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

Yijia Chang[1], Rongmao Chen[2], Chao Lin[3], Songze Li[4,5], and Xinyi Huang[6]

[1] The Hong Kong University of Science and Technology (Guangzhou)
yijia.chang.crypto@gmail.com
[2] National University of Defense Technology chromao@nudt.edu.cn
[3] Fujian Normal University linchao91@fjnu.edu.cn
[4] Southeast University songzeli@seu.edu.cn
[5] Engineering Research Center of Blockchain Application, Supervision and Management (Southeast University), Ministry of Education
[6] Jinan University xyhuang81@gmail.com

Abstract. Threshold linearly homomorphic encryption (ThLHE) is a useful cryptographic tool for secure computation in multi-party settings, with applications in electronic voting, secure multiparty computation (MPC), and beyond. Although ThLHE offers significant advantages such as low communication overhead, its adoption in modern systems is hindered by a critical dilemma between efficiency and utility. Precisely, existing ThLHE schemes either suffer from high decryption complexity—typically $\mathcal{O}(N^2 \log N)$ or worse for N parties—or impose extra restrictions on the message space or participant set, limiting their practicality in large-scale and dynamic settings.

In this work, we resolve this efficiency-utility dilemma for ThLHE by introducing a novel primitive termed *message-space adapter* (MeSA). By developing a lattice-based MeSA for exponential ElGamal (Exp-ElGamal), we mitigate the small-message restriction of Exp-ElGamal while preserving its efficient threshold decryption. This leads to the design of the first ThLHE scheme that achieves quasi-linear decryption complexity without restrictions on the message space or participant set. We implement a prototype of this new ThLHE scheme and validate the quasi-linear growth of its running time with respect to N.

Beyond resolving this dilemma, we further extend the applications of our new ThLHE scheme. Specifically, we apply it to construct the first multi-party fully homomorphic encryption scheme with quasi-linear computation complexity and constant communication complexity, while supporting arbitrary threshold and dynamic participant set. This demonstrates the extra benefits of our ThLHE scheme with broader applicability.

Keywords: Threshold linearly homomorphic encryption · Message-space adapter · Exponential ElGamal · Multi-party FHE

* Corresponding Authors: Xinyi Huang and Rongmao Chen

1 Introduction

Threshold linearly homomorphic encryption (ThLHE) [11,13] is a special type of encryption with two features. One is that it allows the evaluation of *linear* functions over ciphertexts, without requiring decryption. The other is that it does not require a fully-trustworthy secret-key owner. Instead, a T-out-of-N ThLHE scheme distributes the secret-key among N parties, so that the decryption can succeed if and only if T parties participate. These two features make ThLHE a decent tool for constructing advanced primitives that involve multiple *mutually distrusting* parties, such as secure multi-party computation (MPC) [11,18,37], threshold signature [54], threshold oblivious pseudorandom function [51], etc. Notably, although ThLHE only supports the evaluation of linear functions, it can be employed to implement MPC protocols for *general* functions, as suggested by the pioneering CDN framework [18] and revitalized by the recent You Only Speak Once (YOSO) model [37] for MPC in a large universe (e.g., blockchain).

Despite these fancy features, an extra consideration regarding ThLHE's efficiency arises accordingly: Since the decryption requires the collaboration among multiple parties, its complexity usually grows with the number of parties N. In traditional applications [27,41] with a few parties, this complexity is not a significant factor in efficiency and thus is overlooked before. Nonetheless, modern applications such as federated learning [6] and blockchain [47] involve ever-increasing number of parties, often in the thousands or beyond, making low decryption complexity a preferable and desirable property for ThLHE schemes.

To date, researches have proposed plenty of ThLHE schemes. However, almost all of them [11,13,27] have a decryption with computation complexity[7] being $\mathcal{O}(N^2 \log N)$ or even higher, which becomes the bottleneck for high efficiency. For example, although the CDN framework enhances the general-purpose MPC protocols to achieve linear communication complexity, this $\mathcal{O}(N^2 \log N)$ decryption complexity results in a total running time that still grows with N in the super-quadratic order. Fortunately, there exist three exceptional ThLHE schemes with quasi-linear (computation) complexity, as discussed below.

- *Threshold Exp-ElGamal with small-message restriction.* The well-known ElGamal [30] facilitates a quite simple and efficient design of threshold decryption with almost $\mathcal{O}(N \log N)$ complexity [21]. While the standard ElGamal is multiplicatively homomorphic, the exponential ElGamal (Exp-ElGamal) achieves linearly homomorphic by putting messages in the exponential position [21]. Nonetheless, threshold Exp-ElGamal requires that the messages come from a small, known set. Although this small-message restriction is harmless in some applications like electronic voting, it may inhibit the utilization of threshold Exp-ElGamal when the message space is large. For example, it cannot be used in the CDN framework to generate Beaver triples that are usually in a large field (e.g., $\mathbb{Z}_p$ with a 124-bit modulus p [9]).

[7] Throughout this paper, we set $T = \mathcal{O}(N)$ when evaluating the complexity, as T is usually set to be the proportion of N, e.g., $T = 0.5N$ or $T = 0.9N$.

- *Threshold FHE with full-threshold restriction.* The fully homomorphic encryption (FHE) [36], which allows arbitrary computation over ciphertexts, also supports the evaluation of linear functions. Prior work [3] introduces the first threshold FHE scheme. It is full-threshold (i.e., $T = N$) and thus requires the participation of *all* parties for successful decryption. While this scheme achieves efficient decryption with $\mathcal{O}(N \log N)$ complexity, its utility is limited by its lack of resilience to non-participants, which are frequently observed in real-world systems (e.g., federated learning and blockchain) due to dropout devices or denial-of-service attacks [6,15].
- *Threshold FHE with known-participants assumption.* Another work [42,43] proposes a threshold FHE scheme under *arbitrary* threshold, whose decryption complexity remains $\mathcal{O}(N \log N)$. Although it can work with the existence of non-participants, this scheme relies on an assumption that the set of T participants is available as a prior knowledge before decrypting ciphertexts. Nonetheless, as stated in [2], such a solution has two drawbacks, namely, it adds interactions to make agreement on the participant set and it is not resilient to those non-participants caused by random failures or intentional attacks, resulting in it less attractive or even inapplicable in practice.

In summary, no existing ThLHE scheme achieves $\mathcal{O}(N \log N)$ decryption complexity without imposing restrictions such as "small-message", "full-threshold", and "known-participants". Consequently, ThLHE encounters a dilemma between high efficiency and high utility, hindering its widespread adoption in the real world, especially in scenarios involving a large number of participants.

A deeper look at this efficiency-utility dilemma. The building of a ThLHE scheme usually applies *secret sharing* (SS) technique to adapt an LHE scheme for threshold functionality. However, to fit existing LHE schemes (in particular, their security), the SS schemes either suffer from a high complexity of at least $\mathcal{O}(N^2 \log N)$ or face some utility restrictions. Particularly, for all ThLHE schemes whose security is based on the hardness of factoring RSA integers (e.g., threshold Paillier [27,32,34]), the SS must be performed over a group with *unknown* order for security. To avoid using the group order, existing solutions adopt special SS techniques whose algorithms work universally for any order or even over integers, whereas the computation complexity of recovery algorithm is simultaneously increased to $\mathcal{O}(N^2 \log N)$ or even higher. As for those lattice-based threshold FHE schemes [3,7], each ciphertext contains a small, random *error* that should not be leaked even during the decryption process. To hide this error, existing solutions usually add sufficiently-large smudging noises and rely on special SS techniques (e.g., SS with *small* recovery coefficients or additive SS) to prevent the smudging noises from being too large and causing the decryption to fail. Nonetheless, these SS schemes either have a computation complexity of at least $\mathcal{O}(N^2 \log N)$ [7] for recovery algorithm or impose extra utility restrictions including full-threshold [3] and known-participants [42,43].

To the best of our knowledge, threshold Exp-ElGamal [21] from DDH is the only one whose security does not bring much burden on the efficiency. Particu-

larly, it does not prevent the usage of Shamir SS that can achieve quasi-linear complexity with the assistance of number theoretic transform (NTT) [23,53]. Nonetheless, the "small-message" seems an intrinsic restriction that is binded to its low complexity: Even though the recent CL framework [12] enables the design of DDH-based ThLHE schemes without this restriction [11,13], this framework also faces the problem of unknown group order like factoring-based schemes, resulting in a computation complexity of $\mathcal{O}(N^2 \log N)$.

The motivating question. As discussed above, despite extensive exploration of various SS techniques in prior works, a persistent mismatch remains between the security guarantees of LHE and the efficiency of SS in unrestricted settings. This mismatch has left ThLHE struggling with the efficiency-utility dilemma for the past two decades. This naturally leads to the question of whether this longstanding dilemma could be resolved. Specifically:

Is it possible to design a ThLHE scheme that achieves quasi-linear decryption complexity while eliminating the aforementioned utility restrictions?

1.1 Our Results

In this work, we answer this question in affirmative by developing the first ThLHE scheme with quasi-linear decryption complexity but no restrictions of "small-message", "full-threshold", and "known-participants". Unlike prior works who explore SS schemes that fit the security of existing LHE schemes, we consider how to devise an LHE scheme that can use the efficient Shamir SS without the aforementioned utility restrictions. At the core of our construction is a novel primitive termed *message-space adapter* (MeSA): By plugging a customized MeSA into Exp-ElGamal, we obtain a new LHE scheme that not only mitigates the small-message restriction but also enables the usage of Shamir SS like Exp-ElGamal. Below we summarize our contributions and main results.

1. *Formulation of message-space adapter.* We formulate the notion of message-space adapter (MeSA) and define its properties including compactness, correctness, and security. Intuitively, MeSA is a special type of encryption scheme that can switch the message space of an HE scheme to a target space, without modifying the key structure of this HE scheme. To the best of our knowledge, this is the first work that formulates a primitive that can achieve such a functionality.
2. *MeSA-assisted (threshold) HE construction for target message-space.* We show that we can construct a new HE scheme with target message space (say $\overline{\mathcal{M}}$) by combining an existing HE scheme with message space $\underline{\mathcal{M}}$ and a MeSA scheme from $\underline{\mathcal{M}}$ to $\overline{\mathcal{M}}$ together. Furthermore, any techniques for thresholdizing the existing HE scheme can be directly used for constructing the threshold version of new HE scheme. We believe this construction can be of independent interest for other applications. For example, it can be used to construct (threshold) LHE whose message space is $\mathbb{Z}/2^k\mathbb{Z}$ for some integer k [13], which gains lots of interests recently due to its CPU-friendly features.

3. *A new ThLHE scheme that resolves the efficiency-utility dilemma.* We develop a lattice-based MeSA scheme for Exp-ElGamal. This MeSA scheme, together with the MeSA-assisted (threshold) HE construction, enables us to obtain a novel LHE scheme called Lattice-Exp-ElGamal and its threshold version. Particularly, the threshold Lattice-Exp-ElGamal enjoys $\mathcal{O}(N \log N)$ decryption complexity without the aforementioned utility restrictions, which resolves the efficiency-utility dilemma of ThLHE. We also implement a prototype of threshold Lattice-Exp-ElGamal and validate the quasi-linear growth of its running time with respect to N.
4. *A new application of ThLHE: efficient multi-party FHE design.* To highlight the benefits from our work, we suggest a new application of ThLHE, namely the design of efficient *(single-key) multi-party FHE* (MFHE) [42,45]. MFHE can be regarded as an extension of threshold FHE, where the decryption is generalized to a multi-round (rather than one-round) protocol. By applying our threshold Lattice-Exp-ElGamal, we construct the first concretely-efficient MFHE scheme with quasi-linear computation complexity and constant communication complexity under arbitrary threshold and dynamic participants. For comparison, no existing ThLHE schemes can achieve this effect. Particularly, threshold Exp-ElGamal does not apply, as the messages in this application are MFHE ciphertexts and must be sufficiently large.

1.2 Technical Overview

We start from the exponential ElGamal (Exp-ElGamal), an LHE scheme with restricted message space. Particularly, given two coefficients α_1, α_2 and the encryption of messages m_1, m_2, the evaluation algorithm of Exp-ElGamal can output a valid encryption of $\alpha_1 m_1 + \alpha_2 m_2$. The Exp-ElGamal is also shown to have low complexity regarding its threshold decryption. Nonetheless, to reveal the $\alpha_1 m_1 + \alpha_2 m_2$ from its ciphertext, the decryption algorithm of Exp-ElGamal requires to solve a discrete logarithm. Since the discrete logarithm is generally hard to solve, to make the decryption practical, applying Exp-ElGamal usually requires that the output $\alpha_1 m_1 + \alpha_2 m_2$ is from a known set with small size.

Idea 1: Message-space adapter. Our first goal is to mitigate this small-message restriction. Since the small-message restriction is determined by the underlying hardness problem (i.e., solving the discrete logarithm), altering the message space is a non-trivial task and generally necessitates modifications to the internal mechanism of the Exp-ElGamal. The well-known CL framework [12] can be regarded as the preceding exploration along this direction. Still, this framework also has the unknown group order and thus its threshold versions [11,13] still face the $\mathcal{O}(N^2 \log N)$ complexity.

