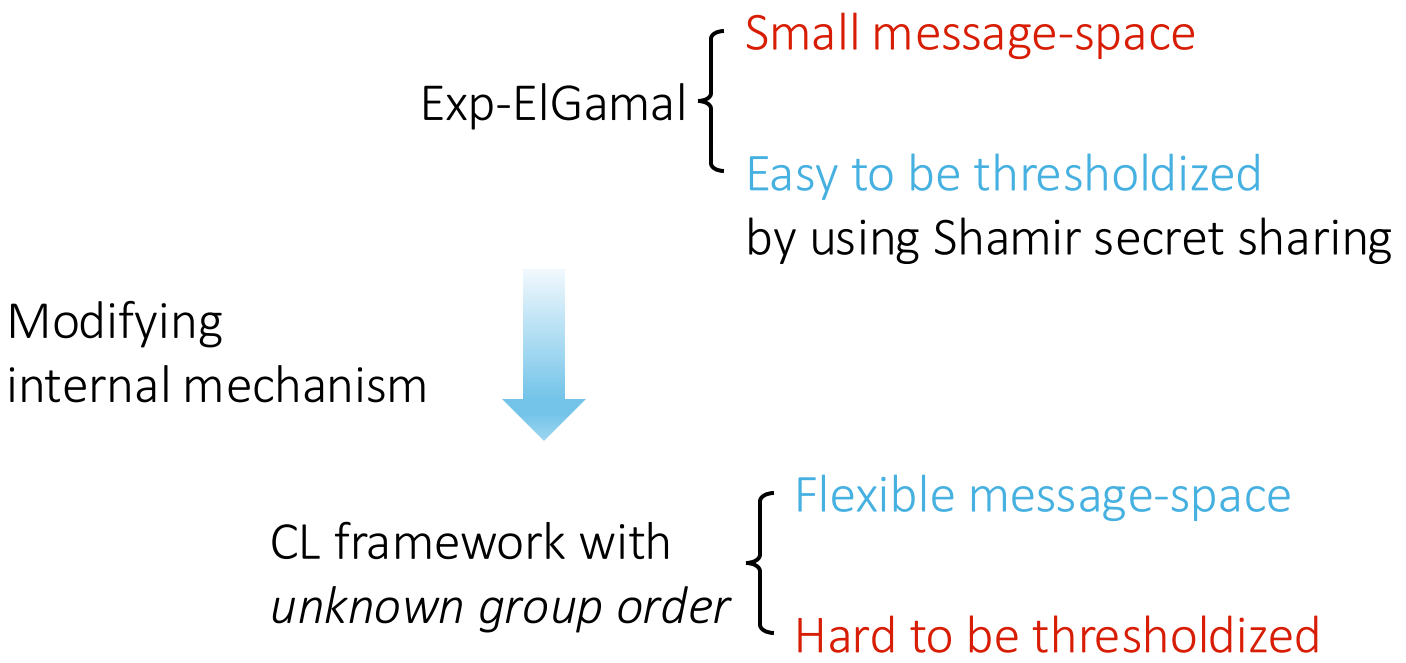
An integer (without mod P) but super-large

$$L_{\mathcal{T},i} = \prod_{j\in\mathcal{T}, j\neq i} \frac{x_j}{x_j - x_i} \pmod P \longrightarrow L'_{\mathcal{T},i} = (N!)\cdot \prod_{j\in\mathcal{T}, j\neq i} \frac{x_j}{x_j - x_i}$$

The aggregation of decryption shares $\{ds_i\}_{i\in\mathcal{T}}$: $\prod_{i\in\mathcal{T}} ds_i^{L'_{\mathcal{T},i}}$ ✓ $O(N^2 \log N)$ Computations

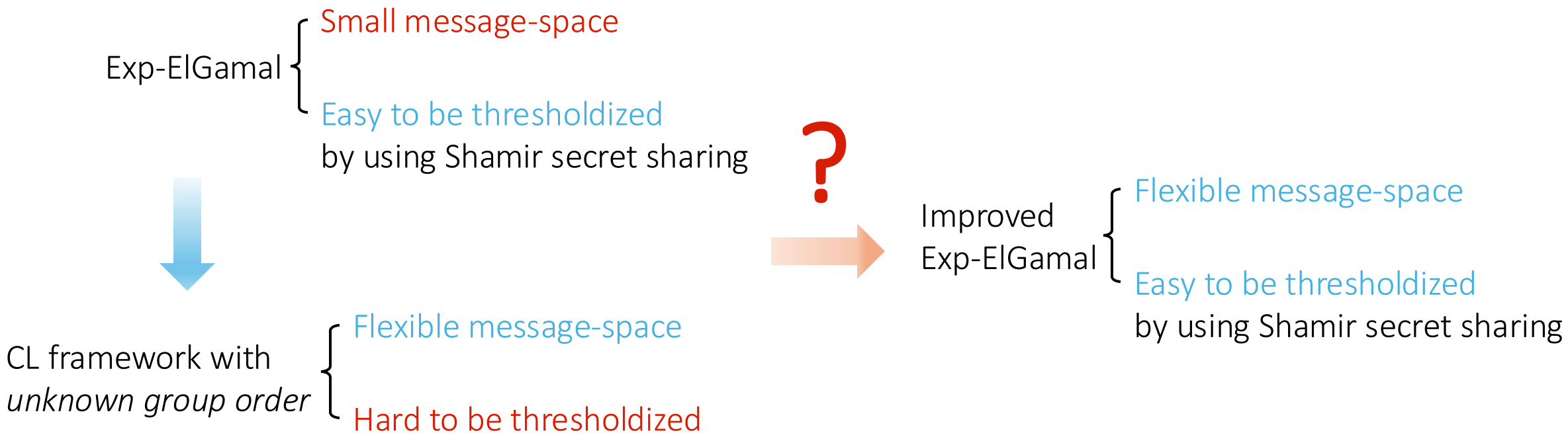
Problems for Thresholdizing DDH-Based ThLHE

- Exp-ElGamal has small message-space restriction
- CL framework via modifying internal mechanism may result in unknown group order



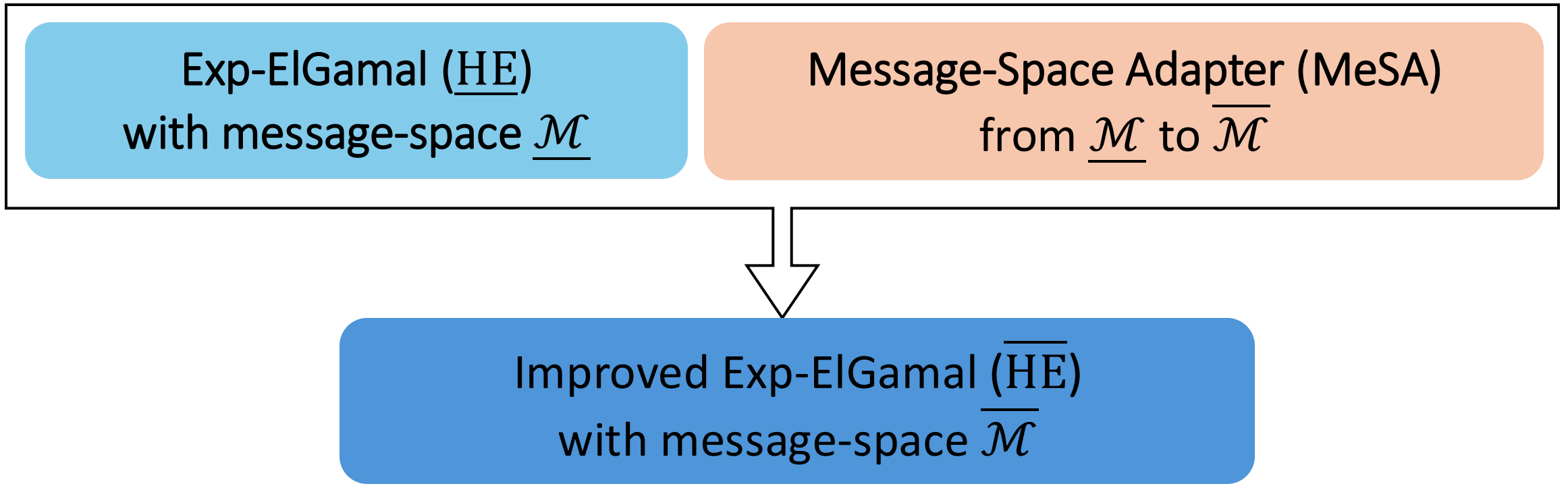
Our Mentality for Constructing “Easy Nails”

The dilemma can be resolved if improved Exp-ElGamal remains easy to be thresholdized



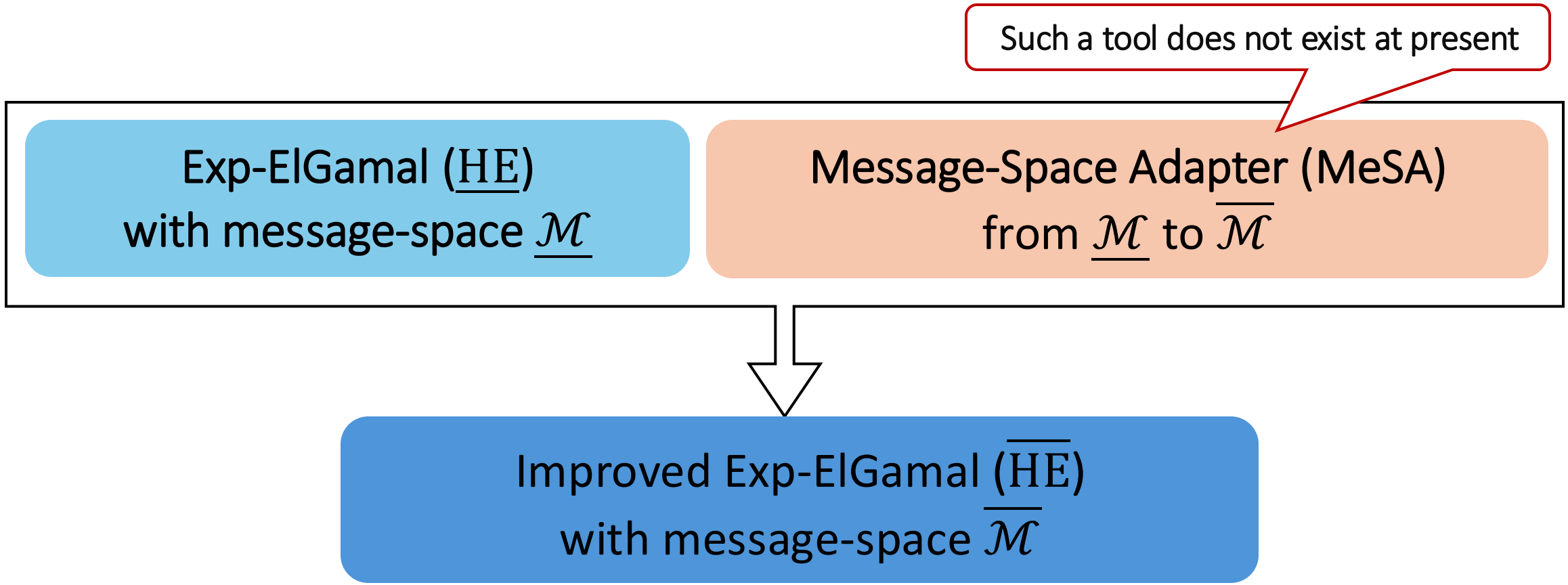
Our Key Idea

Equip Exp-ElGamal with an external adapter that doesn't modify its internal mechanism