To address this issue, we adopt a different idea, namely, to introduce another type of encryption schemes termed *message-space adapter* (`MeSA`). By combining `MeSA` and an existing $\underline{\texttt{HE}}$ scheme with non-satisfactory message space $\underline{\mathcal{M}}$ (e.g, Exp-ElGamal), we expect to obtain a new $\overline{\texttt{HE}}$ scheme with target message space $\overline{\mathcal{M}}$, while the internal mechanism of original $\underline{\texttt{HE}}$ scheme is maintained.

Nonetheless, there are no known primitives that can achieve this functionality. In this work, we suggest that MeSA should fulfill the following requirements as a sufficient (though not sure to be necessary) condition to actualize this idea:

- **R1**: It is a secret-key encryption scheme over message space $\overline{\mathcal{M}}$;
- **R2**: Its encryption key space is $\underline{\mathcal{M}}$, from which the encryption algorithm can sample keys for each message individually, just like one-time pad;
- **R3**: MeSA is homomorphic in both of the messages and the encryption keys, with respect to some function family $\mathcal{F}$. For example, if $\mathcal{F}$ only contains the "addition" function (+), MeSA should support to compute a valid encryption of $m_1 + m_2$ under encryption key $\mathsf{ek}_1 + \mathsf{ek}_2$, from the encryption of m_1 under key ek_1 and the encryption of m_2 under encryption key ek_2.

We formulate MeSA under the above requirements in Section 3.1, along with the definition of its correctness and security. Readers may find that these requirements are somewhat similar to the concept of homomorphic encryption or functional encryption. We argue MeSA is a new primitive and highlight its differences with existing primitives in Remark 1.

Next, we show how the above requirements enable MeSA to construct a new $\overline{\underline{\mathsf{HE}}}$ scheme. We present how $\overline{\underline{\mathsf{HE}}}$ works below and illustrate its pipeline in Fig. 1. For ease of description, we assume there are two messages $m_1, m_2 \in \overline{\mathcal{M}}$ and $\mathcal{F}$ only contains the "addition" function.

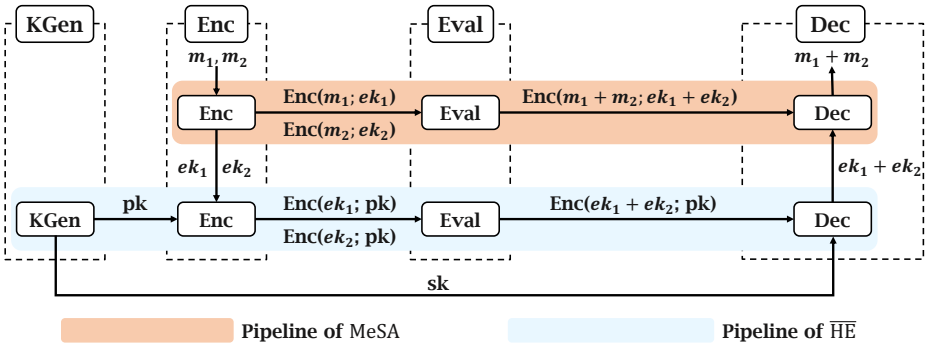

Fig. 1. An illustration of $\overline{\underline{\mathsf{HE}}}$ pipeline for evaluating $m_1 + m_2$ over their ciphertexts. The blue frame and the red frame depict the pipelines of $\underline{\mathsf{HE}}$ and MeSA, respectively.

- *Key Generation.* This is exactly the same as that of $\underline{\mathsf{HE}}$ and outputs the public/secret key pair of $\underline{\mathsf{HE}}$, say, (pk,sk).
- *Encryption.* This contains two sequential steps: 1) using MeSA to encrypt m_1 (resp. m_2) with encryption key ek_1 (resp. ek_2) and 2) using $\underline{\mathsf{HE}}$ (and pk) to encrypt ek_1 and ek_2, respectively. The output ciphertext of each message contains a MeSA ciphertext and an $\underline{\mathsf{HE}}$ ciphertext. We note that the **R1** and **R2** guarantee the feasibility of these two steps. **R1** says that the m_1, m_2 are in the message space of MeSA. By **R2**, ek_1 and ek_2 are generated during the encryption process. Hence, they are available for further encryption by $\underline{\mathsf{HE}}$.

– *Evaluation.* This consists of the evaluation algorithms of $\mathtt{MeSA}$ and $\underline{\mathtt{HE}}$ that deal with their respective ciphertexts. **R3** guarantees that the output ciphertext consists of 1) a valid $\mathtt{MeSA}$ ciphertext of $m_1 + m_2$ under encryption key $\mathsf{ek}_1 + \mathsf{ek}_2$ and 2) a valid $\underline{\mathtt{HE}}$ ciphertext of $\mathsf{ek}_1 + \mathsf{ek}_2$.
– *Decryption.* This contains two sequential steps in the *reverse*-order of encryption: using $\underline{\mathtt{HE}}$ to decrypt $\mathsf{ek}_1 + \mathsf{ek}_2$ with key sk and using $\mathtt{MeSA}$ to decrypt $m_1 + m_2$ with key $\mathsf{ek}_1 + \mathsf{ek}_2$.

In Section 3.2, we formally prove the correctness (resp. security) of $\overline{\mathtt{HE}}$ based on the correctness (resp. security) of $\underline{\mathtt{HE}}$ and $\mathtt{MeSA}$.

We remark that in $\overline{\mathtt{HE}}$, its public/secret key pair does not directly encrypt the messages. It is in fact the $\mathtt{MeSA}$ who handles the messages, while the $\underline{\mathtt{HE}}$ deals with the encryption keys of $\mathtt{MeSA}$. This is somewhat similar to the idea of hybrid cryptosystem that combines the convenient key generation of public-key encryption with the efficiency of secret-key encryption. Still, our construction has a different goal, i.e., it aims to combine the *efficient threshold decryption* of public-key Exp-ElGamal with the *flexible message space* of secret-key MeSA. Consequently, the formulation of MeSA is more complicated than vanilla secret-key encryptions to reach this goal. We left other formulations of MeSA that can achieve the same functionality as an interesting problem for futher work.

The decoupling of messages and public/secret key pair also facilitates the design of threshold HE (ThHE) based on $\overline{\mathtt{HE}}$. Generally speaking, the extension of an HE scheme to the T-out-of-N ThHE requires two operations. One is to modify the key generation algorithm to distribute the secret key into N shares. The other is to decompose the decryption into two algorithms: partial decryption and final decryption. The partial decryption computes a *decryption share* from the ciphertext and a secret key share, while the final decryption reveals the message from any T decryption shares. Recall that the construction of $\overline{\mathtt{HE}}$ does not modify the key structure and the algorithms of $\underline{\mathtt{HE}}$. Hence, any techniques behind threshold $\underline{\mathtt{HE}}$ (i.e., $\underline{\mathtt{ThHE}}$) can be directly applied to construct threshold $\overline{\mathtt{HE}}$ (i.e., $\overline{\mathtt{ThHE}}$), as shown below:

– *Key Generation.* Due to the same key structure, this invokes the key generation algorithm of $\underline{\mathtt{ThHE}}$ to output pk and secret key shares $\{\mathsf{sk}_i\}_{i\in[N]}$.
– *Partial Decryption.* This invokes the partial decryption algorithm of $\underline{\mathtt{ThHE}}$ to output a decryption share ds_i.
– *Final Decryption.* This also invokes the final decryption algorithm of $\underline{\mathtt{ThHE}}$ to generate $\mathsf{ek}_1 + \mathsf{ek}_2$ from any T decryption shares. After then, it uses $\mathtt{MeSA}$ to decrypt $m_1 + m_2$ with the key $\mathsf{ek}_1 + \mathsf{ek}_2$.

In the above construction, only the algorithms of $\underline{\mathtt{ThHE}}$ involve the collaboration of multiple parties, while the algorithms of $\mathtt{MeSA}$ is the same as in traditional $\overline{\mathtt{HE}}$. Hence, regarding threshold decryption, $\overline{\mathtt{ThHE}}$ has the identical complexity with respect to N as $\underline{\mathtt{ThHE}}$. If we set $\underline{\mathtt{ThHE}}$ to be threshold Exp-ElGamal, the above construction allows $\overline{\mathtt{ThHE}}$ to enjoy the low complexity of threshold decryption, while the restriction on message space is mitigated by $\mathtt{MeSA}$.

Idea 2: Lattice-based MeSA for Exp-ElGamal. Based on the above MeSA-assisted threshold HE construction, to resolve the efficiency-utility dilemma of ThLHE, we just need to construct a customized MeSA for Exp-ElGamal. Our first observation is that lattice-based encryption is a good fit for the $\texttt{MeSA}$ of Exp-ElGamal as it satisfies the requirements **R1**–**R3**.

- For **R1**: Lattice-based encryption enables the flexible choice of message space. An example is an interval $(-P/2, P/2]$ of integers, where P is neither necessarily prime nor coprime with other integers. In addition, P can be set very large to mitigate the small-message restriction of Exp-ElGamal.
- For **R2**: The encryption keys of lattice-based encryption typically have a small value, or more specifically, are vectors with entries being -1, 0, or 1. This aligns with the small message-space of Exp-ElGamal. Although the vanilla message-space of Exp-ElGamal consists of numbers rather than vectors, to extend Exp-ElGamal for encrypting a length-D vector, we can simply encrypt each entry of this vector separately. We apply such an extension throughout this paper so that Exp-ElGamal can encrypt vectors.
- For **R3**: Lattice-based secret-key encryption can be made *linearly* homomorphic in both of the messages and the keys. Particularly, the ciphertext of message m_1 (resp. m_2) can be $\mathsf{ct}_1 = \mathbf{a}_1 \cdot \mathsf{ek}_1 + \mathbf{e}_1 + \Delta \cdot m_1$ (resp. $\mathsf{ct}_2 = \mathbf{a}_2 \cdot \mathsf{ek}_2 + \mathbf{e}_2 + \Delta \cdot m_2$), where $\mathbf{a}_1, \mathbf{a}_2$ are public random values, $\mathbf{e_1}, \mathbf{e}_2$ are random errors, and Δ is a fixed parameter. By using the same public random value (i.e., $\mathbf{a}_1 = \mathbf{a}_2 = \mathbf{a}$), we have

  $$\alpha_1 \cdot \mathsf{ct}_1 + \alpha_2 \cdot \mathsf{ct}_2 = \mathbf{a} \cdot (\alpha_1 \cdot \mathsf{ek}_1 + \alpha_2 \cdot \mathsf{ek}_2) + (\alpha_1 \cdot \mathbf{e}_1 + \alpha_2 \cdot \mathbf{e}_2) + \Delta \cdot (\alpha_1 m_1 + \alpha_2 m_2).$$

  This indicates that $\alpha_1 \cdot \mathsf{ct}_1 + \alpha_2 \cdot \mathsf{ct}_2$ is a valid encryption of $\alpha_1 m_1 + \alpha_2 m_2$ under the key $\alpha_1 \cdot \mathsf{ek}_1 + \alpha_2 \cdot \mathsf{ek}_2$ and the error $\alpha_1 \cdot \mathbf{e}_1 + \alpha_2 \cdot \mathbf{e}_2$.

Since the requirements **R1**–**R3** are satisfied, we can use lattice-based encryption to develop a $\texttt{MeSA}$ scheme for Exp-ElGamal. By combining it with (threshold) Exp-ElGamal, we obtain a novel LHE scheme called Lattice-Exp-ElGamal. Notably, no matter how large the messages m_1, m_2 are, the (threshold) Lattice-Exp-ElGamal only needs to decrypt $\alpha_1 \cdot \mathsf{ek}_1 + \alpha_2 \cdot \mathsf{ek}_2$, which can be quite small.

However, this MeSA design still has a issue: After homomorphic evaluation, the key $\alpha_1 \cdot \mathsf{ek}_1 + \alpha_2 \cdot \mathsf{ek}_2$ and the error $\alpha_1 \cdot \mathbf{e}_1 + \alpha_2 \cdot \mathbf{e}_2$ both grow rapidly with a multiplicative factor $\mathcal{O}(\alpha)$ when α_1, α_2 becomes large. If these two values exceed some bound, this would result in the decryption failures of either Exp-ElGamal or lattice-based secret-key encryption. We address this issue by the *gadget decomposition* technique. Below we present the main idea by utilizing binary digit decomposition and readers can refer to [35] for more general methods. For simplicity, we only discuss the case with one ciphertext, i.e., to evaluate $\alpha \cdot m$ over the ciphertext of m, and the case with multiple ciphertexts follows accordingly. Suppose that the coefficient α is bounded by 2^L, then we can rewrite $\alpha = \sum_{i=0}^{L-1} \alpha^{(i)} \cdot 2^i$ with $\alpha^{(i)} \in \{0, 1\}$ and $\alpha \cdot m = \sum_{i=0}^{L-1} \alpha^{(i)} \cdot (2^i m)$. Inspired by this rewriting, we encrypt $m, 2m, \ldots, 2^{L-1}m$ instead of solely encrypting m. When evaluating $\alpha \cdot m$ over the ciphertexts of m (i.e., $\texttt{Enc}(2^i m)$ for

$i = 0, \ldots, L-1$), one can simply compute $\sum_{i=0}^{L-1} \alpha^{(i)} \cdot \texttt{Enc}(2^i m)$ by evaluating homomorphic addition at most L times (note that $\alpha^{(i)}$ is either 0 or 1). In this way, the encryption keys and errors only grow with a multiplicative factor $\mathcal{O}(L)$. Hence, this idea reduces the growth of encryption keys and errors from $\mathcal{O}(\alpha)$ to $\mathcal{O}(L) = \mathcal{O}(\log \alpha)$, at the cost of encrypting $\mathcal{O}(L) = \mathcal{O}(\log \alpha)$ ciphertexts.