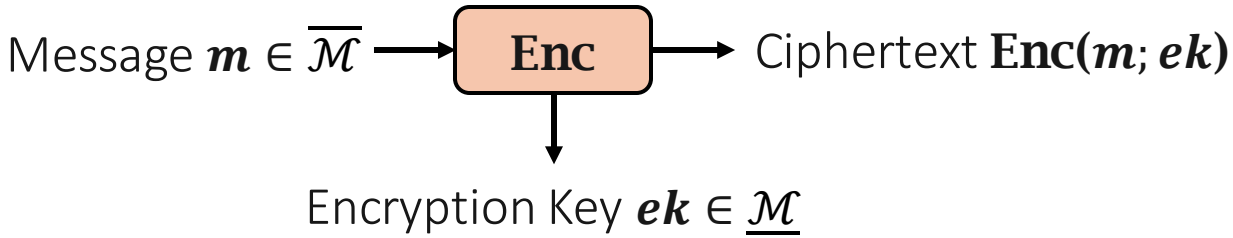
Our Key Idea

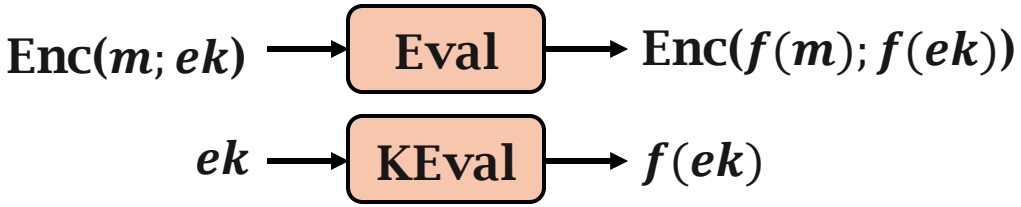
Equip Exp-ElGamal with an external adapter that doesn't modify its internal mechanism

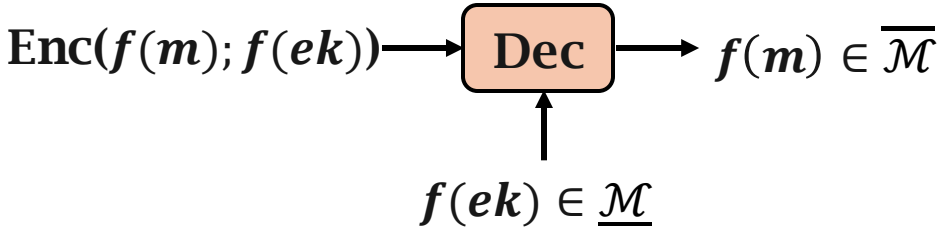


Formulation of MeSA: Algorithms and Requirements

MeSA is an encryption scheme with four main algorithms and five special requirements

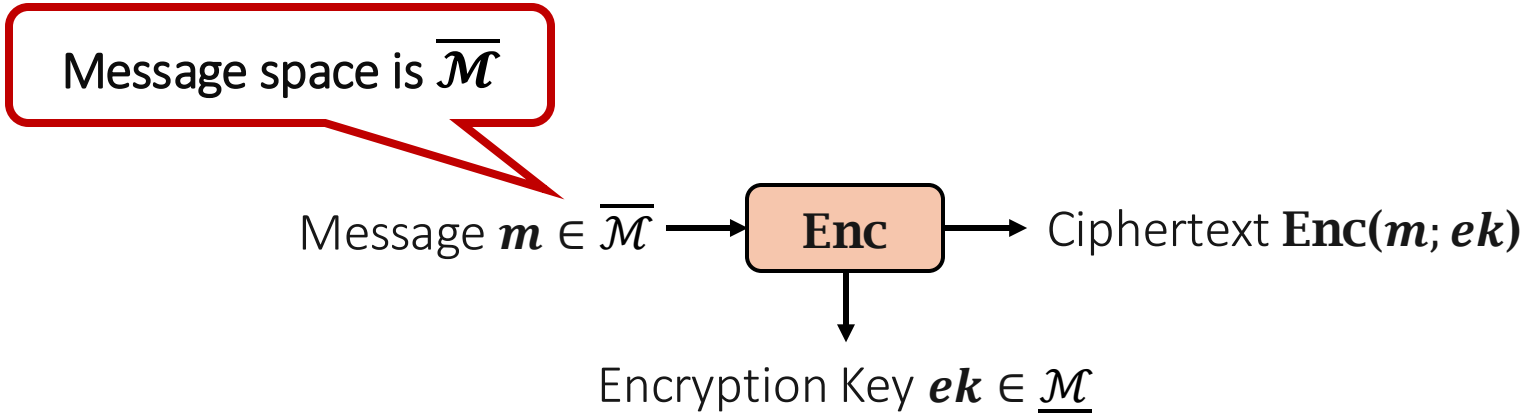


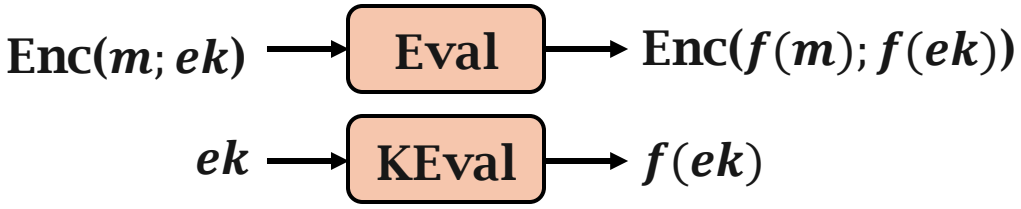


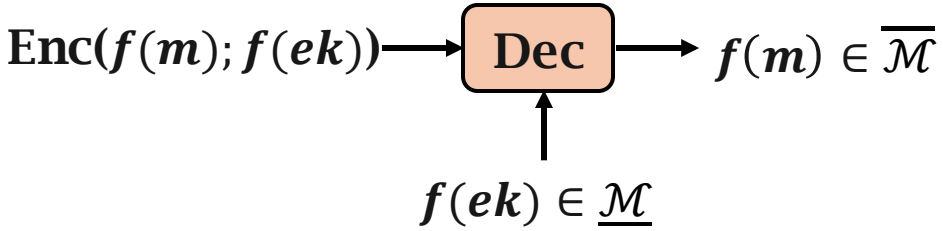


Formulation of MeSA: Algorithms and Requirements

MeSA is an encryption scheme with four main algorithms and five special requirements

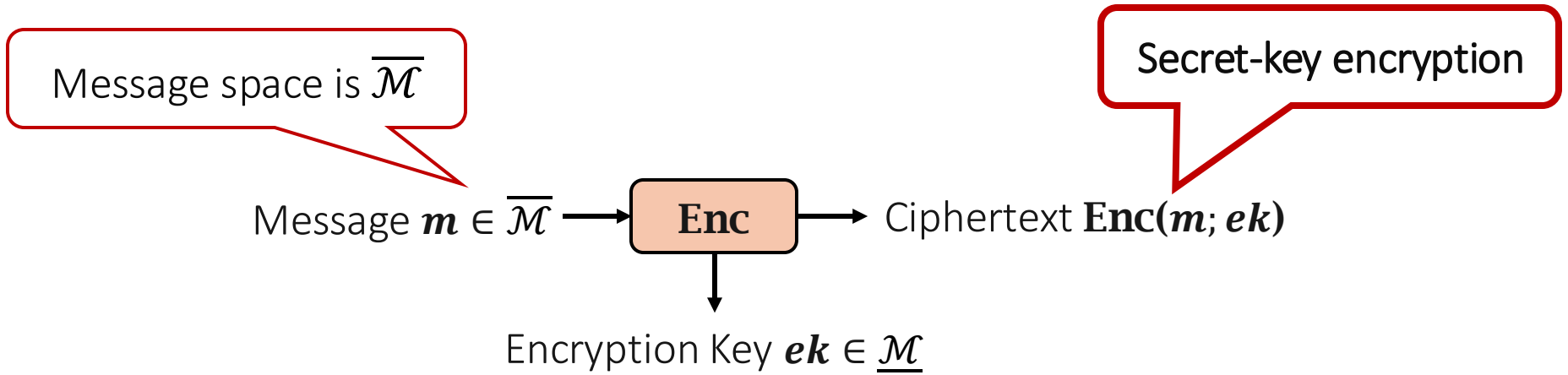


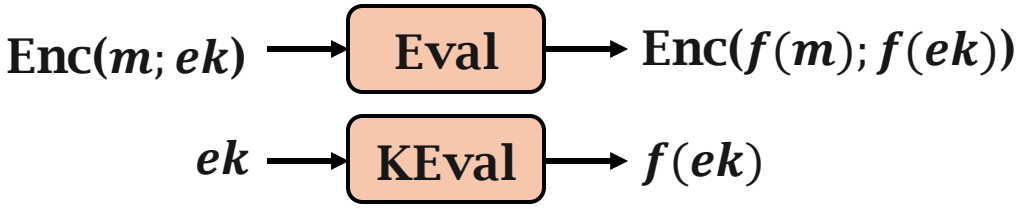


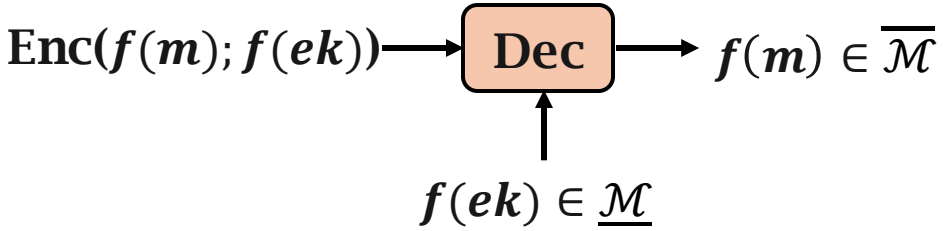


Formulation of MeSA: Algorithms and Requirements

MeSA is an encryption scheme with four main algorithms and five special requirements