By adopting this technique, the improved `MeSA` scheme now enables the (threshold) Lattice-Exp-ElGamal to evaluate linear functions with large messages and large coefficients over a large number of ciphertexts. As we will show in Remark 2, the (threshold) Lattice-Exp-ElGamal can already meet the utility demands in practice. Meanwhile, threshold Lattice-Exp-ElGamal has the same quasi-linear complexity for decryption algorithms. Hence, threshold Lattice-Exp-ElGamal helps to resolve the efficiency-utility dilemma of ThLHE.

Idea 3: ThLHE for Multi-Party Fully HE. Multi-party fully HE (MFHE) generalizes ThFHE so that the decryption is a multi-round protocol [45], while existing ThFHE schemes can be regarded as a special case of MFHE with one-round decryption. To date, the decryption protocols of almost all MFHE schemes [7,24,52,15] still suffer from high $\mathcal{O}(N^2 \log N)$ complexity. The only two exceptions with quasi-linear complexity are those aforementioned ThFHE schemes with either "full-threshold" restriction or "known-participants" assumption. Our third idea aims to show that ThLHE, and in particular, our new threshold Lattice-Exp-ElGamal, can be applied to construct MFHE schemes with quasi-linear complexity without those utility restrictions.

We first recap that Chang and Li recently establish a reduction from MFHE to a new primitive called *approximate secret sharing* (ApproxSS) [15]. Differing from vanilla secret sharing, ApproxSS recovers an approximate message m', such that 1) m' is "close" to the original message m (called approximate correctness) and 2) the adversary who corrupts $T-1$ parties cannot learn the exact value of m (called approximate security). They further show that ApproxSS is the bottleneck of MFHE's higher efficiency and suggest "encrypted share" is a better idea for efficient ApproxSS construction. Nonetheless, their adopted encryption scheme fails to break the barrier of quadratic complexity.

In this work, we demonstrate that ThLHE is another candidate for ApproxSS construction. Specifically, a ThLHE-based ApproxSS works as follows.

- *Share.* To share a message m, our ThLHE-based ApproxSS requires two steps: 1) invokes the key generation algorithm of ThLHE to generate a public key pk and N secret key shares $\{\mathsf{sk}_i\}_{i\in[N]}$; 2) invokes the encryption algorithm of ThLHE to encrypt m with pk. Then the i-th share of m consists of the ciphertext of m and a secret key share sk_i.
- *Approximate Recovery.* This protocol consists of two rounds. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ denote the set of at least T participants in the first round and the second round, respectively. In the first round, each party $i \in \mathcal{T}_1$ encrypts a noise n_i and sends the ciphertext to other parties, such that the receiver can obtain the ciphertext of $m + \sum_{i\in\mathcal{T}_1} n_i$ by homomorphic evaluation. In the second round, each party $i \in \mathcal{T}_2$ generates a decryption share via the partial

decryption algorithm. With the decryption shares from $\mathcal{T}_2$, any party can recover the approximate message $m' = m + \sum_{i \in \mathcal{T}_1} n_i$ via the final decryption algorithm. In this way, the approximate correctness is achieved by setting each n_i to be small, while the approximate security is guaranteed as any $T-1$ parties cannot learn the exact value of $\sum_{i \in \mathcal{T}_1} n_i$.

By instantiating the ThLHE in above ApproxSS construction to our threshold Lattice-Exp-ElGamal, we obtain the first concretely-efficient MFHE scheme with quasi-linear decryption complexity but without the "full-threshold" restriction and "known-participants" assumption. We note that other ThLHE schemes cannot achieve this improvement. Specifically, most of ThLHE schemes do not have the quasi-linear complexity. As for the Exp-ElGamal, the "small-message" restriction disables its usage in this application, as the messages of ApproxSS correspond to the ciphertexts of MFHE that must belong to a large space.

1.3 Related Work

Below we review existing ThLHE schemes by categorizing them into three types based on their underlying security assumptions.

- *Factoring-Based Schemes.* The factoring-based LHE schemes include the initial scheme by Goldwasser and Micali in [39], the improved schemes by Naccache and Stern in [46], and by Okamoto and Uchiyama in [48]. A remarkable scheme is the well-known Paillier [49] and its generalization by Damgård and Jurik [25]. These schemes base their security on the problem of factoring RSA integers, namely, guessing two large primes p and q from $N = p \cdot q$. Under this problem, the p and q, which reveal the order of underlying group, should remain unknown. Indeed, when thresholdizing these schemes, the secret sharing (SS) should be performed *without knowing the group order*. To tackle this challenge, potential solutions include linear *integer* SS [26,5] and *black-box* SS [28,19,20,22] that usually have higher complexity than vanilla Shamir SS. A widely-used idea is the "clearing-out denominator" technique, which was initially proposed by Shoup in [50] and used in threshold Goldwasser-Micali [40] by Katz and Yung, as well as threshold Paillier by Damgård et al. [27], by Fouque et al. [32] and by Friedman et al. [34]. This idea needs to raise extremely-large values to the power of exponents, which is quite inefficient and results in the complexity being $\mathcal{O}(N^2 \log N)$ [2].
- *DDH-Based Schemes.* The DDH-based LHE schemes include the classical exponential ElGamal (Exp-ElGamal) proposed by Cramer et al. [21] and the recent CL framework proposed by Castagnos and Laguillaumie [12]. The Exp-ElGamal does not have unknown-order group problem and thus its threshold decryption [21] can directly apply the Shamir secret sharing to recover the secret "in the exponent". By optimizing the efficiency of Shamir secret shairing with number theoretic transform [53,23], the threshold decryption of Exp-ElGamal can achieve quasi-linear complexity. Nonetheless, the decryption of Exp-ElGamal requires to solve a discrete logarithm, which

is hard in general. To make decryption practical, Exp-ElGamal usually assumes that the messages come from a small, known set. In comparison, the CL framework [12] does not have such a restriction and recent works have proposed two threshold variants [11,13]. In [11], Braun, Damgård, and Orlandi propose the first actively-secure CL-based ThLHE and enables the key to be generated in a distributed manner, which addresses the open problem in [37]. In [13], Castagnos, Laguillaumie, and Tucker provide the first ThLHE whose message space is $\mathbb{Z}/2^k\mathbb{Z}$ for any integer k. Despite of the effectiveness of their breakthroughs, these two schemes face the unknown-order group issue like factoring-based schemes and adopt the same "clearing-out denominator" technique. Consequently, the complexity is also $\mathcal{O}(N^2 \log N)$.

– *Lattice-Based Schemes.* In the past decade, researchers devise multiple lattice-based FHE schemes that can evaluate arbitrary functions, including Gentry's pioneering scheme [36] and the subsequent improvements [10,31,16,17,29]. For these schemes, the encryption adds a small random error to the message for security. Asharov et al. proposes the first threshold FHE scheme, which adopts additive secret sharing and works well with the errors [3]. Nonetheless, this scheme only supports full-threshold and requires all parties to be available for decryption. Hence, it is not resilient to non-participants that can be caused by dropout devices or malicious behaviors. Later, Boneh et al. proposed two schemes with arbitrary-threshold [7]. One adopts $\{0, 1\}$-linear integer secret sharing, which naturally supports the errors in ciphertexts. However, the complexity of decryption is $\mathcal{O}(N^{5.2})$ on average, which is relatively high. The other adopts Shamir secret sharing. In this scheme, the errors will be blown up to an arbitrarily-large unknown number by the Lagrange coefficients, resulting in decryption failures. To address this issue, they adopt the "clearing-out denominator" technique so that the error can be bounded. Nonetheless, the complexity becomes $\mathcal{O}(N^2 \log N)$ again. Recently, Mouchet et al. propose threshold FHE schemes with quasi-linear complexity and arbitrary-threshold [42,43]. However, they assume that the set of participants is available as a prior knowledge, just like the setting in [33]. This solution is still not resilient to malicious non-participants or simple random failures, which limits its potential applications. We also note that a recent work [2] by Applebaum, Nir, and Pinkas presents the first secret sharing technique with only $\mathcal{O}(N)$ additions for recovery, which is potentially useful for efficient ThFHE construction. Still, this technique is a *ramp* secret sharing that has two different thresholds for privacy and correctness, respectively. This does not align with the setting of ThFHE.

In summary, ThLHE currently faces either high complexity or utility restrictions. Resolving this efficiency-utility dilemma is the main goal of this work.

2 Preliminaries

Basic Notation. We use bold lower-case letters (e.g., $\mathbf{a}$) to denote vectors and bold upper-case letters (e.g., $\mathbf{A}$) to denote matrices. For a vector $\mathbf{a}$, its i-th

element is denoted by $\mathbf{a}_i$ and its (infinity) norm is defined as $\|\mathbf{a}\| = \max_i |\mathbf{a}_i|$. For a matrix $\mathbf{A}$, its i-th row is a vector denoted by $\mathbf{A}_i$, while $\mathbf{A}_{i,j}$ is the j-th element of $\mathbf{A}_i$. For a number $x \in \mathbb{R}$, we use $\lfloor x \rceil$ to denote rounding to the nearest integer and $\lfloor x \rfloor$, $\lceil x \rceil$ to denote rounding up or down. Let a and P be two integers, then by $[a]_P$ we denote the *unique* integer in the interval $(-P/2, P/2]$ such that $[a]_P \equiv a \mod P$. Most of sets are denoted by calligraphic letters (e.g., $\mathcal{T}$), except that $[N]$ is the set $\{1, \cdots, N\}$ for some positive integer N.

For a security parameter λ, a function $\epsilon(\cdot)$ is negligible in λ if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$. We use $\mathsf{negl}(\lambda)$ to denote such a negligible function. In addition, $a \leftarrow\!\!\$\ \chi$ denotes that a is sampled at random according to distribution χ. We also use $\mathsf{Uniform}(\mathcal{M})$ to denote the uniform distribution over a set $\mathcal{M}$. For sampling a from $\mathsf{Uniform}(\mathcal{M})$, we simply write $a \leftarrow\!\!\$\ \mathcal{M}$. A distribution χ over integers is called B-bounded if it is supported on $[-B, B]$. Throughout this paper, we use K to denote the number of inputs for a function f in some family $\mathcal{F}$. Since $\mathcal{F}$ may contain many functions f whose number of inputs are different, we note that K is just a notation representing a variable rather than a constant value.

2.1 Homomorphic Encryption

Definition 1 (HE). *An $\mathcal{F}$-homomorphic encryption scheme over message space $\mathcal{M}$ is a tuple $\mathtt{HE} = (\mathtt{HE.Setup}, \mathtt{HE.KGen}, \mathtt{HE.Enc}, \mathtt{HE.Eval}, \mathtt{HE.Dec})$ of PPT algorithms with the following properties:*
***Setup**: $\mathsf{pp} \leftarrow \mathtt{HE.Setup}(1^\lambda)$. On input the security parameter λ, this algorithm outputs a public parameter pp that is the implicit argument of other algorithms.*
***Key Generation**: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{HE.KGen}()$. This algorithm outputs a public key pk and a secret key sk.*
***Encryption**: $\mathsf{ct} \leftarrow \mathtt{HE.Enc}(m; \mathsf{pk})$. On input a message $m \in \mathcal{M}$ and the pk, this algorithm outputs the ciphertext ct of m.*
***Evaluation**: $\mathsf{ct}' \leftarrow \mathtt{HE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$. On input a K-ary function $f \in \mathcal{F}$ and K ciphertexts, this algorithm outputs a new ciphertext ct'.*
***Decryption**: $m' \leftarrow \mathtt{HE.Dec}(\mathsf{ct}', \mathsf{sk})$. On input a ciphertext ct' and the sk, this algorithm outputs a message m'.*

This work involves two types of HE schemes. One is linearly HE (LHE) in which $\mathcal{F}$ contains addition (i.e., $m_1 + m_2$) and scalar multiplication (i.e., $\alpha \cdot m$ for some coefficient α). The other is fully HE (FHE) that supports the evaluation of arbitrary functions. For all types of HE schemes, they are generally required to satisfy compactness, correctness, and security.