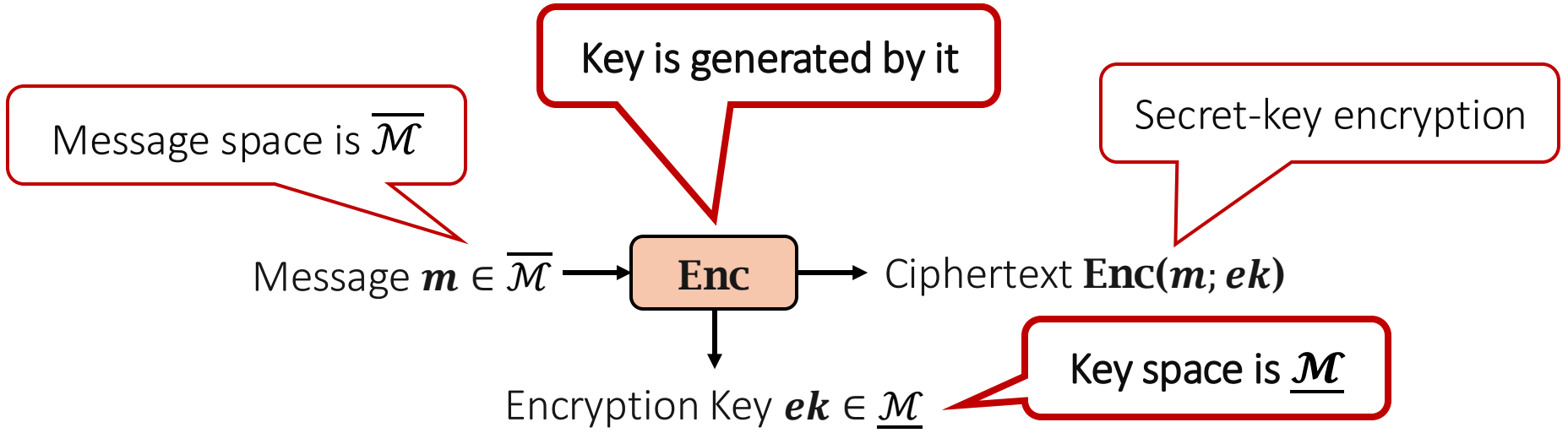


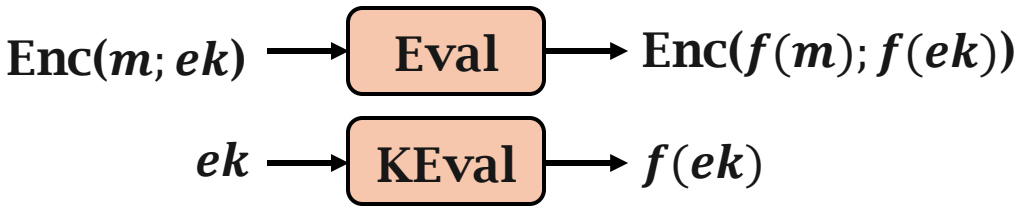


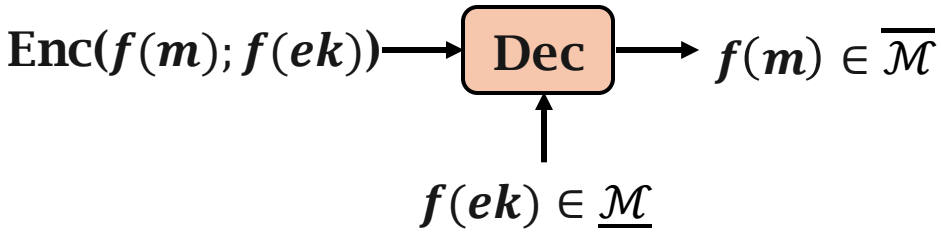


Formulation of MeSA: Algorithms and Requirements

MeSA is an encryption scheme with four main algorithms and five special requirements

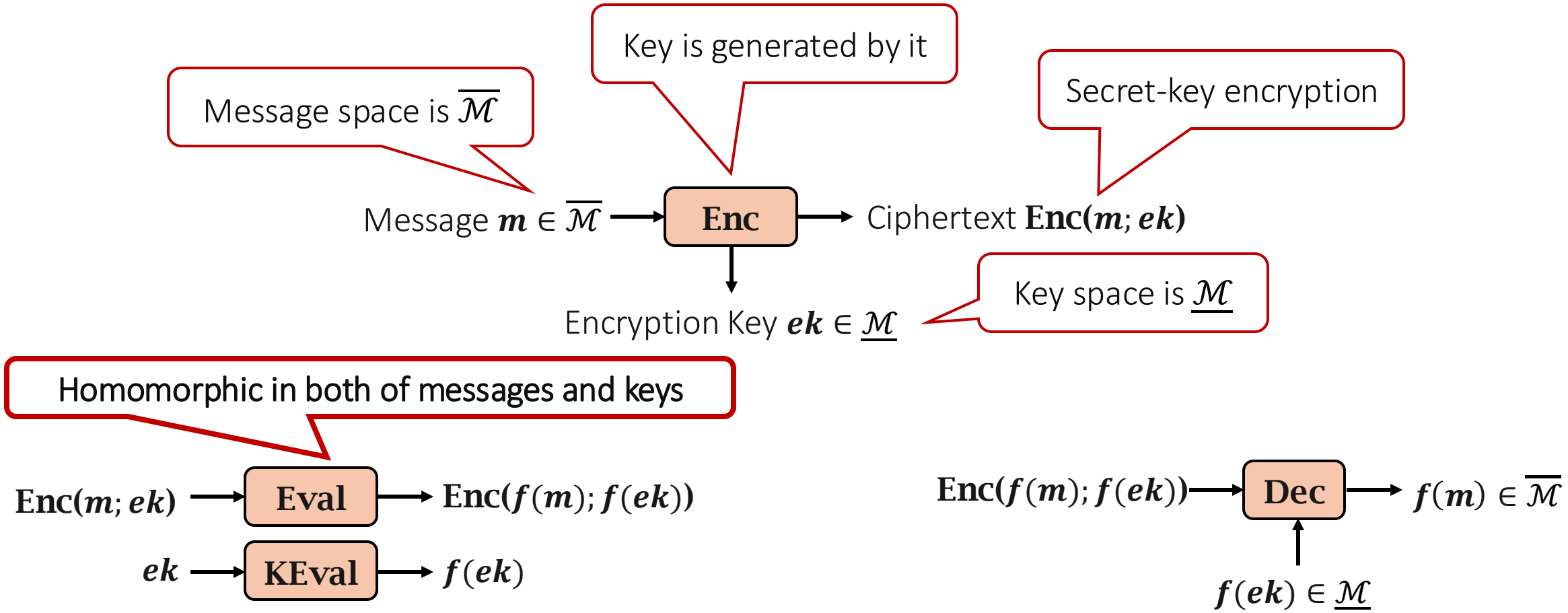






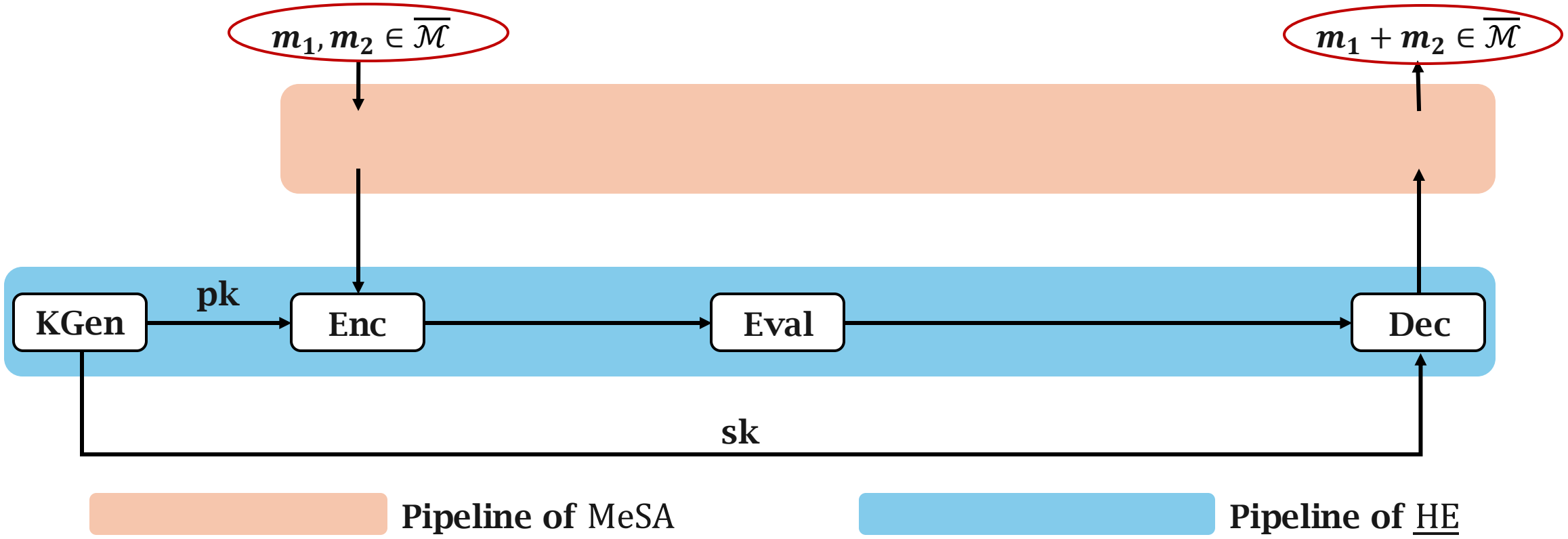
Formulation of MeSA: Algorithms and Requirements

MeSA is an encryption scheme with four main algorithms and five special requirements



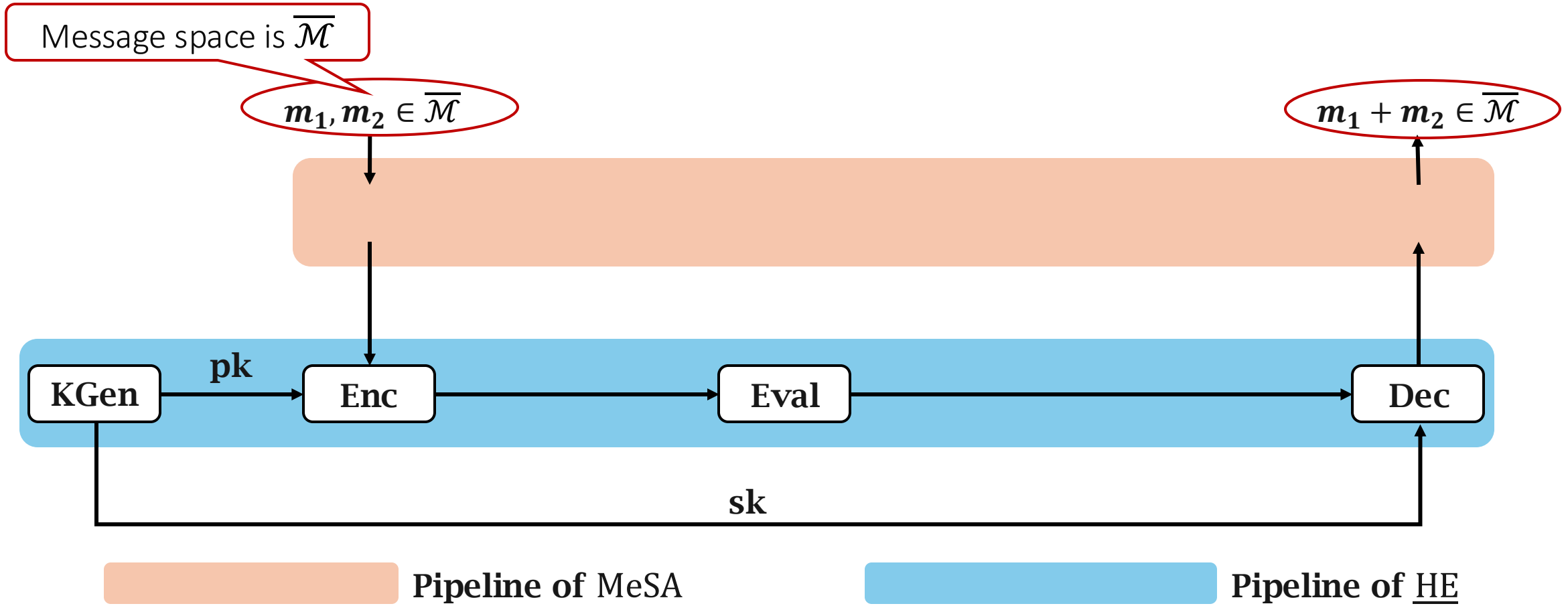
MeSA-Assisted HE Construction: An Illustrative Example