Definition 2 (Compactness of HE). *An $\mathtt{HE}$ scheme is compact if there is a polynomial p, such that for any security parameter λ the following statement holds: For any function $f \in \mathcal{F}$ and any K ciphertexts $\mathsf{ct}_k \leftarrow \mathtt{HE.Enc}(m_k; \mathsf{pk})$ with $k \in [K]$, the size of the evaluation output $\mathsf{ct}' \leftarrow \mathtt{HE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ is no more than $p(\lambda)$ bits, independent of the size of f and the value of K.*

Definition 3 (Correctness of HE). *An $\mathtt{HE}$ scheme is correct if for all λ, the following two statements hold.*

– *For any* $m \in \mathcal{M}$*, if* $\mathsf{ct} \leftarrow \mathtt{HE.Enc}(m; \mathsf{pk})$*, then*

$$Pr[\mathtt{HE.Dec}(\mathsf{ct}; \mathsf{sk}) = m] = 1 - \mathsf{negl}(\lambda).$$

– *For any* K*-ary function* $f \in \mathcal{F}$ *and* K *messages* $m_1, \ldots, m_K \in \mathcal{M}$*, if* $\mathsf{ct}_k \leftarrow \mathtt{HE.Enc}(m_k; \mathsf{pk})$ *for* $k \in [K]$ *and* $\mathsf{ct}' \leftarrow \mathtt{HE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$*, then*

$$Pr[\mathtt{HE.Dec}(\mathsf{ct}'; \mathsf{sk}) = f(m_1, \ldots, m_K)] = 1 - \mathsf{negl}(\lambda).$$

Definition 4 (Security of HE). *An* HE *scheme satisfies security if for all* λ *and any PPT adversary* $\mathcal{A}$*, the following experiment* $\mathsf{Expt}_{\mathcal{A},\mathtt{HE}}(1^\lambda)$ *outputs* 1 *with probability* $\frac{1}{2} + \mathsf{negl}(\lambda)$*:*
$\mathsf{Expt}_{\mathcal{A},\mathtt{HE}}(1^\lambda)$:

1. *On input security parameter* λ*, the challenger runs* $\mathsf{pp} \leftarrow \mathtt{HE.Setup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{HE.KGen}()$*. It then sends the* pp, pk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *selects two messages* $m_0, m_1 \in \mathcal{M}$ *and sends them to the challenger.*
3. *The challenger runs* $\mathsf{ct} \leftarrow \mathtt{HE.Enc}(m_b; \mathsf{pk})$ *for* $b \leftarrow\!\!\$\ \{0, 1\}$ *and sends* ct *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs a guess* b'*. The experiment outputs* 1 *if* $b = b'$*.*

2.2 Threshold Homomorphic Encryption

Definition 5 (Threshold HE). *Given a message space* $\mathcal{M}$*, a* T*-out-of-*N *threshold* $\mathcal{F}$*-homomorphic encryption scheme is a tuple of PPT algorithms* $\mathtt{ThHE} = (\mathtt{ThHE.Setup}, \mathtt{ThHE.KGen}, \mathtt{ThHE.Enc}, \mathtt{ThHE.Eval}, \mathtt{ThHE.PartDec}, \mathtt{ThHE.FinDec})$ *with the following properties:*
Setup*:* $\mathsf{pp} \leftarrow \mathtt{ThHE.Setup}(1^\lambda)$*. On input security parameter* λ*, this algorithm outputs public parameters* pp*, which is the implicit argument of other algorithms.*
Key Generation*:* $\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \mathtt{ThHE.KGen}()$*. This algorithm outputs a public key* pk *and* N *secret key shares* $\{\mathsf{sk}_i\}_{i\in[N]}$*.*
Encryption*:* $\mathsf{ct} \leftarrow \mathtt{ThHE.Enc}(m; \mathsf{pk})$*. On input a message* $m \in \mathcal{M}$ *and the* pk*, this algorithm outputs a ciphertext* ct*.*
Evaluation*:* $\mathsf{ct}' \leftarrow \mathtt{ThHE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$*. On input a* K*-ary function* $f \in \mathcal{F}$ *and* K *ciphertexts, this algorithm outputs a new ciphertext* ct'*.*
Partial Decryption*:* $\mathsf{ds}_i \leftarrow \mathtt{ThHE.PartDec}(\mathsf{ct}'; \mathsf{sk}_i)$*. On input a ciphertext* ct' *and a secret key share* sk_i*, this algorithm outputs a decryption share* ds_i*.*
Final Decryption*:* $m' \leftarrow \mathtt{ThHE.FinDec}(\mathsf{ct}'; \{\mathsf{ds}_i\}_{i\in\mathcal{T}})$*. On input a ciphertext* ct' *and any* T *decryption shares* $\{\mathsf{ds}_i\}_{i\in\mathcal{T}}$ *from set* $\mathcal{T}$*, this algorithm outputs a message* $m' \in \mathcal{M}$*.*

Similarly to linearly HE and fully HE, threshold linearly HE (ThLHE) is a special ThHE in which $\mathcal{F}$ contains the addition and scalar multiplication, while threshold fully HE (ThFHE) is a special ThHE in which $\mathcal{F}$ contains arbitrary functions. Below we review the requirements on ThHE, including compactness, correctness, and security.

Definition 6 (Compactness of ThHE). *A T-out-of-N $\mathtt{ThHE}$ scheme is compact if there are two polynomials p_1, p_2, such that for any security parameter λ the following statement holds: For any function $f \in \mathcal{F}$ and any K ciphertexts $\mathsf{ct}_k \leftarrow \mathtt{ThHE.Enc}(m_k; \mathsf{pk})$ with $k \in [K]$, the size of the evaluation output $\mathsf{ct}' \leftarrow \mathtt{ThHE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ is no more than $p_1(\lambda)$ bits and the size of the decryption share $\mathsf{ds}_i \leftarrow \mathtt{ThHE.PartDec}(\mathsf{ct}'; \mathsf{sk}_i)$ is no more than $p_2(\lambda, N, T)$, independent of the size of f and the value of K.*

Definition 7 (Correctness of ThHE). *A T-out-of-N $\mathtt{ThHE}$ scheme is correct if for all λ, the following two statements hold.*

- *For any $m \in \mathcal{M}$ and size-T set $\mathcal{T} \subseteq [N]$, if $\mathsf{ct} \leftarrow \mathtt{ThHE.Enc}(m; \mathsf{pk})$, then*

$$Pr[\mathtt{ThHE.FinDec}(\mathsf{ct}; \{\mathtt{ThHE.PartDec}(\mathsf{ct}; \mathsf{sk}_i)\}_{i \in \mathcal{T}}) = m] = 1 - \mathsf{negl}(\lambda).$$

- *For any K-ary function $f \in \mathcal{F}$ and K messages $m_1, \ldots, m_K \in \mathcal{M}$, if $\mathsf{ct}_k \leftarrow \mathtt{ThHE.Enc}(m_k; \mathsf{pk})$ for $k \in [K]$ and $\mathsf{ct}' \leftarrow \mathtt{ThHE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$, then*

$$Pr[\mathtt{ThHE.FinDec}(\mathsf{ct}'; \{\mathtt{ThHE.PartDec}(\mathsf{ct}'; \mathsf{sk}_i)\}_{i \in \mathcal{T}}) \neq f(m_1, \ldots, m_K)] = \mathsf{negl}(\lambda).$$

Definition 8 (Security of ThHE). *A T-out-of-N $\mathtt{ThHE}$ scheme satisfies simulation based security if for all λ and K, the following holds. There exist stateful PPT simulator algorithms $\mathsf{S}_1, \mathsf{S}_2$ such that for any PPT adversary $\mathcal{A}$, the following experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^K)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^K)$ are indistinguishable: (we use red parts to highlight the differences between these two experiments)*

$\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^K)$:

1. On input the security parameter λ, the challenger runs $\mathsf{pp} \leftarrow \mathtt{ThHE.Setup}(1^\lambda)$ and provides pp to $\mathcal{A}$.
2. $\mathcal{A}$ outputs K messages $m_1, \ldots, m_K \in \overline{\mathcal{M}}$ and a size-$(T-1)$ set $\mathcal{C} \subset [N]$.
3. The challenger runs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in [N]}) \leftarrow \mathtt{ThHE.KGen}()$ and $\mathsf{ct}_k \leftarrow \mathtt{ThHE.Enc}(m_k; \mathsf{pk})$ for $k \in [K]$, then provides pk, $\{\mathsf{sk}_i\}_{i \in \mathcal{C}}$, and $\{\mathsf{ct}_k\}_{k \in [K]}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a set $\mathcal{T}$ of at least T parties and a function $f \in \mathcal{F}$ with K inputs if such f exists; otherwise, aborts this experiment and outputs a random bit b with $P(b = 0) = P(b = 1) = 0.5$.
5. The challenger runs $\mathsf{ct}' \leftarrow \mathtt{ThHE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ and $\mathsf{ds}_i \leftarrow \mathtt{ThHE.PartDec}(\mathsf{ct}'; \mathsf{sk}_i)$ for $i \in \mathcal{T}$, then sends $\mathsf{ct}', \{\mathsf{ds}_i\}_{i \in \mathcal{T}}$ to $\mathcal{A}$.
6. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

$\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^K)$:

1. On input the security parameter λ, the challenger runs $\mathsf{pp} \leftarrow \mathtt{ThHE.Setup}(1^\lambda)$ and provides pp to $\mathcal{A}$.
2. $\mathcal{A}$ outputs K messages $m_1, \ldots, m_K \in \overline{\mathcal{M}}$ and a size-$(T-1)$ set $\mathcal{C} \subset [N]$.
3. The challenger runs $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i \in [N]}, \{\mathsf{ct}_k\}_{k \in [K]}) \leftarrow \mathsf{S}_1(1^\lambda, 1^K)$, then provides pk, $\{\mathsf{sk}_i\}_{i \in \mathcal{C}}$, and $\{\mathsf{ct}_k\}_{k \in [K]}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a set $\mathcal{T}$ of at least T parties and a function $f \in \mathcal{F}$ with K inputs if such f exists; otherwise, aborts this experiment and outputs a random bit b with $P(b = 0) = P(b = 1) = 0.5$.
5. The challenger runs $\mathsf{ct}' \leftarrow \mathtt{ThHE.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ and $\{\mathsf{ds}_i\}_{i \in \mathcal{T}} \leftarrow \mathsf{S}_2(\{\mathsf{sk}_i\}_{i \in \mathcal{C}}, \{\mathsf{ct}_k\}_{k \in [K]}, \mathsf{ct}', f(m_1, \ldots, m_K), \mathsf{st})$, then sends $\mathsf{ct}', \{\mathsf{ds}_i\}_{i \in \mathcal{T}}$ to $\mathcal{A}$.
6. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

2.3 Exp-ElGamal and Its Threshold Variants

We review (threshold) Exp-ElGamal scheme as follows. For brevity, we do not explicitly state the theorems of their compactness, correctness, and security.

Exp-ElGamal Encryption. In the common setting of Exp-ElGamal ($\mathtt{EG}$), the public parameters $\mathsf{pp} \leftarrow \mathtt{EG.Setup}(1^\lambda)$ include two large primes p, q and one generator g of $\mathbb{G}_q$, such that $p = 2q + 1$ and the DDH problem is hard in $\mathbb{G}_q$. The algorithm $\mathtt{EG.KGen}$ samples $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_q$ and computes $\mathsf{pk} = g^{\mathsf{sk}}$. To encrypt $m \in \mathbb{Z}_q$, the algorithm $\mathtt{EG.Enc}$ chooses $y \leftarrow\!\!\$\ \mathbb{Z}_q$ and outputs $\mathsf{ct} = (c, d)$ with $c = g^y$ and $d = \mathsf{pk}^y \cdot g^m$.

The algorithm $\mathtt{EG.Eval}$ works as follows. Given two ciphertexts $\mathsf{ct}_1 = (c_1, d_1)$ and $\mathsf{ct}_2 = (c_2, d_2)$ that are the encryptions of m_1 and m_2 using random values y_1 and y_2, respectively, then it computes $(c_1c_2, d_1d_2) = (g^{y_1+y_2}, \mathsf{pk}^{y_1+y_2} \cdot g^{m_1+m_2})$, which is a valid encryption of $m_1 + m_2$. Furthermore, given a coefficient α, it computes $(c_1^\alpha, d_1^\alpha) = (g^{\alpha y_1}, \mathsf{pk}^{\alpha y_1} \cdot g^{\alpha m_1})$, which is a valid encryption of αm_1.

The algorithm $\mathtt{EG.Dec}$ decrypts the ciphertext $\mathsf{ct} = (c, d)$ by computing $g^m = d/c^{\mathsf{sk}}$ and recovering m from g^m. Note that recovering m from g^m requires to solve a discrete logarithm in $\mathbb{G}_q$, which is hard in general. Hence, we usually assume that the m is in an interval $[a, b]$ with $b - a$ being sufficiently small, so that the state-of-the-art solver for discrete logarithm (e.g., Pollard's Lambda algorithm) can recover m with a computation complexity of $\mathcal{O}(\sqrt{b-a})$. Mitigating this small-message restriction is a major goal of this work.

Threshold Exp-ElGamal Encryption. A T-out-of-N threshold Exp-ElGamal ($\mathtt{ThEG}$) scheme is constructed by modifying the key generation and decryption algorithms of Exp-ElGamal. The algorithm $\mathtt{ThEG.KGen}$ executes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{HE.KGen}()$ and then generates N shares $\{\mathsf{sk}_i\}_{i\in[N]}$ of sk using T-out-of-N Shamir secret sharing. In this way, sk can be recovered from any set $\mathcal{T}$ of at least T shares as $\mathsf{sk} = \sum_{i\in\mathcal{T}} L_i^{(\mathcal{T})} \cdot \mathsf{sk}_i$. Here, $L_i^{(\mathcal{T})} = \prod_{j\in\mathcal{T}, j\neq i}[x_j/(x_j - x_i)]$ is usually called the Lagrange coefficients, where x_i is a number corresponding to share i. By this property, the decryption algorithms work as follows. Given a ciphertext $\mathsf{ct} = (c, d)$ and a secret-key share sk_i, the algorithm $\mathtt{ThEG.PartDec}$ computes a decryption share $\mathsf{ds}_i = c^{\mathsf{sk}_i}$. After collecting enough decryption shares from set $\mathcal{T}$, the algorithm $\mathtt{ThEG.FinDec}$ reveals m from $g^m = d/\prod_{i\in\mathcal{T}} \mathsf{ds}_i^{L_i^{(\mathcal{T})}}$.

Regarding the complexity of decryption algorithms, we note that sk_i and $L_i^{(\mathcal{T})}$ can be treated as random elements of $\mathbb{G}_q$ for sufficiently large N. Hence, their values are bounded by $q - 1$ and independent of N. Given the knowledge of Lagrange coefficients, the decryption algorithms consist of several modular exponentiation operations with $\mathcal{O}(1)$ complexity and the multiplication of T numbers with $\mathcal{O}(N)$ complexity. In fact, the most expensive operation is to compute the Lagrange coefficients, which generally has a complexity of $\mathcal{O}(N^2)$. With the help of number theoretic transform, the complexity of this operation, as well as the whole decryption process, can be reduced to $\mathcal{O}(N \log N)$.

Extension for Encrypting Matrices. We note that Exp-ElGamal can be naturally extended to support the cases where a message is a matrix rather than a single number. Specifically, we can simply run algorithms of encryption,

evaluation, and decryption over every entries of matrices, respectively. In this way, these algorithms are executed $D * L$ times if each matrix has D rows and L columns. Throughout this paper, we apply such an extension to Exp-ElGamal so that its message-space consists of matrices.

2.4 Lattice-Based Secret-Key Encryption

Lattice-based cryptography typically involves the hardness of learning-with-error (LWE) problem and its variants (e.g., RLWE and LPN problems). The LWE problem is defined as follows. Let $\mathbb{Z}_Q$ denote the ring of integers modulo Q and $\mathbb{Z}_Q^D$ denote the set of size-D vectors over $\mathbb{Z}_Q$. For a secret vector $\mathbf{s} \leftarrow\!\$\ \chi_s$, the decision-LWE problem $\mathsf{LWE}_{Q,D,\chi_s,\chi_e}$ asks to distinguish between $(\mathbf{a} \leftarrow\!\$\ \mathbb{Z}_Q^D, \mathbf{a}^T\mathbf{s}+e)$ and $(\mathbf{a} \leftarrow\!\$\ \mathbb{Z}_Q^D, u \leftarrow\!\$\ \mathbb{Z}_Q)$, where $e \leftarrow\!\$\ \chi_e$ is the so-called error term. Generally, this problem is assumed to be hard even when χ_e is a B-bounded discretized Gaussian distribution and χ_s samples $\mathbf{s}$ with $Pr[\mathbf{s}_i = 0] = 0.5$ and $Pr[\mathbf{s}_i = 1] = Pr[\mathbf{s}_i = -1] = 0.25$.

Based on the hardness of $\mathsf{LWE}_{Q,D,\chi_s,\chi_e}$ problem, we can obtain a secret-key encryption scheme over message space $\mathbb{Z}_P$, where P is neither necessarily a prime nor co-prime with other integers. In this encryption scheme, the encryption of message m with encryption key $\mathbf{s} \leftarrow\!\$\ \chi_s$ outputs $\mathsf{ct} = (\mathbf{a}, b = \mathbf{a}^T\mathbf{s} + e + \Delta \cdot [m]_P)$, where $\mathbf{a} \leftarrow\!\$\ \mathbb{Z}_Q^D$, $e \leftarrow\!\$\ \chi_e$, and $\Delta = \lfloor \frac{Q}{P} \rfloor$. The decryption algorithm computes $m = \lfloor \frac{P \cdot [b - \mathbf{a}^T\mathbf{s}]_Q}{Q} \rceil \mod P$. Existing works have shown that the decryption will successfully output m if and only if the error $|e| < \Delta/2$. Following existing works [4,15], we use this encryption scheme for its key homomorphism and message homomorphism properties. Specifically, given $\mathsf{ct}_1 = (\mathbf{a}, b_1 = \mathbf{a}^T\mathbf{s}_1 + e_1 + \Delta \cdot m_1)$ and $\mathsf{ct}_2 = (\mathbf{a}, b_2 = \mathbf{a}^T\mathbf{s}_2 + e_2 + \Delta \cdot m_2)$, one can compute $\mathsf{ct} = (\mathbf{a}, b_1 + b_2 = \mathbf{a}^T(\mathbf{s}_1 + \mathbf{s}_2) + (e_1 + e_2) + \Delta \cdot (m_1 + m_2))$, which is a valid encryption of $m_1 + m_2$ under key $(\mathbf{s}_1 + \mathbf{s}_2)$ with error $e_1 + e_2$.

Throughout this paper, we use this simple encryption scheme as a concrete instance to present our work (in particular, the MeSA design). For brevity, we do not explicitly demonstrate its correctness and security here; instead, we will prove that our MeSA from this lattice-based encryption is correct and secure. Meanwhile, our work can be naturally extended by using other lattice-based schemes, such as BGV [10], BFV [31], CKKS [16], Torus-FHE [17], FHEW [29], etc. For example, the RLWE-based schemes can help to reduce the running time with its single-instruction-multiple-data (SIMD) feature.

3 Message-Space Adapter for Homomorphic Encryption

In this section, we present the definitions of message-space adapter (MeSA) for HE in Section 3.1. Then, in Sections 3.2 and 3.3, we use MeSA to construct HE and its threshold version with target message space, respectively.

3.1 Formulation of Message-Space Adapter

On a high level, message-space adapter is a special encryption scheme that can alter the message space of an $\mathcal{F}$-homomorphic encryption scheme from $\underline{\mathcal{M}}$ to $\overline{\mathcal{M}}$. We point out that an encryption scheme can achieve this functionality if it meets three conditions: 1) It is secret-key encryption over message space $\overline{\mathcal{M}}$; 2) Its encryption key space is $\underline{\mathcal{M}}$, from which the encryption algorithm can sample keys for each message individually; and 3) It is homomorphic in both of messages and encryption keys with respect to the function family $\mathcal{F}$. To make these conditions concrete, we formulate the message-space adapter as follows.

Definition 9 (Message-Space Adapter). *For a family $\mathcal{F}$ of functions, a **message-space adapter** from message-space $\underline{\mathcal{M}}$ to message-space $\overline{\mathcal{M}}$ is a tuple $\mathtt{MeSA} = (\mathtt{MeSA.Setup}, \mathtt{MeSA.Enc}, \mathtt{MeSA.Eval}, \mathtt{MeSA.KEval}, \mathtt{MeSA.Dec})$ of PPT algorithms specified as follows:*
***Setup**: $\mathsf{pp} \leftarrow \mathtt{MeSA.Setup}(1^\lambda)$. On input security parameter λ, this algorithm outputs public parameters pp, the implicit argument of other algorithms.*
***Encryption**: $(\mathsf{ek}, \mathsf{ct}) \leftarrow \mathtt{MeSA.Enc}(m)$. On input a message $m \in \overline{\mathcal{M}}$, this algorithm outputs a pair of encryption key $\mathsf{ek} \in \underline{\mathcal{M}}$ and ciphertext ct.*
***Evaluation**: $\mathsf{ct}' \leftarrow \mathtt{MeSA.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$. On input a K-ary function $f \in \mathcal{F}$ and K ciphertexts, this algorithm outputs a new ciphertext ct'.*
***Key Evaluation**: $\mathsf{ek}' \leftarrow \mathtt{MeSA.KEval}(f; \mathsf{ek}_1, \ldots, \mathsf{ek}_K)$. On input a K-ary function $f \in \mathcal{F}$ and K encryption keys, this algorithm outputs a new $\mathsf{ek}' \in \underline{\mathcal{M}}$.*
***Decryption**: $m' \leftarrow \mathtt{MeSA.Dec}(\mathsf{ct}'; \mathsf{ek}')$. On input a pair of ciphertext ct' and encryption key ek', this algorithm outputs a message $m' \in \overline{\mathcal{M}}$.*

Remark 1. We remark that this definition differs from that of homomorphic encryption (HE) or functional encryption (FE), making MeSA a novel primitive. Regarding HE and its multi-key variant, MeSA has a KEval algorithm that requires key-homomorphic property, which HE does not have. Without this KEval algorithm, HE and its multi-key variant may have to reveal $\mathsf{ek}_1, \ldots, \mathsf{ek}_K$ when decrypting m', which leaks $m_1, \ldots, m_K$ and results in security issues. As for FE [8] and its multi-input variant [38], they adopt a single key that encrypts all messages, while MeSA uses different keys to encrypt each message individually. The most relevant primitive close to MeSA should be multi-client FE [38] that involves multiple encryption keys. Nonetheless, it is different from MeSA on three aspects. *First*, the encryption of multi-client FE is associated with some tag to address the "mix-and-match" issue of multi-input FE, while MeSA does not have this feature. *Second*, it enables a party who owns a functional key associated with f to learn $f(m_1, \ldots, m_K)$ for *any* messages $m_1, \ldots, m_K$ with the same tag. In contrast, MeSA enables the owner of ek' to decrypt only one ciphertext ct' and learn the $f(m_1, \ldots, m_K)$ for certain $m_1, \ldots, m_K$. *Third*, it samples encryption keys once for all, while MeSA allows each encryption key to be solely sampled only when that key is needed. Although this paper does not explicitly discuss this, MeSA can achieve its functionality due to these differences, while existing primitives like HE and FE cannot.

We require that a $\mathtt{MeSA}$ scheme satisfies coherence, compactness, correctness, and security. Intuitively, the coherence of $\mathtt{MeSA}$ states that the circuit of $\mathtt{MeSA.KEval}$ algorithm is a function in $\mathcal{F}$. Regarding the compactness, we require that the size of ciphertexts is independent of the size of function f and the value of K. As for the correctness, the ciphertext of m should be decrypted to m, no matter whether it is generated by $\mathtt{MeSA.Enc}$ or $\mathtt{MeSA.Eval}$.

Definition 10 (Coherence of MeSA). *A* $\mathtt{MeSA}$ *scheme is $\mathcal{F}$-coherent if 1) it is designed for a function family $\mathcal{F}$ and 2) for any $f \in \mathcal{F}$ and any K encryption keys $\{\mathsf{ek}_k\}_{k\in[K]}$, $f^* \in \mathcal{F}$ with f^* being the circuit of* $\mathtt{MeSA.KEval}(f; \mathsf{ek}_1, \ldots, \mathsf{ek}_K)$.

Definition 11 (Compactness of MeSA). *A* $\mathtt{MeSA}$ *scheme is compact if there exists a polynomial p, such that for any security parameter λ the following statement holds: For any function $f \in \mathcal{F}$ and any K ciphertexts $\{\mathsf{ct}_k\}_{k\in[K]}$ generated by* $\mathtt{MeSA.Enc}$*, the size of the evaluation output* $\mathsf{ct}' \leftarrow \mathtt{MeSA.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ *is no more than $p(\lambda)$, independent of the size of f and the value of K.*

Definition 12 (Correctness of MeSA). *A* $\mathtt{MeSA}$ *scheme is correct if for any* $\mathsf{pp} \leftarrow \mathtt{MeSA.Setup}(1^\lambda)$*, the following two statements hold.*

- *For any $m \in \overline{\mathcal{M}}$, if* $(\mathsf{ek}, \mathsf{ct}) \leftarrow \mathtt{MeSA.Enc}(m)$*, then*

$$Pr[\mathtt{MeSA.Dec}(\mathsf{ek}, \mathsf{ct}) = m] = 1 - \mathsf{negl}(\lambda).$$

- *Suppose that $f \in \mathcal{F}$ has K inputs. For any K messages $m_1, \ldots, m_K$, if* $(\mathsf{ek}_k, \mathsf{ct}_k) \leftarrow \mathtt{MeSA.Enc}(m_k)$ *for $k \in [K]$,* $\mathsf{ct}' \leftarrow \mathtt{MeSA.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$*, and* $\mathsf{ek}' \leftarrow \mathtt{MeSA.KEval}(f; \mathsf{ek}_1, \ldots, \mathsf{ek}_K)$*, then*

$$Pr[\mathtt{MeSA.Dec}(\mathsf{ek}', \mathsf{ct}') = f(m_1, \ldots, m_K)] = 1 - \mathsf{negl}(\lambda).$$

We now define the security for MeSA. Intuitively, the security should describe that no private information is leaked by public informations that can be obtained by the adversary $\mathcal{A}$. We provide two definitions of security, each of which has a special consideration on the "public information". One is semantic security and the other is simulation-based security. As formulated below, the semantic security assumes that the adversary can only see the ciphertexts, while it cannot obtain the encryption keys.

Definition 13 (Semantic Security of MeSA). *An* $\mathtt{MeSA}$ *scheme satisfies semantic security if for all λ and any PPT adversary $\mathcal{A}$, the following experiment* $\mathsf{Expt}_{\mathcal{A},\mathtt{MeSA}}(1^\lambda)$ *outputs 1 with probability* $\frac{1}{2} + \mathsf{negl}(\lambda)$*:*
$\mathsf{Expt}_{\mathcal{A},\mathtt{MeSA}}(1^\lambda)$:

1. *On input security parameter λ, the challenger runs* $\mathsf{pp} \leftarrow \mathtt{MeSA.Setup}(1^\lambda)$ *and sends the* pp *to $\mathcal{A}$.*
2. *$\mathcal{A}$ selects two messages $m_0, m_1 \in \mathcal{M}$ and sends them to the challenger.*
3. *The challenger executes* $(\mathsf{ek}, \mathsf{ct}) \leftarrow \mathtt{MeSA.Enc}(m_b)$ *for* $b \leftarrow\!\!\$\ \{0, 1\}$*. It then provides* ct *to $\mathcal{A}$.*

4. $\mathcal{A}$ *outputs a guess* b'. *The experiment outputs* 1 *if* $b = b'$.