For ease of description, we assume two messages and additive-homomorphism



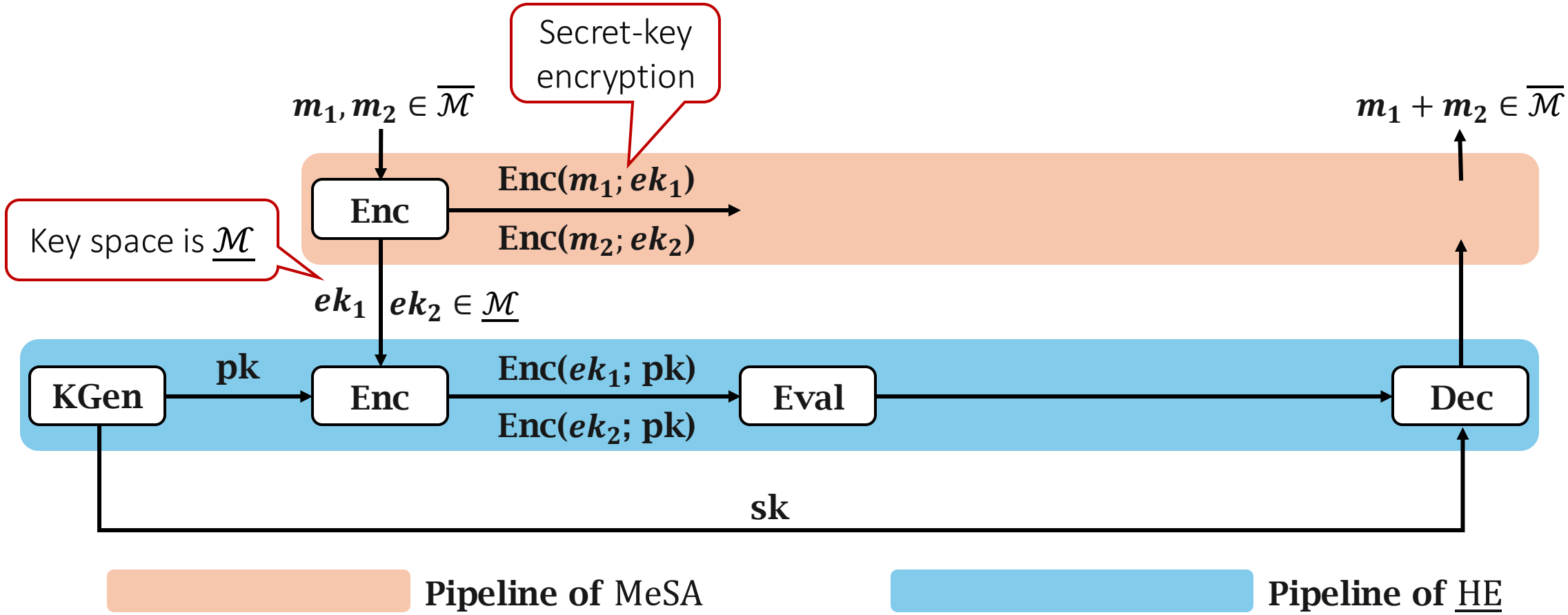
MeSA-Assisted HE Construction: An Illustrative Example

For ease of description, we assume two messages and additive-homomorphism



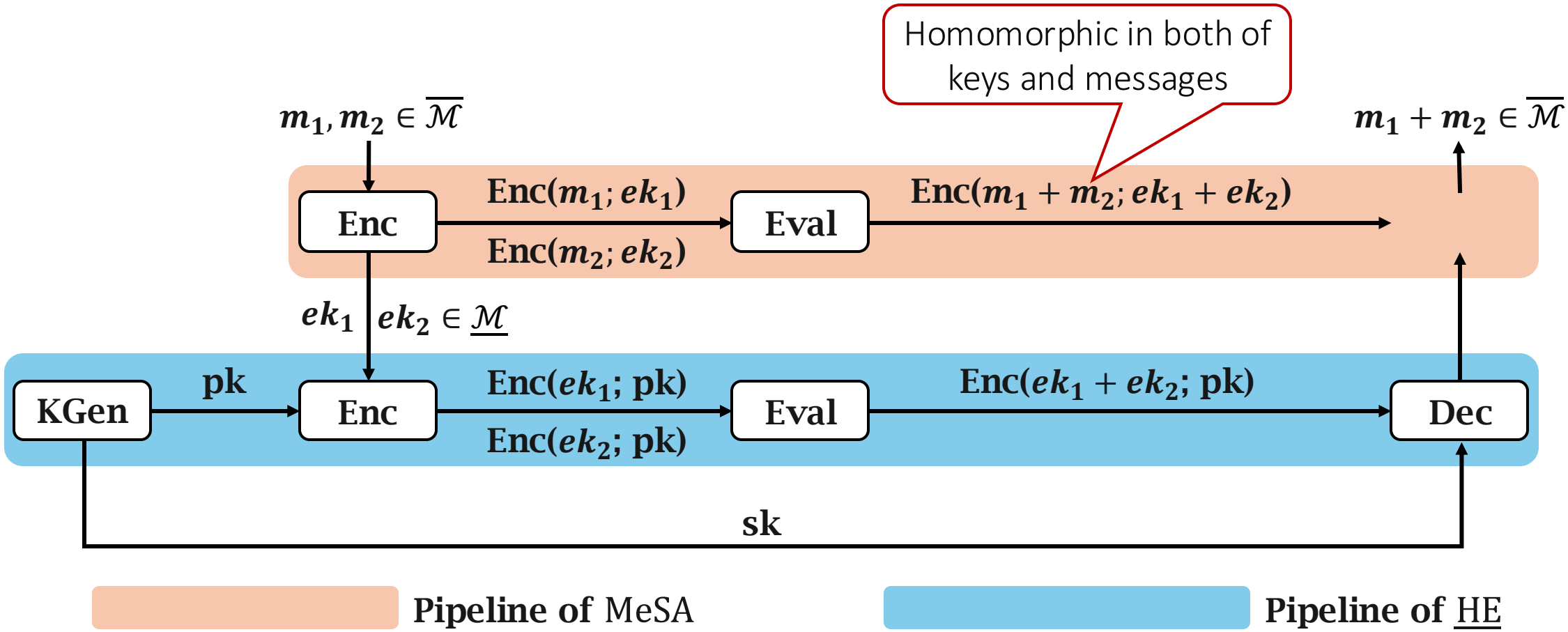
MeSA-Assisted HE Construction: Encryption

Encryption consists of encrypting messages via MeSA and encrypting MeSA's key via HE



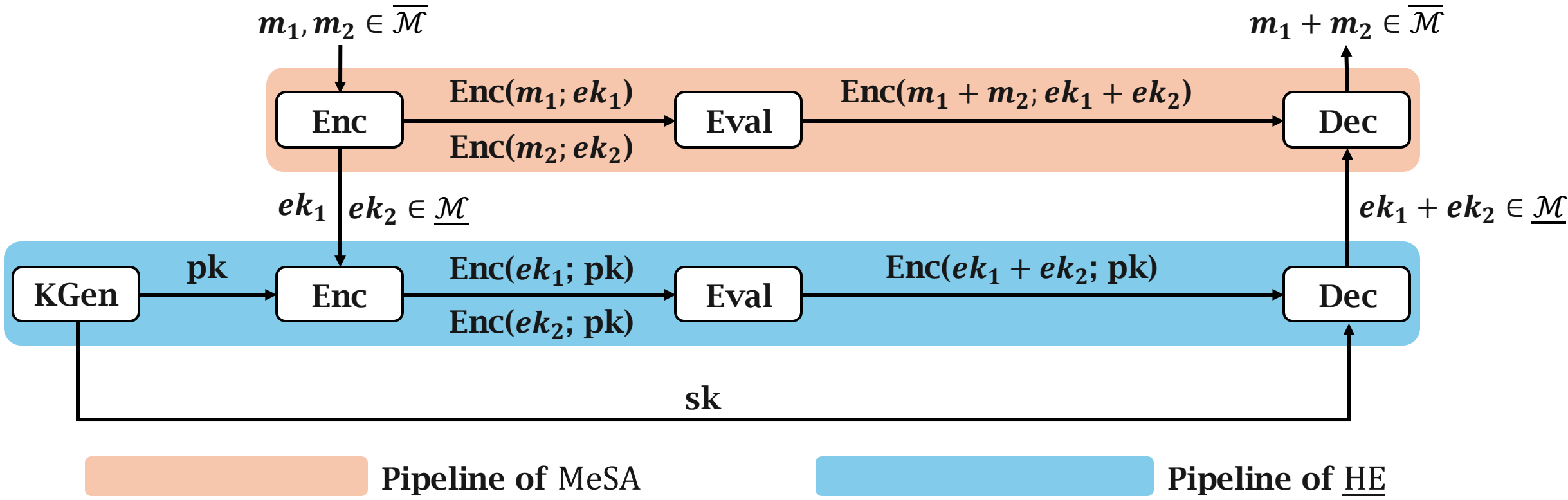
MeSA-Assisted HE Construction: Evaluation

Evaluation consists of evaluating MeSA's ciphertexts and evaluating HE's ciphertexts