For the notion of simulation-based security, the adversary may be allowed to decrypt a particular ciphertext ct' and thus can learn the corresponding encryption key ek', as well as the final message m'. We also assume that the adversary can corrupt a set $\mathcal{C}$ of encryption keys. Based on the above considerations, in the definition of simulation-based security, we restrict that no information about the uncorrupted keys $\{\mathsf{ek}_k\}_{k\notin\mathcal{C}}$ or their corresponding messages $\{m_k\}_{k\notin\mathcal{C}}$ should be leaked by the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_K, \mathsf{ct}'$ and the key ek', other than what is already implied by the decryption result m' and the corrupted keys $\{\mathsf{ek}_k\}_{k\in\mathcal{C}}$.

Definition 14 (Simulation-Based Security of MeSA). *A* MeSA *scheme satisfies simulation-based security if for all* λ *and* K, *the following holds. There exist stateful PPT simulator algorithms* $\mathtt{S}_1, \mathtt{S}_2$ *such that for any PPT adversary* $\mathcal{A}$, *the following experiments* $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^K)$ *and* $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^K)$ *are indistinguishable: (we use red parts to highlight the differences between these two experiments)*

$\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, 1^K)$:

1. On input the security parameter λ, the challenger runs $\mathsf{pp} \leftarrow \mathtt{MeSA.Setup}(1^\lambda)$ and provides pp to $\mathcal{A}$.
2. $\mathcal{A}$ outputs K messages $m_1, \ldots, m_K \in \overline{\mathcal{M}}$ and a subset $\mathcal{C} \subseteq [K]$.
3. The challenger runs $(\mathsf{ek}_k, \mathsf{ct}_k) \leftarrow \mathtt{MeSA.Enc}(m_k)$ for $k \in [K]$, then provides $\{\mathsf{ct}_k\}_{k\in[K]}$ and $\{\mathsf{ek}_k\}_{k\in\mathcal{C}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a function $f \in \mathcal{F}$ with K inputs if such f exists; otherwise, aborts this experiment and outputs a random bit b with $P(b = 0) = P(b = 1) = 0.5$.
5. The challenger runs $\mathsf{ct}' \leftarrow \mathtt{MeSA.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ and $\mathsf{ek}' \leftarrow \mathtt{MeSA.KEval}(f; \mathsf{ek}_1, \ldots, \mathsf{ek}_K)$, then sends $\mathsf{ct}', \mathsf{ek}'$ to $\mathcal{A}$.
6. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

$\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, 1^K)$:

1. On input the security parameter λ, the challenger runs $\mathsf{pp} \leftarrow \mathtt{MeSA.Setup}(1^\lambda)$ and provides pp to $\mathcal{A}$.
2. $\mathcal{A}$ outputs K messages $m_1, \ldots, m_K \in \overline{\mathcal{M}}$ and a subset $\mathcal{C} \subseteq [K]$.
3. The challenger runs $(\mathsf{ek}_1, \mathsf{ct}_1, \ldots, \mathsf{ek}_K, \mathsf{ct}_K) \leftarrow \mathtt{S}_1(1^\lambda, 1^K)$, then provides $\{\mathsf{ct}_k\}_{k\in[K]}$ and $\{\mathsf{ek}_k\}_{k\in\mathcal{C}}$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a function $f \in \mathcal{F}$ with K inputs if such f exists; otherwise, aborts this experiment and outputs a random bit b with $P(b = 0) = P(b = 1) = 0.5$.
5. The challenger runs $\mathsf{ct}' \leftarrow \mathtt{MeSA.Eval}(f; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$ and $\mathsf{ek}' \leftarrow \mathtt{S}_2(\{\mathsf{ek}_k\}_{k\in\mathcal{C}}, \mathsf{ct}', f(m_1, \ldots, m_K), \mathsf{st})$, then sends $\mathsf{ct}', \mathsf{ek}'$ to $\mathcal{A}$.
6. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

3.2 MeSA-Assisted HE Construction

Based on the above formulation, we now demonstrate how MeSA works to switch the message space of HE schemes. Specifically, we show how to construct an HE scheme over target message space by combining an existing HE scheme and a MeSA scheme together.

Construction 1 *This construction relies on the following two schemes:*

- *Let* $\underline{\mathtt{HE}} = (\underline{\mathtt{HE}}.\mathtt{Setup}, \underline{\mathtt{HE}}.\mathtt{KGen}, \underline{\mathtt{HE}}.\mathtt{Enc}, \underline{\mathtt{HE}}.\mathtt{Eval}, \underline{\mathtt{HE}}.\mathtt{Dec})$ *be an* $\mathcal{F}$*-homomorphic encryption scheme over message space* $\underline{\mathcal{M}}$*.*
- *Let* $\mathtt{MeSA} = (\mathtt{MeSA}.\mathtt{Setup}, \mathtt{MeSA}.\mathtt{Enc}, \mathtt{MeSA}.\mathtt{Eval}, \mathtt{MeSA}.\mathtt{KEval}, \mathtt{MeSA}.\mathtt{Dec})$ *be an* $\mathcal{F}$*-coherent message-space adapter scheme from* $\underline{\mathcal{M}}$ *to* $\overline{\mathcal{M}}$*.*

We construct a new $\mathcal{F}$*-homomorphic encryption scheme over message space* $\overline{\mathcal{M}}$*, denoted by* $\overline{\mathtt{HE}} = (\overline{\mathtt{HE}}.\mathtt{Setup}, \overline{\mathtt{HE}}.\mathtt{KGen}, \overline{\mathtt{HE}}.\mathtt{Enc}, \overline{\mathtt{HE}}.\mathtt{Eval}, \overline{\mathtt{HE}}.\mathtt{Dec})$*, as follows:*

- $\mathsf{pp} \leftarrow \overline{\mathtt{HE}}.\mathtt{Setup}(1^\lambda)$*: On input security parameter* λ*, this algorithm outputs* $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$ *by executing* $\mathsf{pp}_1 \leftarrow \mathtt{MeSA}.\mathtt{Setup}(1^\lambda)$ *and* $\mathsf{pp}_2 \leftarrow \underline{\mathtt{HE}}.\mathtt{Setup}(1^\lambda)$*.*
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \overline{\mathtt{HE}}.\mathtt{KGen}()$*: This algorithm outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$ *by executing* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \underline{\mathtt{HE}}.\mathtt{KGen}()$*.*
- $\mathsf{ct} \leftarrow \overline{\mathtt{HE}}.\mathtt{Enc}(m; \mathsf{pk})$*: On input a message* $m \in \overline{\mathcal{M}}$ *and* pk*, this algorithm outputs a ciphertext* $\mathsf{ct} = (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)})$ *by sequentially executing* $(\mathsf{ek}, \mathsf{ct}^{(1)}) \leftarrow \mathtt{MeSA}.\mathtt{Enc}(m)$ *and* $\mathsf{ct}^{(2)} \leftarrow \underline{\mathtt{HE}}.\mathtt{Enc}(\mathsf{ek}; \mathsf{pk})$*.*
- $\mathsf{ct}' \leftarrow \overline{\mathtt{HE}}.\mathtt{Eval}(f; \{\mathsf{ct}_k\}_{k\in[K]})$*: On input a* K*-ary function* $f \in \mathcal{F}$ *and a set of* K *ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_K$*, this algorithm parses the* $\{\mathsf{ct}_k\}_{k\in[K]}$ *as MeSA ciphertexts* $\{\mathsf{ct}_k^{(1)}\}_{k\in[K]}$ *and HE ciphertexts* $\{\mathsf{ct}_k^{(2)}\}_{k\in[K]}$*. Let* f^* *denote the algorithm* $\mathtt{MeSA}.\mathtt{KEval}(f; \{\mathsf{ek}_1, \ldots, \mathsf{ek}_K\})$*. It then executes* $\mathsf{ct}'^{(1)} = \mathtt{MeSA}.\mathtt{Eval}(f, \{\mathsf{ct}_k^{(1)}\}_{k\in[K]})$ *and* $\mathsf{ct}'^{(2)} = \underline{\mathtt{HE}}.\mathtt{Eval}(f^*; \{\mathsf{ct}_k^{(2)}\}_{k\in[K]})$*, where* f^* *denotes the circuit of* $\mathtt{MeSA}.\mathtt{KEval}(f, \{\mathsf{ek}_k\}_{k\in[K]})$*. Finally, it outputs* $\mathsf{ct}' = (\mathsf{ct}'^{(1)}, \mathsf{ct}'^{(2)})$*.*
- $m' \leftarrow \overline{\mathtt{HE}}.\mathtt{Dec}(\mathsf{ct}'; \mathsf{sk})$*: On input a ciphertext* ct' *and* sk*, this algorithm parses* ct' *as a MeSA ciphertext* $\mathsf{ct}'^{(1)}$ *and an HE ciphertext* $\mathsf{ct}'^{(2)}$*. It outputs* m' *by sequentially executing* $\mathsf{ek}' \leftarrow \underline{\mathtt{HE}}.\mathtt{Dec}(\mathsf{ct}'^{(2)}; \mathsf{sk})$ *and* $m' \leftarrow \mathtt{MeSA}.\mathtt{Dec}(\mathsf{ct}'^{(1)}; \mathsf{ek}')$*.*

The theorems of compactness, correctness, and semantic security for Construction 1 are stated as follows and formally proved in the full version [14].

Theorem 1 (Compactness of Construction 1). *Suppose* $\underline{\mathtt{HE}}$ *and* $\mathtt{MeSA}$ *satisfy compactness (Definitions 2 and 11), then the* $\overline{\mathtt{HE}}$ *from Construction 1 satisfies compactness (Definition 2).*

Theorem 2 (Correctness of Construction 1). *Suppose* $\underline{\mathtt{HE}}$ *and* $\mathtt{MeSA}$ *satisfy the requirements specified in Construction 1 as well as correctness (Definitions 3 and 12), then the* $\overline{\mathtt{HE}}$ *from Construction 1 satisfies correctness (Definition 3).*

Theorem 3 (Security of Construction 1). *Suppose* $\underline{\mathtt{HE}}$ *and* $\mathtt{MeSA}$ *satisfy semantic security (Definitions 4 and 13), then the* $\overline{\mathtt{HE}}$ *from Construction 1 satisfies semantic security (Definition 4).*

We note that the public/secret key pair of $\overline{\mathtt{HE}}$ is exactly the same as that of $\underline{\mathtt{HE}}$. The encryption keys of $\mathtt{MeSA}$ are only one-time nonce that cannot be reused across different messages. Consequently, the key generation and decryption algorithms of $\overline{\mathtt{HE}}$ directly invoke those algorithms of $\underline{\mathtt{HE}}$, without modifying their internal mechanism. This also facilitates the design of threshold variant of $\overline{\mathtt{HE}}$, as shown in the next subsection.