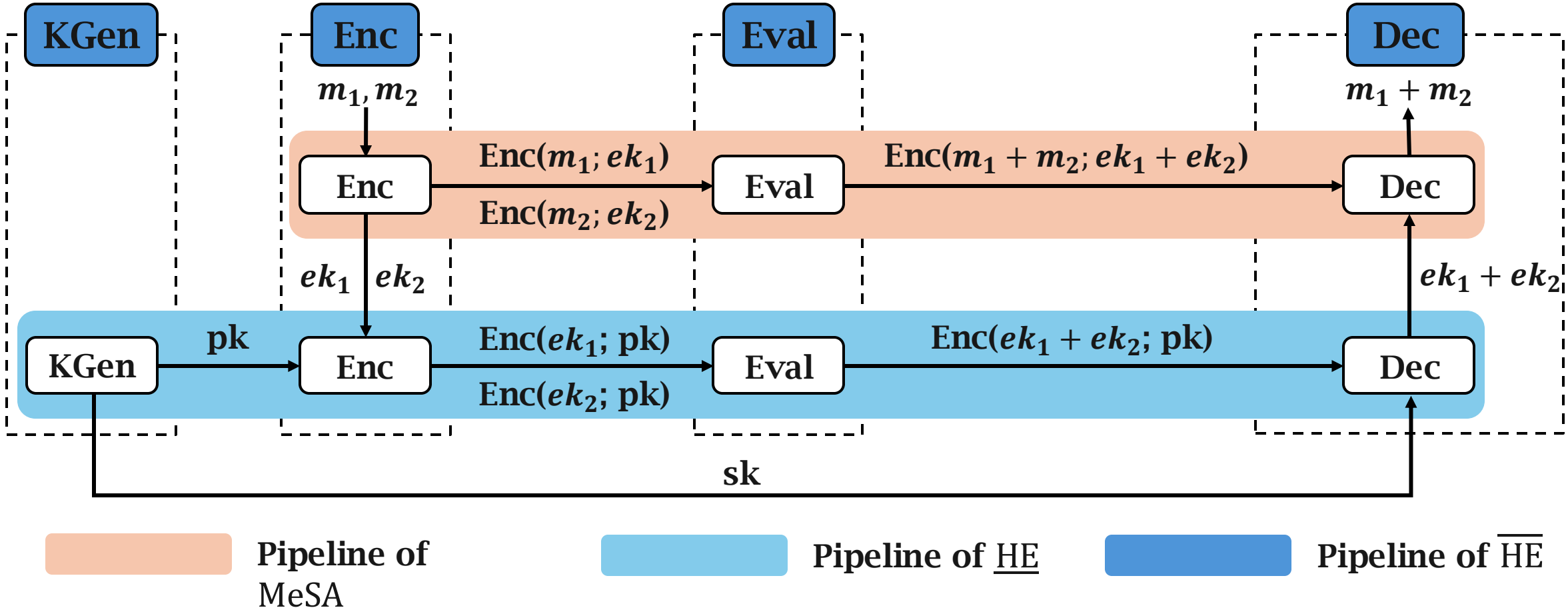
MeSA-Assisted HE Construction: Decryption

Decryption consists of decrypting MeSA's keys via HE and decrypting messages via MeSA



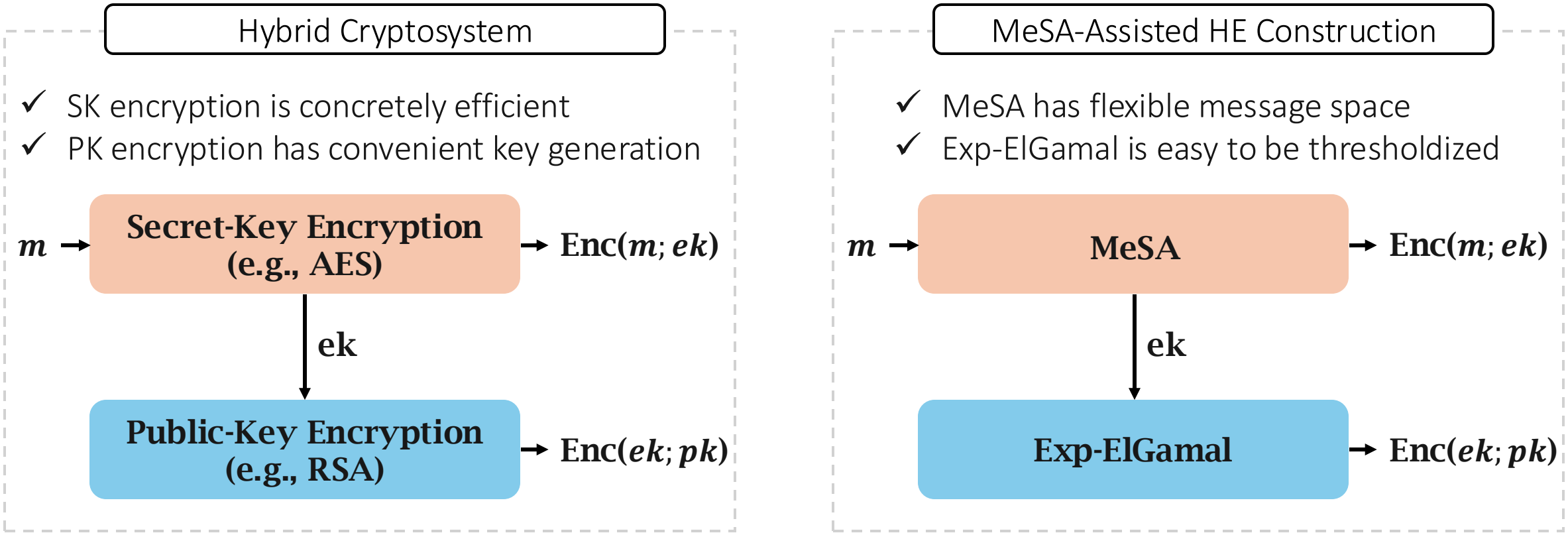
MeSA-Assisted HE Construction: Final Result

$\overline{\text{HE}}$ is obtained by simply combining $\underline{\text{HE}}$ and MeSA together



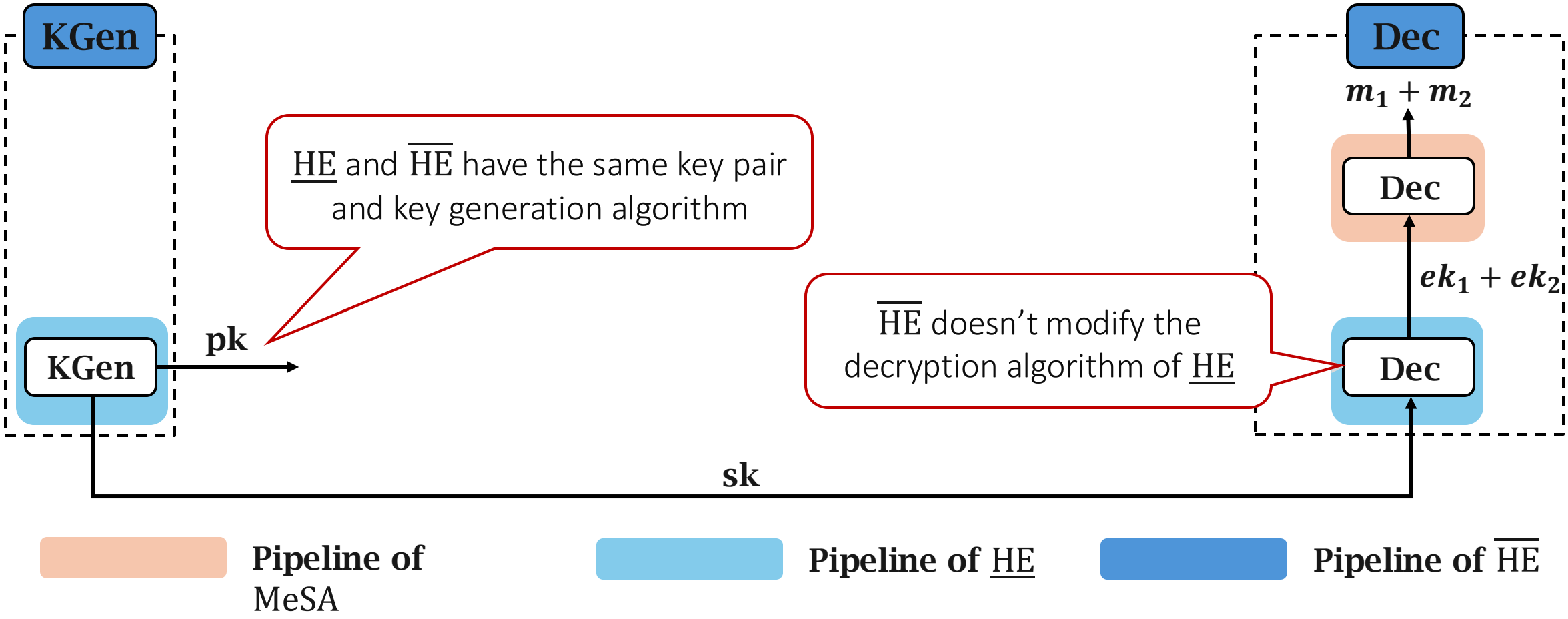
Another Perspective of MeSA: Hybrid Cryptosystem

MeSA is a special type of secret-key encryption with extra requirements on homomorphism



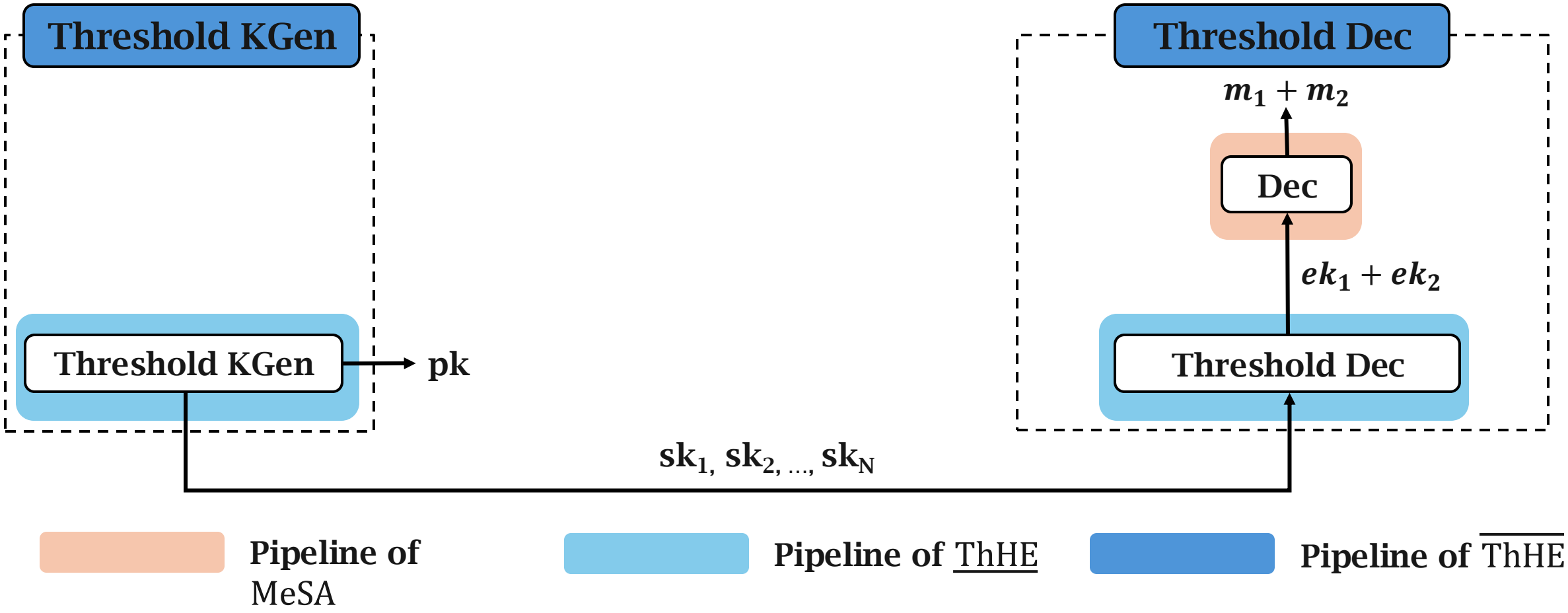
MeSA-Assisted HE Construction: Easy to Be Thresholdized

Any method for thresholdizing $\underline{\mathrm{HE}}$ can directly thresholdize $\overline{\mathrm{HE}}$ due to same key structure



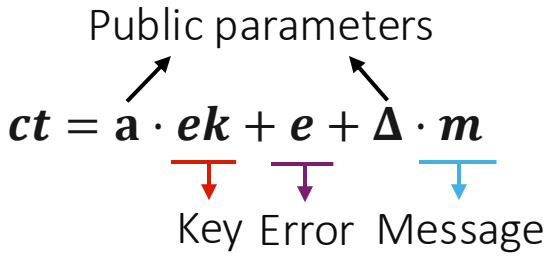
MeSA-Assisted Threshold HE (ThHE) Construction

$\overline{\text{ThHE}}$'s threshold decryption consists of ThHE's threshold decryption and MeSA's decryption



Lattice-Based MeSA Scheme for Exp-ElGamal

Lattice-based secret-key encryption satisfies all requirements of MeSA for Exp-ElGamal



- ✓ Message space is $\overline{\mathcal{M}}$: $\mathbb{Z}_P$, where P can be fairly large
- ✓ Key space is $\underline{\mathcal{M}}$: each key is vector/matrix with small entries (e.g., 0, 1, or -1)
- ✓ Secret-key encryption
- ✓ Key is generated by encryption algorithm
- ✓ Homomorphic in both of messages and keys

$$\alpha_1 \cdot ct_1 + \alpha_2 \cdot ct_2 = \mathbf{a} \cdot (\underbrace{\alpha_1 \cdot ek_1 + \alpha_2 \cdot ek_2}_{\text{Key}}) + (\underbrace{\alpha_1 \cdot e_1 + \alpha_2 \cdot e_2}_{\text{Error}}) + \Delta \cdot (\underbrace{\alpha_1 \cdot m_1 + \alpha_2 \cdot m_2}_{\text{Message}})$$

Lattice-Based MeSA Scheme for Exp-ElGamal

Lattice-based secret-key encryption satisfies all requirements of MeSA for Exp-ElGamal

Public parameters

$$ct = \mathbf{a} \cdot \underbrace{ek}_{\text{Key}} + \underbrace{e}_{\text{Error}} + \Delta \cdot \underbrace{m}_{\text{Message}}$$

- ✓ Message
- ✓ Key spac
- ✓ Secret-Ke
- ✓ Key is ge
- ✓ Homomorphic in both of messages and keys

r -1)

Problem

- The key and error grow rapidly with a multiplicative factor $\mathcal{O}(\alpha)$
- Decryption will fail if key/error exceeds some bound

$$\alpha_1 \cdot ct_1 + \alpha_2 \cdot ct_2 = \mathbf{a} \cdot \underbrace{(\alpha_1 \cdot ek_1 + \alpha_2 \cdot ek_2)}_{\text{Key}} + \underbrace{(\alpha_1 \cdot e_1 + \alpha_2 \cdot e_2)}_{\text{Error}} + \Delta \cdot \underbrace{(\alpha_1 \cdot m_1 + \alpha_2 \cdot m_2)}_{\text{Message}}$$

Controlling Key & Error: Basic Idea

Bit decomposition can convert the homomorphic evaluation from multiplication to addition

Another way to compute $\alpha \cdot m$

Suppose the message space is $\mathbb{Z}_P = \{0,1,2,\dots,P-1\}$ and $P \le 2^L$, then

$$\alpha = \sum_{i=0}^{L-1} \alpha_i \cdot 2^i \pmod{P}$$

$\alpha_i \in \{0,1\}$

$$m \pmod{P}$$
$$2 \cdot m \pmod{P}$$
$$2^2 \cdot m \pmod{P}$$
$$\dots$$
$$2^{L-1} \cdot m \pmod{P}$$

$$\longrightarrow \quad \alpha \cdot m = \sum_{i=0}^{L-1} \alpha_i \cdot (2^i \cdot m) \pmod{P}$$

At most L additions

Controlling Key & Error: Concrete Design

Reduce the growth from $\mathcal{O}(\alpha)$ to $\mathcal{O}(\log \alpha)$ by encrypting multiple versions of messages

An Illustrative Example with Message Space being $\mathbb{Z}_P$

Before Controlling

$$ct = \mathbf{a} \cdot ek + e + \Delta \cdot m \longrightarrow \alpha \cdot ct = \mathbf{a} \cdot (\alpha \cdot ek) + \alpha \cdot e + \Delta \cdot \alpha \cdot m$$

$\mathcal{O}(\alpha)$ growth

After Controlling

$$\begin{cases} ct_1 = \mathbf{a} \cdot ek_1 + e_1 + \Delta \cdot m \\ ct_2 = \mathbf{a} \cdot ek_2 + e_2 + \Delta \cdot 2m \\ \vdots \\ ct_j = \mathbf{a} \cdot ek_j + e_j + \Delta \cdot 2^{j-1} m \\ \vdots \\ ct_L = \mathbf{a} \cdot ek_L + e_L + \Delta \cdot 2^{L-1} m \end{cases} \longrightarrow$$

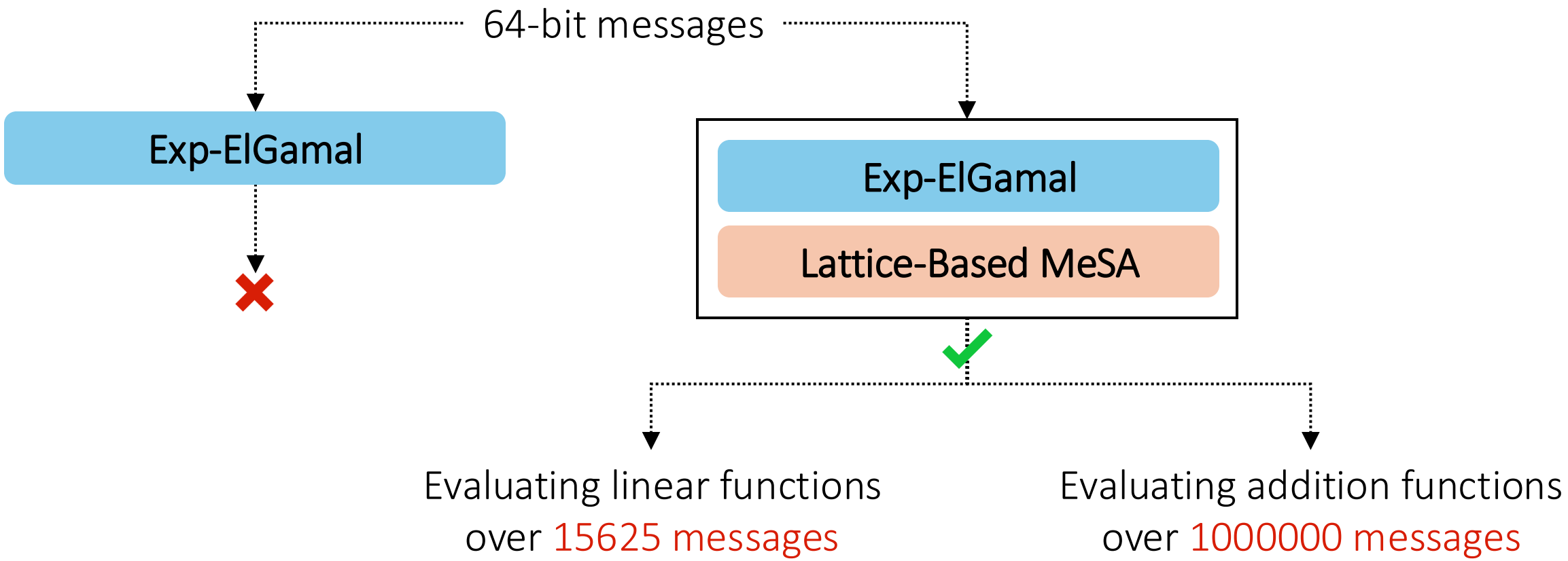
$$(\alpha \cdot ct)_j = \sum_{i=0}^{L-1} \alpha_{i,j} \cdot ct_i$$

$$= \mathbf{a} \cdot \left(\sum_{i=0}^{L-1} \alpha_{i,j} \cdot ek_i \right) + \sum_{i=0}^{L-1} \alpha_{i,j} \cdot e_i + \Delta \cdot \sum_{i=0}^{L-1} \alpha_{i,j} \cdot 2^i m$$

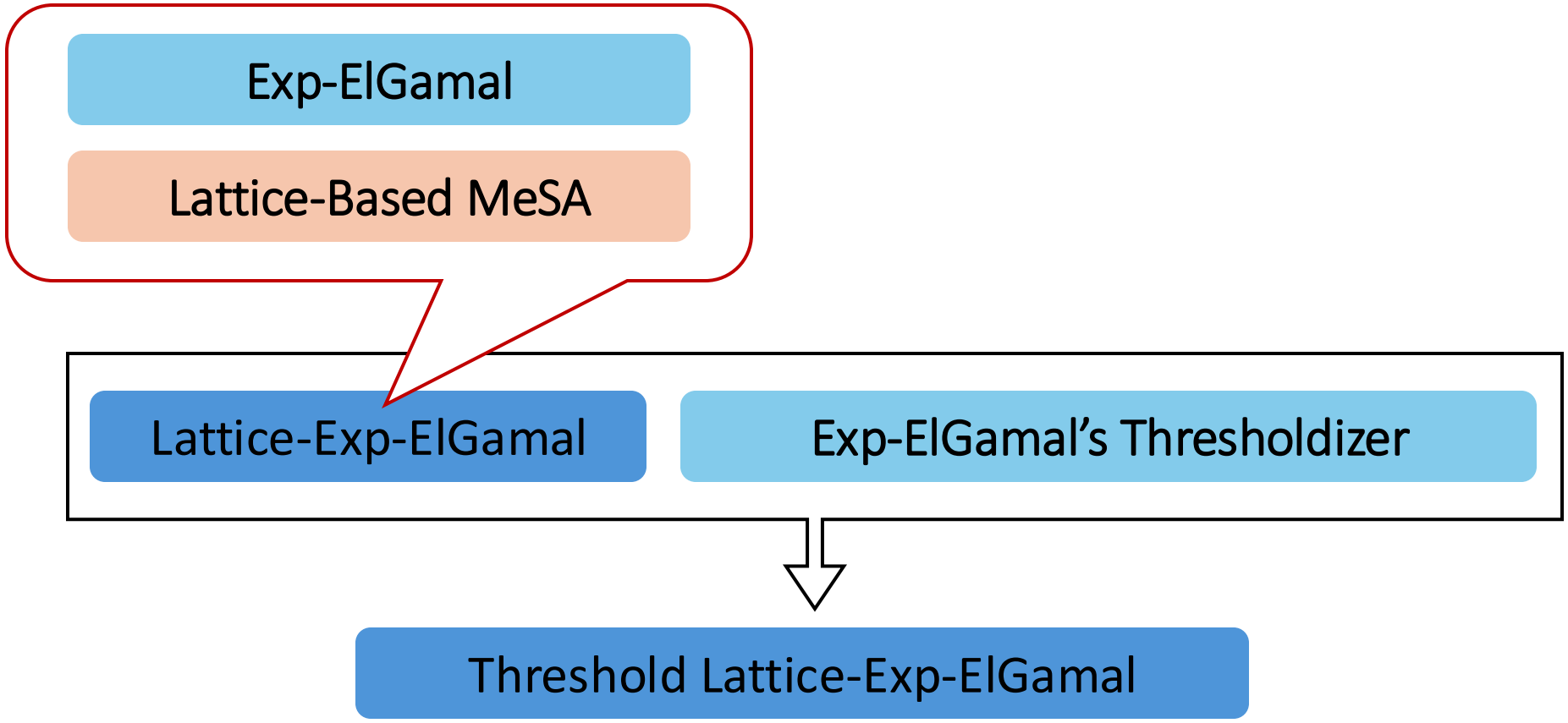
$\mathcal{O}(L) = \mathcal{O}(\log \alpha)$ growth