3.3 MeSA-Assisted ThHE Construction

Next, we discuss how to obtain a threshold HE scheme with target message-space via MeSA. Particularly, given $\underline{\mathtt{ThHE}}$ as a threshold variant of $\underline{\mathtt{HE}}$, we can construct a threshold variant of $\overline{\mathtt{HE}}$ by directly invoking the key generation and decryption algorithms of $\underline{\mathtt{ThHE}}$. We demonstrate this by Construction 2

Construction 2 *This construction relies on the following two schemes:*

- *Let* $\underline{\mathtt{ThHE}} = (\underline{\mathtt{ThHE}}.\mathtt{Setup}, \underline{\mathtt{ThHE}}.\mathtt{KGen}, \underline{\mathtt{ThHE}}.\mathtt{Enc}, \underline{\mathtt{ThHE}}.\mathtt{Eval}, \underline{\mathtt{ThHE}}.\mathtt{PartDec}, \underline{\mathtt{ThHE}}.\mathtt{FinDec})$ *be a threshold* $\mathcal{F}$*-HE scheme over message space* $\underline{\mathcal{M}}$*.*
- *Let* $\mathtt{MeSA} = (\mathtt{MeSA}.\mathtt{Setup}, \mathtt{MeSA}.\mathtt{Enc}, \mathtt{MeSA}.\mathtt{Eval}, \mathtt{MeSA}.\mathtt{KEval}, \mathtt{MeSA}.\mathtt{Dec})$ *be an* $\mathcal{F}$*-coherent message-space adapter scheme from* $\underline{\mathcal{M}}$ *to* $\overline{\mathcal{M}}$*.*

We construct a threshold $\mathcal{F}$*-homomorphic encryption scheme over message space* $\overline{\mathcal{M}}$*, i.e.,* $\overline{\mathtt{ThHE}} = (\overline{\mathtt{ThHE}}.\mathtt{Setup}, \overline{\mathtt{ThHE}}.\mathtt{KGen}, \overline{\mathtt{ThHE}}.\mathtt{Enc}, \overline{\mathtt{ThHE}}.\mathtt{Eval}, \overline{\mathtt{ThHE}}.\mathtt{PartDec}, \overline{\mathtt{ThHE}}.\mathtt{FinDec})$*, as follows:*

- $\mathsf{pp} \leftarrow \overline{\mathtt{ThHE}}.\mathtt{Setup}(1^\lambda, N, T)$*: On input security parameter* λ*, the number of parties* N*, and the threshold* T*, the setup algorithm executes* $\mathsf{pp}_1 \leftarrow \mathtt{MeSA}.\mathtt{Setup}(1^\lambda)$ *and* $\mathsf{pp}_2 \leftarrow \underline{\mathtt{ThHE}}.\mathtt{Setup}(1^\lambda, N, T)$*. It then outputs public parameters* $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$*.*
- $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \overline{\mathtt{ThHE}}.\mathtt{KGen}()$*: The key generation algorithm outputs a public key* pk *and* N *secret key shares* $\{\mathsf{sk}_i\}_{i\in[N]}$ *by executing* $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \underline{\mathtt{ThHE}}.\mathtt{KGen}()$*.*
- $\mathsf{ct} \leftarrow \overline{\mathtt{ThHE}}.\mathtt{Enc}(m; \mathsf{pk})$*: On input a message* $m \in \overline{\mathcal{M}}$ *and the public key* pk*, the encryption algorithm outputs a ciphertext* $\mathsf{ct} = (\mathsf{ct}^{(1)}, \mathsf{ct}^{(2)})$ *by sequentially executing* $(\mathsf{ek}, \mathsf{ct}^{(1)}) \leftarrow \mathtt{MeSA}.\mathtt{Enc}(m)$ *and* $\mathsf{ct}^{(2)} \leftarrow \underline{\mathtt{ThHE}}.\mathtt{Enc}(\mathsf{ek}; \mathsf{pk})$*.*
- $\mathsf{ct}' \leftarrow \overline{\mathtt{ThHE}}.\mathtt{Eval}(f; \{\mathsf{ct}_k\}_{k\in[K]})$*: On input a* K*-ary function* $f \in \mathcal{F}$ *and a set of* K *ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_K$*, the evaluation algorithm parses the* $\{\mathsf{ct}_k\}_{k\in[K]}$ *as MeSA ciphertexts* $\{\mathsf{ct}_k^{(1)}\}_{k\in[K]}$ *and HE ciphertexts* $\{\mathsf{ct}_k^{(2)}\}_{k\in[K]}$*. Let* f^* *denote the algorithm* $\mathtt{MeSA}.\mathtt{KEval}(f; \{\mathsf{ek}_1, \ldots, \mathsf{ek}_K\})$*. It then executes* $\mathsf{ct}'^{(1)} \leftarrow \mathtt{MeSA}.\mathtt{Eval}(f, \{\mathsf{ct}_k^{(1)}\}_{k\in[K]})$ *and* $\mathsf{ct}'^{(2)} \leftarrow \underline{\mathtt{ThHE}}.\mathtt{Eval}(f^*; \{\mathsf{ct}_k^{(2)}\}_{k\in[K]})$*, where* f^* *denotes the circuit of* $\mathtt{MeSA}.\mathtt{KEval}(f, \{\mathsf{ek}_k\}_{k\in[K]})$*. Finally, it outputs* $\mathsf{ct}' = (\mathsf{ct}'^{(1)}, \mathsf{ct}'^{(2)})$*.*
- $\mathsf{ds}_i \leftarrow \overline{\mathtt{ThHE}}.\mathtt{PartDec}(\mathsf{ct}'; \mathsf{sk}_i)$*: On input a ciphertext* ct' *and a secret key share* sk_i*, the partial decryption algorithm parses* ct' *as a MeSA ciphertext* $\mathsf{ct}'^{(1)}$ *and an HE ciphertext* $\mathsf{ct}'^{(2)}$*. It then outputs a decryption share* ds_i *by executing* $\mathsf{ds}_i \leftarrow \underline{\mathtt{ThHE}}.\mathtt{PartDec}(\mathsf{ct}'^{(2)}; \mathsf{sk}_i)$*.*
- $m' \leftarrow \overline{\mathtt{ThHE}}.\mathtt{FinDec}(\mathsf{ct}'; \{\mathsf{ds}_i\}_{i\in\mathcal{T}})$*: On input a ciphertext* ct' *and the decryption shares* $\{\mathsf{ds}_i\}_{i\in\mathcal{T}}$ *from set* $\mathcal{T}$*, the decryption algorithm parses* ct' *as a MeSA ciphertext* $\mathsf{ct}'^{(1)}$ *and an HE ciphertext* $\mathsf{ct}'^{(2)}$*. It then outputs* m' *by sequentially executing* $\mathsf{ek}' \leftarrow \underline{\mathtt{ThHE}}.\mathtt{FinDec}(\mathsf{ct}'^{(2)}; \{\mathsf{ds}_i\}_{i\in\mathcal{T}})$ *and* $m' \leftarrow \mathtt{MeSA}.\mathtt{Dec}(\mathsf{ct}'^{(1)}; \mathsf{ek}')$*.*

The theorems of compactness, correctness, and simulation-based security for Construction 2 are stated as follows and formally proved in the full version [14].

Theorem 4 (Compactness of Construction 2). *Suppose* $\underline{\texttt{ThHE}}$ *and* $\texttt{MeSA}$ *satisfy compactness (Definitions 6 and 11), then the* $\overline{\texttt{ThHE}}$ *from Construction 2 satisfies compactness (Definition 6).*

Theorem 5 (Correctness of Construction 2). *Suppose that* $\underline{\texttt{ThHE}}$ *is a threshold* $\mathcal{F}$*-homomorphic encryption scheme over message space* $\underline{\mathcal{M}}$ *that satisfies correctness (Definition 7) and* $\underline{\texttt{MeSA}}$ *is an* $\mathcal{F}$*-coherent message-space adapter scheme from* $\underline{\mathcal{M}}$ *to* $\overline{\mathcal{M}}$ *that satisfies correctness (Definition 12), then the* $\overline{\texttt{ThHE}}$ *from Construction 2 satisfies correctness (Definition 7).*

Theorem 6 (Security of Construction 2). *Suppose that* $\underline{\texttt{ThHE}}$ *is a threshold* $\mathcal{F}$*-homomorphic encryption scheme that satisfies simulation-based security (Definition 8) and* $\underline{\texttt{MeSA}}$ *is an* $\mathcal{F}$*-coherent message-space adapter scheme that satisfies simulation-based security (Definition 14), then the* $\overline{\texttt{ThHE}}$ *from Construction 2 satisfies simulation-based security (Definition 8).*

In addition, the operations of decryption process that involve the collaboration among parties are exactly the same between $\overline{\texttt{ThHE}}$ and $\underline{\texttt{ThHE}}$. Actually, the only additional operation in $\overline{\texttt{ThHE}}$'s decryption process beyond $\underline{\texttt{ThHE}}$'s decryption process is the $\texttt{MeSA.Dec}$ algorithm whose complexity is independent of N. Hence, the $\overline{\texttt{ThHE}}$ and $\underline{\texttt{ThHE}}$ schemes have the same complexity with respect to N. This inspires us to develop a customized MeSA scheme for Exp-ElGamal, so that the small-message restriction can be mitigated while the decryption algorithms of new HE scheme can enjoy low complexity as Exp-ElGamal.

4 Lattice-Exp-ElGamal and Its Threshold Variants

We use lattice-based encryption to develop a customized MeSA for Exp-ElGamal in Section 4.1 and then construct (threshold) Lattice-Exp-ElGamal along with its performance analysis in Section 4.2. We also validate the complexity of Lattice-Exp-ElGamal via experiments in Section 4.3.

4.1 Lattice-Based MeSA for Exp-ElGamal

Exp-ElGamal has two features that affect the design of its customized MeSA. One is its linearly-homomorphic property. The other is that the messages of Exp-ElGamal are usually set to be numbers or vectors over the interval $[a, b]$ with $b - a$ being sufficiently small. Hence, to apply Exp-ElGamal as the $\underline{\texttt{HE}}$ in Construction 1, we need a linearly-coherent MeSA scheme from small message-space $\underline{\mathcal{M}}$ to some target space $\overline{\mathcal{M}}$. By adapting lattice-based encryption, below we instantiate such a MeSA scheme with $a = -b$ for some small b and $\overline{\mathcal{M}} = \mathbb{Z}_P$ for some integer P.

Construction 3 *We construct a lattice-based MeSA* $\texttt{LM}$ *from message-space* $\underline{\mathcal{M}} = [-b, b]^{D \times L}$ *to message-space* $\overline{\mathcal{M}} = \mathbb{Z}_P$*, which is coherent for linear functions in the form of* K *coefficients* $\alpha_1, \ldots, \alpha_K \in \mathbb{Z}_P$*, as follows.*

– $\mathsf{pp} \leftarrow \mathtt{LM.Setup}(1^\lambda)$*: On input security parameter* λ*, this algorithm outputs* pp *that consists of an integer* Q *for modulus, an integer* D *for vector length, an integer* R *for digital base, an integer* $L = \lceil \log_R P \rceil$*, an integer* $\Delta = \lfloor Q/P \rfloor$*, a uniformly-random vector* $\mathbf{a} \leftarrow\!\!\$\ \mathbb{Z}_Q^D$*, and a* B*-bounded distribution* $\chi_{e,B}$*.*

– $(\mathsf{ek}, \mathsf{ct}) \leftarrow \mathtt{LM.Enc}(m)$*: On input a message* $m \in \mathbb{Z}_P$*, this algorithm executes as follows for each* $l \in [L]$*: 1) samples a size-*D *vector* $\mathbf{S}_l$ *with* $Pr(\mathbf{S}_{l,i} = -1) = Pr(\mathbf{S}_{l,i} = 1) = 0.25$ *and* $Pr(\mathbf{S}_{l,i} = 0) = 0.5$ *for* $i \in [D]$ *and 2) computes* $\mathbf{c}_l = \mathbf{a}^T\mathbf{S}_l + e_l + \Delta \cdot [m_l]_P$ *with* $e_l \leftarrow\!\!\$\ \chi_{e,B}$ *and* $m_l = m \cdot R^{l-1} \mod P$*. After concatenating all* $\mathbf{S}_l$ *as a matrix* $\mathbf{S}$ *and all* $\mathbf{c}_l$ *as a vector* $\mathbf{c}$*, it outputs* $\mathsf{ek} = \mathbf{S}, \mathsf{ct} = \mathbf{c}$*.*

– $\mathsf{ct}' \leftarrow \mathtt{LM.Eval}(\alpha_1, \ldots, \alpha_K; \mathsf{ct}_1, \ldots, \mathsf{ct}_K)$*: On input* K *coefficients and ciphertexts, this algorithm computes a size-*L *vector* $\mathbf{c}'$ *as follows and outputs* $\mathsf{ct}' = \mathbf{c}'$*. First, for each* $l \in [L]$ *and each* $k \in [K]$*, computes* $\alpha_{k,l} = \alpha_k \cdot R^{l-1} \mod P$ *and rewrites* $\alpha_{k,l} = \sum_{l' \in [L]} \alpha_{k,l}^{(l')} \cdot R^{l'-1}$ *with* $\alpha_{k,l}^{(l')} \in \{0, \ldots, R-1\}$*. Second, for each* $l \in [L]$*, computes* $\mathbf{c}'_l = \sum_{k \in [K]} \sum_{l' \in [L]} \mathsf{ct}_{k,l'} \cdot \alpha_{k,l}^{(l')}$*. Third, concatenates all* $\mathbf{c}'_l$ *as a vector* $\mathbf{c}'$*.*

– $\mathsf{ek}' \leftarrow \mathtt{LM.KEval}(\alpha_1, \ldots, \alpha_K; \mathsf{ek}_1, \ldots, \mathsf{ek}_K)$*: On input* K *coefficients and encryption keys, this algorithm computes an* L*-by-*D *matrix* $\mathbf{S}'$ *as follows and outputs* $\mathsf{ek}' = \mathbf{S}'$*. First, for each* $l \in [L]$ *and each* $k \in [K]$*, computes* $\alpha_{k,l} = \alpha_k \cdot R^{l-1} \mod P$ *and rewrites* $\alpha_{k,l} = \sum_{l' \in [L]} \alpha_{k,l}^{(l')} \cdot R^{l'-1}$ *with* $\alpha_{k,l}^{(l')} \in \{0, \ldots, R-1\}$*. Second, for each* $l \in [L]$*, computes* $\mathbf{S}'_l = \sum_{k \in [K]} \sum_{l' \in [L]} \mathsf{ek}_{k,l'} \cdot \alpha_{k,l}^{(l')}$*. Third, concatenates all* $\mathbf{S}'_l$ *as a matrix* $\mathbf{S}'$*.*

– $m' \leftarrow \mathtt{LM.Dec}(\mathsf{ct}', \mathsf{ek}')$*: On input a ciphertext* ct' *and an encryption key* ek'*, this algorithm parses* ct' *as a vector* $\mathbf{c}'$ *and* ek' *as a matrix* $\mathbf{S}'$*, then outputs*

$$m' = \left\lfloor \frac{P \cdot [\mathbf{c}'_1 - \mathbf{a}^T\mathbf{S}'_1]_Q}{Q} \right\rceil \mod P.$$

The lattice-based encryption has the following decent properties to enable the construction of LM. First, its encryption keys have ternary values and thus match the message space of Exp-ElGamal. Second, it enables a flexible choice on message space $\mathbb{Z}_P$, as P is neither necessarily a prime nor co-prime with other integers. Third, it can be made linearly homomorphic in messages and keys both, making LM a linearly-coherent MeSA by Definition 10.