Controlling Key & Error: Practical Effect

Our lattice-based MeSA can already support real-world applications



Lattice-Exp-ElGamal and Its Threshold Version



Performance Analysis of Threshold Lattice-Exp-ElGamal

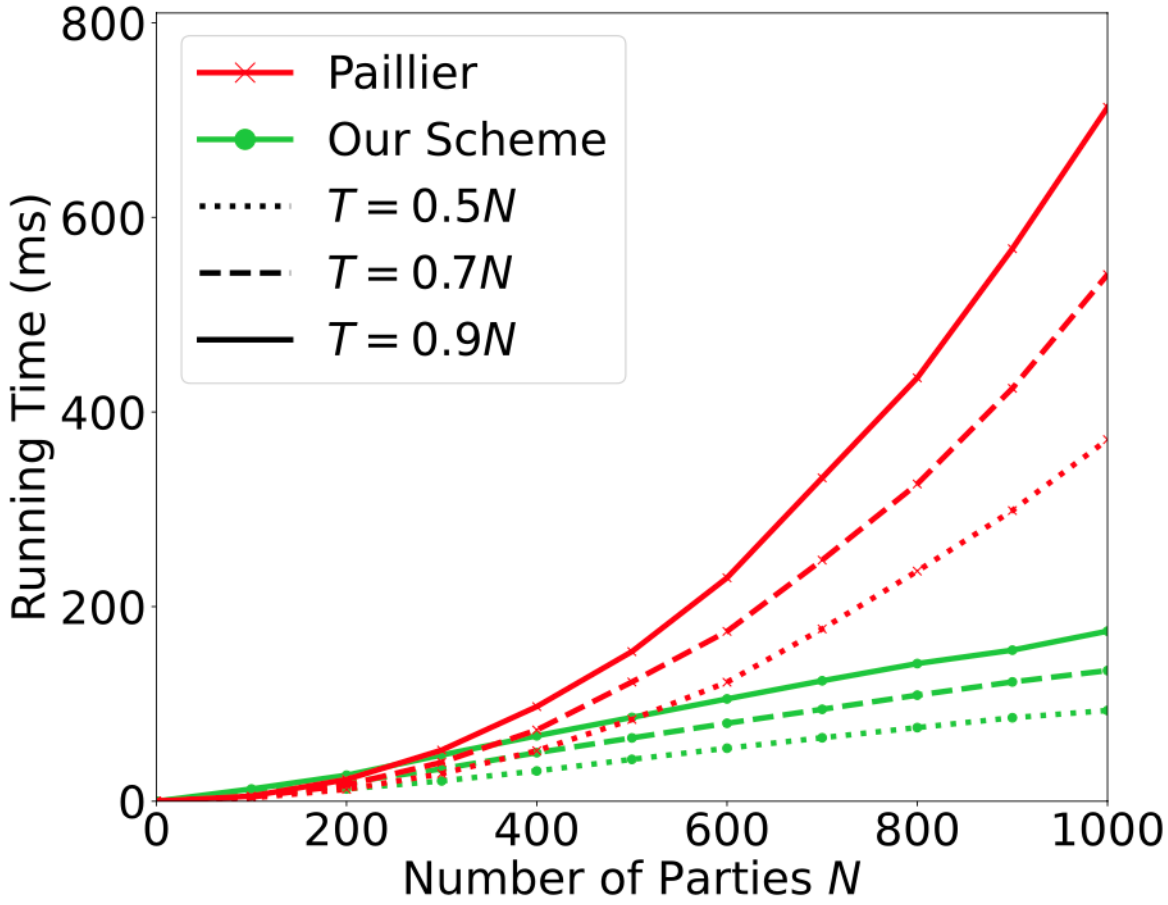
Threshold Lattice-Exp-ElGamal is the first unrestricted ThLHE with quasi-linear complexity

	ThLHE with unknown group order	ThLHE with noisy recovery	Threshold Lattice-Exp-ElGamal
Computation Complexity	$\mathcal{O}(N^2 \log N)$	$\mathcal{O}(N^2 \log N)$	$\boldsymbol{\mathcal{O}(N \log N)}$
Communication Complexity	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(N \log N)$	$\boldsymbol{\mathcal{O}(1)}$

$\boldsymbol{N}$: the number of parties

Performance Analysis of Threshold Lattice-Exp-ElGamal

Threshold Lattice-Exp-ElGamal is the first unrestricted ThLHE with quasi-linear complexity



Performance Analysis of Threshold Lattice-Exp-ElGamal

Threshold Lattice-Exp-ElGamal has even more significant advantages for larger messages

	ThLHE with unknown group order	ThLHE with noisy recovery	Threshold Lattice-Exp-ElGamal
Computation Complexity	$\mathcal{O}(N^2 \log N \cdot M)$	$\mathcal{O}(N^2 \log N \cdot M)$	$\boldsymbol{\mathcal{O}(N \log N + ND + M)}$
Communication Complexity	$\boldsymbol{\mathcal{O}(M)}$	$\mathcal{O}(N \log N \cdot M)$	$\boldsymbol{\mathcal{O}(D)}$

$\approx \mathcal{O}(ND + M)$ for sufficiently-large M

$\approx \mathcal{O}(1)$ for sufficiently-large M

$\boldsymbol{N}$: the number of parties $\boldsymbol{M}$: message size $\boldsymbol{D}$: MeSA's key size

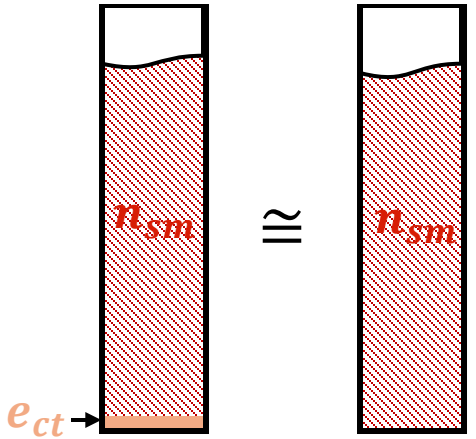
Problems for Thresholdizing Lattice-Based ThFHE

ThFHE decryption should hide the error in ciphertexts, generally with a flooding noise

Cannot be disclosed!

$$\mathbf{Dec}(\mathbf{ct}_m; sk): \quad \boxed{b} = \mathbf{ct}_m \cdot sk \to \Delta \cdot m + e_{ct}$$

Message — Error in ciphertext



Safe to be disclosed!

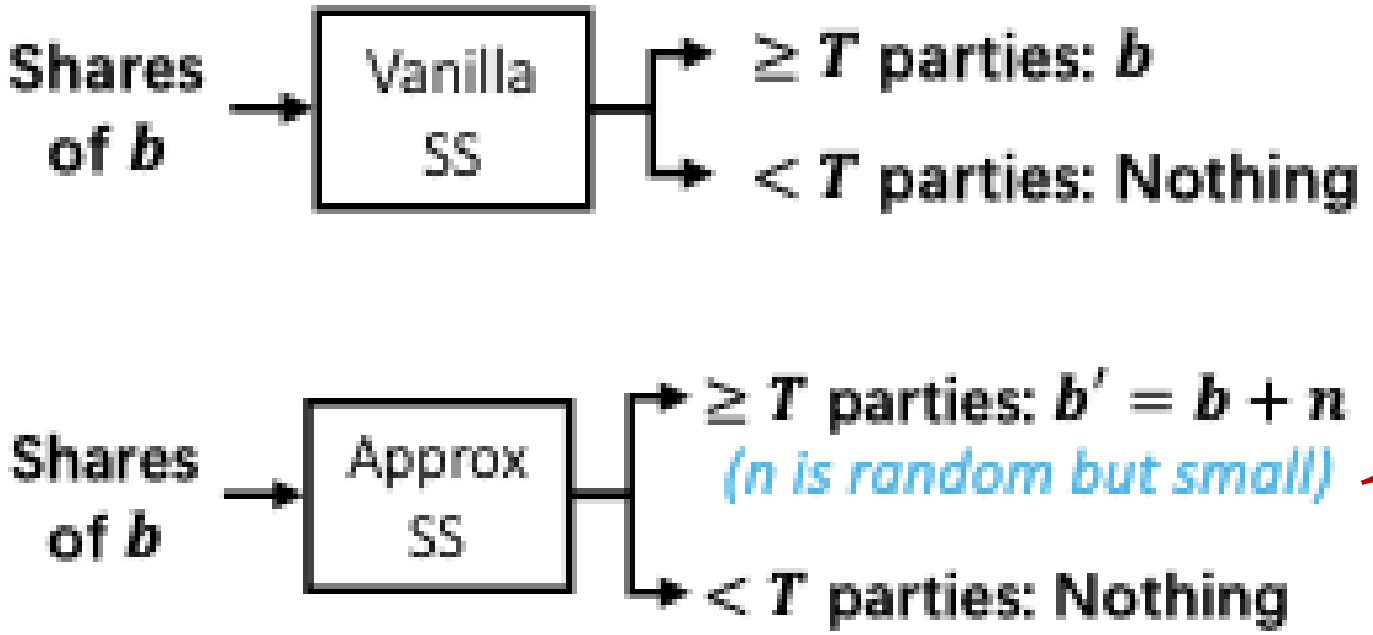
$$\mathbf{Dec}(\mathbf{ct}_m; sk): \quad \boxed{b'} = \mathbf{ct}_m \cdot sk + n_{sm} \to \Delta \cdot m + e_{ct} + n_{sm} \cong \Delta \cdot m + n_{sm}$$

Message — Flooding Noise

Our Previous Work in USENIX Security'25

The core of multi-party FHE design is to construct approximate secret sharing (ApproxSS)

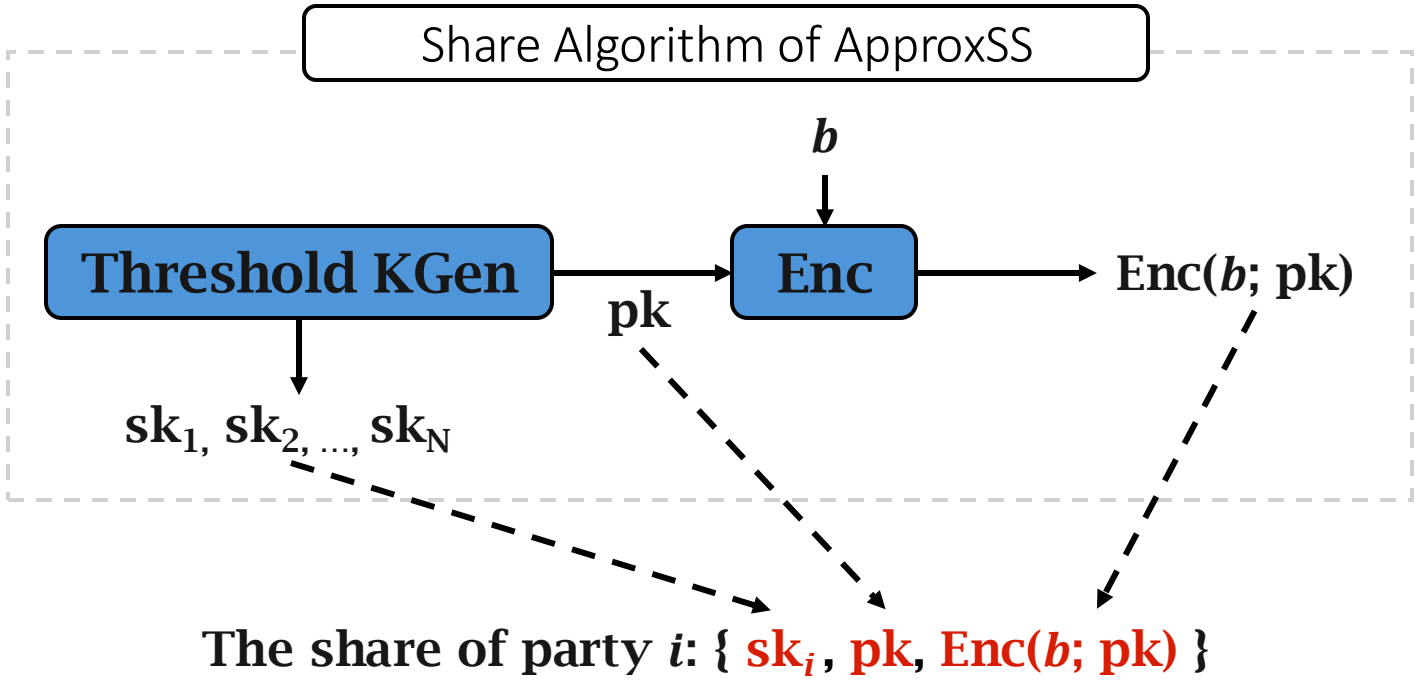
Threshold decryption may take multiple rounds



"Encrypted Share" could be a better idea for ApproxSS

ApproxSS Construction via Threshold Lattice-Exp-ElGamal

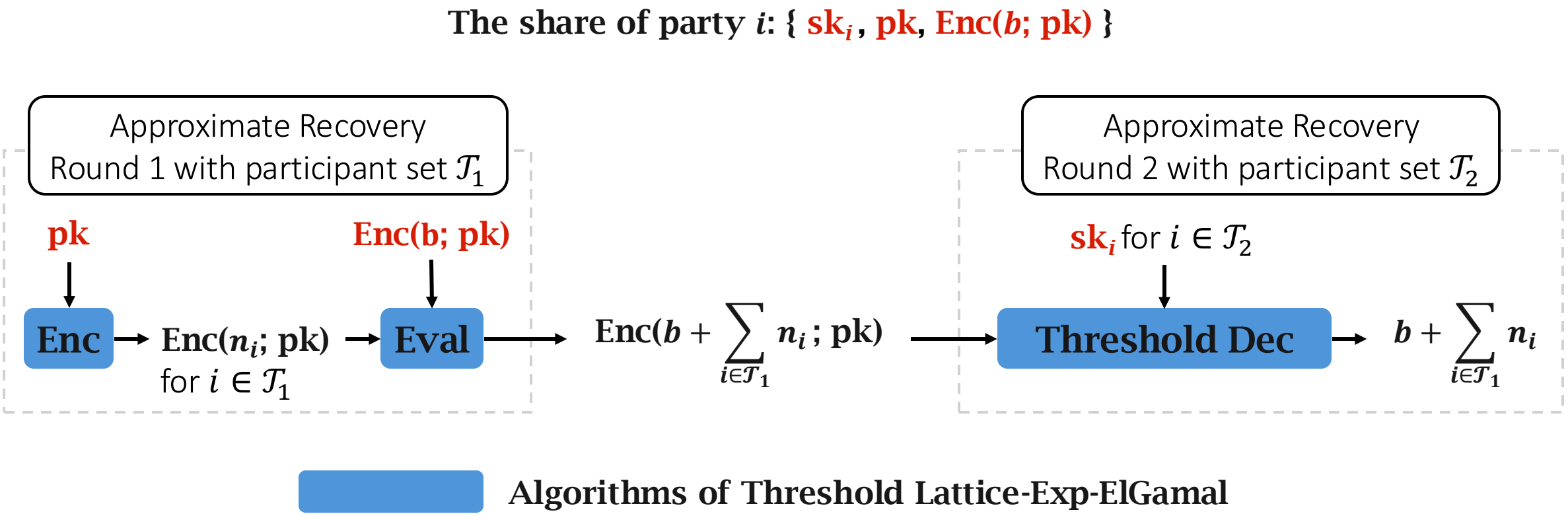
The share algorithm encrypts message b using threshold Lattice-Exp-ElGamal



ApproxSS Construction via Threshold Lattice-Exp-ElGamal

The approximate recovery protocol consists of two rounds

- Round 1: every parties encrypt a small noise $n_i \leftarrow \chi$ and add all ciphertexts together
- Round 2: every parties execute threshold decryption to output approximate message



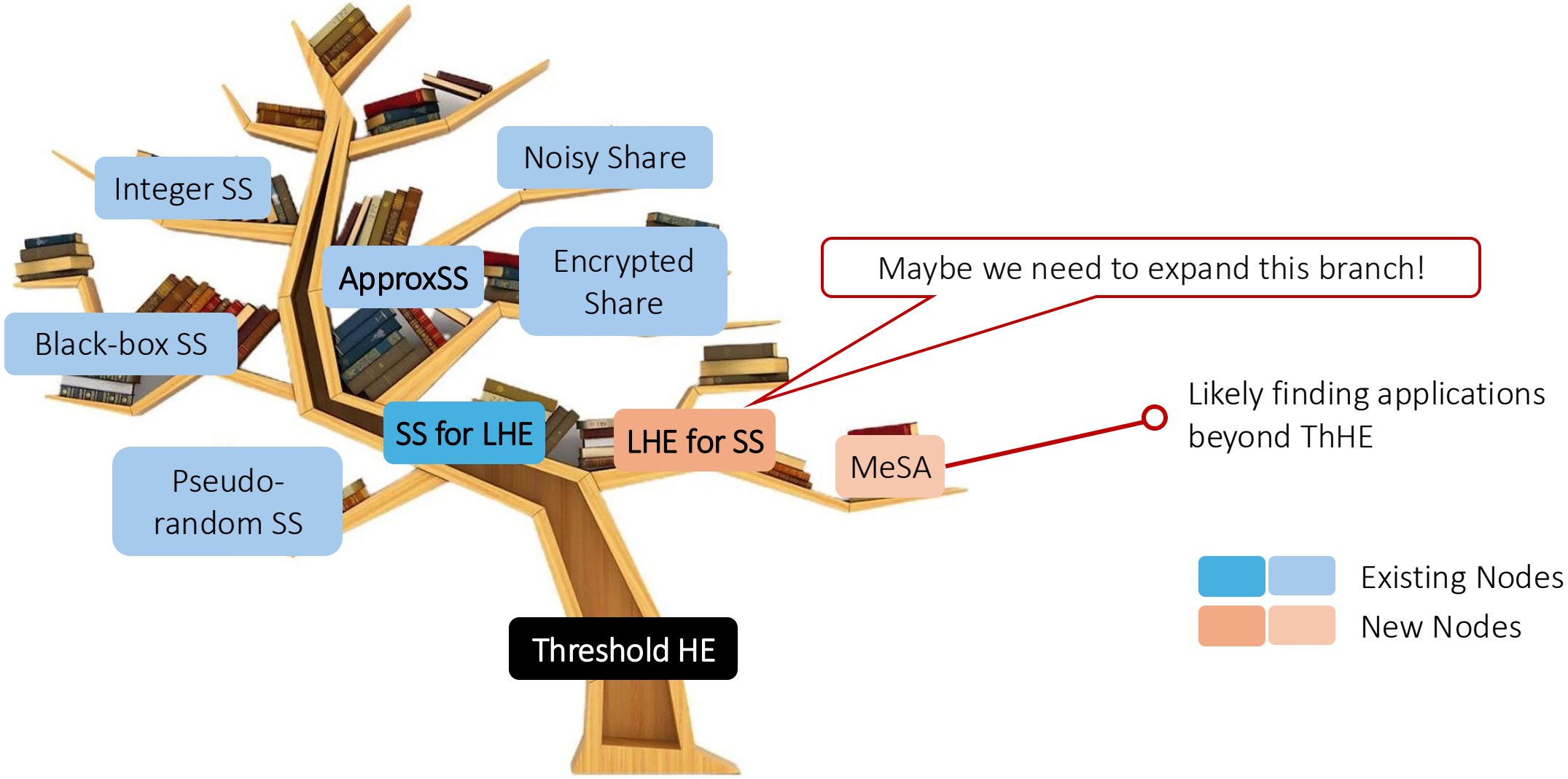
Performance Analysis of Our ThLHE-Based ApproxSS

Our ThLHE-based ApproxSS resolves the multi-party FHE's efficiency-utility dilemma

- Efficiency: Quasi-linear computation complexity and constant communication complexity
- Utility: Can work for any T decryptors

	{0,1}-ApproxSS	Shamir-Based ApproxSS			ThLHE-Based ApproxSS
		Scheme 1	Scheme 2	Scheme 3	
Computation Complexity	$\mathcal{O}(N^{5.2})$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\boldsymbol{\mathcal{O}(N \log N)}$
Communication Complexity	$\mathcal{O}(N^{4.2})$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\boldsymbol{\mathcal{O}(1)}$
Round Number	**1**	**1**	2	2	2

Summary and Discussion



Thank you!

Any questions?