Despite these advantages, we need to make the following modifications to adapt the lattice-based encryption for MeSA construction. On one hand, the vector $\mathbf{a}$ is generated by LM.Setup algorithm as part of pp, while the encryption key ek is sampled by LM.Enc algorithm. In this way, the encryption of different messages uses the same $\mathbf{a}$ but distinct encryption keys, rather than the same encryption key but different $\mathbf{a}$. This also enables the LM to be linearly homomorphic in both of messages and keys, as mentioned in Section 2.4. On the other hand, instead of encrypting a single message m, we encrypt $m, R \cdot m, \ldots, R^{L-1} \cdot m$ with $L = \lceil \log_R P \rceil$ for some digital base R. As illustrated in Section 1.2, this helps to reduce the growth of errors and encryption keys when evaluating ciphertexts, at the cost of operating more ciphertexts. Particularly, when evaluating over

at most K_{max} input ciphertexts, the error in the output ciphertext is at most $B' = K_{max} \cdot B \cdot L \cdot (R-1)$ and the encryption key ek' corresponding to the output ciphertext has a norm bounded by $K_{max} \cdot L \cdot (R-1)$.

Even though the growth of noises and encryption keys become slower, evaluating over too many ciphertexts still makes the errors and encryption keys in the output ciphertext exceed the pre-specified bound, resulting in decryption failures. Recall that the requirements of successful decryption for lattice-based encryption is $B' < \Delta/2$, where B' is the upper-bound of error in the output ciphertext. In addition, to make the encryption key ek' of the output ciphertext still in the $\underline{\mathcal{M}} = [-b, b]$, we need to guarantee that $\left\|\mathsf{ek}'\right\| < b$. As a result, the LM needs to obey the following condition to satisfy the correctness requirement:

Condition 1 *$K_{max} \cdot B \cdot L \cdot (R-1) < \Delta/2$ and $K_{max} \cdot L \cdot (R-1) < b$.*

Based on the above discussions, we now state the theorems of LM for compactness, correctness, and security. Their proofs are put in the full version [14].

Theorem 7 (Compactness of Construction 3). LM *is a MeSA scheme that satisfies compactness (Definition 11).*

Theorem 8 (Correctness of Construction 3). *Suppose that the parameters of* LM *satisfy Condition 1. Then* LM *is a correct MeSA scheme (Definition 12).*

Theorem 9 (Security of Construction 3). *Suppose that the LWE problem is hard under the parameters of* LM*, then* LM *satisfies both of semantic security and simulation-based security (Definitions 13 and 14). Furthermore, if the LWE problem is κ-bit hard, then* LM *is $(\kappa - \mathcal{O}(\log K))$-bit secure in the simulation-based sense, i.e., has logarithm security loss in the number of operated keys K.*

4.2 (Threshold) Lattice-Exp-ElGamal: Design and Analysis

By setting HE to be the Exp-ElGamal and MeSA = LM, we obtain a new LHE scheme by Construction 1. We term it Lattice-Exp-ElGamal. Similarly, by setting ThHE to be the threshold Exp-ElGamal and MeSA = LM, we also obtain the threshold version of Lattice-Exp-ElGamal by Construction 2. Thanks to the LM scheme, the message space of (threshold) Lattice-Exp-ElGamal is switched to the $\mathbb{Z}_P$, where P is neither necessarily a prime nor co-prime with other integers. When applying (threshold) Lattice-Exp-ElGamal to some application, one can freely set the value of P to satisfy the application demands, which mitigates the small-message restriction of vanilla (threshold) Exp-ElGamal. Due to the page limit, we put the description of these schemes in the full version [14].

With the establishment of LM's properties, including compactness, correctness, and security, the properties of (threshold) Lattice-Exp-ElGamal directly hold as natural results of Theorems 1 – 6, as stated below.

Corollary 1 (Compactness of (Threshold) Lattice-Exp-ElGamal). *The Lattice-Exp-ElGamal and threshold Lattice-Exp-ElGamal both satisfy compactness (Definitions 2 and 6).*

Corollary 2 (Correctness of (Threshold) Lattice-Exp-ElGamal). *Suppose that the parameters of* LM *satisfy Condition 1. Then Lattice-Exp-ElGamal is an LHE scheme that satisfies correctness (Definition 3) and threshold Lattice-Exp-ElGamal is a ThLHE scheme that satisfies correctness (Definition 7).*

Corollary 3 (Security of (Threshold) Lattice-Exp-ElGamal). *Suppose that LWE and DDH problems are hard under the parameters of* LM *and Exp-ElGamal, Lattice-Exp-ElGamal satisfies semantic security (Definition 4) and threshold Lattice-Exp-ElGamal satisfies simulation-based security (Definition 8).*

By Corollary 2 (in particular, Condition 1), there exists an upper bound on K to guarantee the correctness of (threshold) Lattice-Exp-ElGamal. Specifically, we need to set $K < \min\{b, \Delta/2B\}/(L \cdot (R-1))$. This provides several guidelines for parameter instantiation in practice.

1. Since K represents the number of ciphertexts to be evaluated, we need to set the upper bound $\min\{b, \Delta/2B\}/(L \cdot (R-1))$ to be large enough.
2. One way to increase the upper bound is to choose a smaller $L \cdot (R-1)$. Considering $L = \lceil \log_R P \rceil$ with P being the modulus of plaintext space, given a fixed P, a smaller value of R (e.g., $R = 2$) helps to reduce the value of $L \cdot (R-1)$ and to increase the upper bound. Nonetheless, a smaller value of R also leads to a larger value of L and more operations in the encryption and evaluation algorithms, which degrades the efficiency.
3. The other way to increase the upper bound is to choose a larger b and $\Delta/2B$. However, a larger value of b or $\Delta/2B$ results in other negative effects. On one hand, a larger value of b means more time for solving the discrete logarithm and thus lower efficiency. On the other hand, the setting of $\Delta/2B$ affects the hardness of underlying RLWE problem as well as the security of Lattice-Exp-ElGamal. Hence, one needs to set the values of Δ and B very cautiously. As suggested by the HE standard [1], a larger value of $\Delta/2B$ requires to increase the value of D to maintain the same level of security. Nonetheless, a larger value of D also implies lower efficiency for encryption and decryption.

Based on the above discussions, there exists a tradeoff between the number of input ciphertexts for evaluation and the efficiency of (threshold) Lattice-Exp-ElGamal. We leave the concrete parameter instantiation for target application as an open problem for future research. Nonetheless, to highlight the superiority of (threshold) Lattice-Exp-ElGamal over vanilla (threshold) Exp-ElGamal, we use an instance to estimate the capability of (threshold) Lattice-Exp-ElGamal for large message space and the evaluation over multiple ciphertexts.

Remark 2. Suppose that we fix $b = 10^6$ for efficient decryption of Exp-ElGamal. By setting $\Delta/2B > b$, one can meet Condition 1 as long as $K \leq b/(L \cdot (R-1))$. To pursue the maximum value of K, we set $R = 2$ and thus $L = \lceil \log_2 P \rceil$. Correspondingly, the maximum value of K in this case is $10^6/\lceil \log_2 P \rceil$. If some application involves the 64-bit messages (i.e., $\lceil \log_2 P \rceil = 64$), then (threshold) Lattice-Exp-ElGamal can support the evaluation over $K = 15625$ ciphertexts,

which can meet the demands of most of existing applications. For comparison, the vanilla (threshold) Exp-ElGamal even fails to support the decryption of 64-bit messages, as a single message may exceed b ($2^{64} \gg 10^6$).

Finally, we compare the complexity between Lattice-Exp-ElGamal and existing baselines. Particularly, we consider two state-of-the-art ThLHE baselines under the condition of no utility restrictions on small-message, full-threshold, and known-participants. One is threshold paillier [27,32,34], which can represent all ThLHE schemes with unknown group order (i.e., factoring-based ThLHE schemes [40] and existing ThLHE schemes from the CL framework [11,13]) as they have the same complexity. The other is ThFHE scheme from Shamir secret sharing [7], which is the optimal one among existing ThFHE schemes.

Our comparison consists of two parts. One is the complexity of decryption algorithms, which is the main focus of this work. For supplementary comparison, we also discuss the complexity of encryption and evaluation algorithms. In each part, we consider the complexity with respect to three variables: the number of parties N, the message size M, and the MeSA's key size D.

Complexity of decryption algorithms. For threshold decryption, the computation complexity is the summation of two decryption algorithms, while the communication complexity is the size of each decryption share. The analysis are detailed in the full version [14] and summarized in Table 1.

Table 1. Comparison between threshold Lattice-Exp-ElGamal and existing baselines in terms of the complexity of decryption algorithms.

	Threshold Paillier [27,32,34]	Shamir-based ThFHE [7]	Threshold Lattice-Exp-ElGamal
Computation Complexity	$\mathcal{O}(N^2 \log N \cdot M)$	$\mathcal{O}(N^2 \log N \cdot M)$	$\mathcal{O}(N \log N + ND + M)$
Communication Complexity	$\mathcal{O}(M)$	$\mathcal{O}(N \log N \cdot M)$	$\mathcal{O}(D)$

Among those ThLHE schemes without those utility restrictions, threshold Lattice-Exp-ElGamal is the first scheme with quasi-linear computation complexity with respect to N for threshold decryption. Furthermore, when the MeSA's key size D is large enough, we can even omit the $N \log N$ term and safely regard the computation complexity of Lattice-Exp-ElGamal as linear order. This validates that our threshold Lattice-Exp-ElGamal truly resolves the efficiency-utility dilemma. In addition, our scheme enjoys *constant* communication complexity with respect to N, confirming its advantage for low communication cost.

As for the complexity with respect to message size M and MeSA's key size D, we observe that existing baselines have a multiplicative factor in M, while our scheme has a multiplicative factor in D but only an additive factor in M. This is mainly caused by the fact that existing schemes decrypt messages directly through party-to-party cooperations, while our scheme recovers MeSA's key through party-to-party cooperations and then decrypt messages without any cooperations. Due to this observation, our scheme has more significant advantage when the size of messages is larger than that of MeSA's keys.

Complexity of encryption and evaluation algorithms. In this part, we only analyze the complexity with respect to D and M, as the encryption and evaluation algorithms have constant complexity with respect to N due to no party-to-party cooperation.

In threshold Lattice-Exp-ElGamal, the algorithms of encryption, evaluation, and decryption deal with not only messages but also MeSA's keys, while existing schemes only handle messages. Hence, the complexity of our scheme (resp. existing schemes) can be estimated as $\mathcal{O}(D+M)$ (resp. $\mathcal{O}(M)$). If $D << M$, the cost of handling keys can be neglected. Otherwise, handling keys may bring extra burden. This inspires us to choose MeSA with a small value of D/M.

4.3 Empirical Evaluation

We empirically evaluate the running time of threshold Lattice-Exp-ElGamal and threshold Paillier to validate the theoretical analysis results on computation complexity. In the experiment, we adopt multiple values of N and T to observe their impacts on running time. Particularly, we set N to range from 100 to 1000 and $T \in \{0.5N, 0.7N, 0.9N\}$. The more detailed discussions on the implementation can be found in the full version [14].

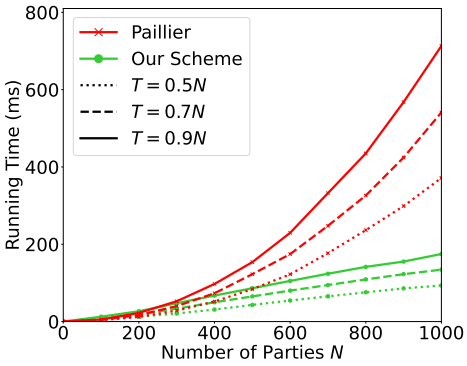

Fig. 2. Running time of decryption algorithms of threshold Paillier and threshold Lattice-Exp-ElGamal (our scheme) with different values of N and T.

In Fig. 2, we show the running time of the summation of two decryption algorithms. Regarding the growth of running time with N, we can observe a quasi-linear growth trend of threshold Lattice-Exp-ElGamal and a super-quadratic growth trend of threshold Paillier, which aligns with our theoretical analysis. As a result, although Lattice-Exp-ElGamal is slightly slower than Paillier for small N, our scheme achieves four times acceleration when $N = 1000$. Meanwhile, this observation holds for different values of T.

5 New Application of ThLHE: Multi-Party FHE

Below we first provide preliminaries on MFHE in Sec. 5.1 and then present our ThLHE-based MFHE construction along with performance analysis in Sec. 5.2.

5.1 Preliminaries on MFHE

We provide the definition of (single-key) multi-party FHE [44] as follows.

Definition 15 (Multi-Party FHE). *A T-out-of-N multi-party FHE scheme is a tuple* $\mathtt{MFHE} = (\mathtt{MFHE.Setup}, \mathtt{MFHE.KGen}, \mathtt{MFHE.Enc}, \mathtt{MFHE.Eval}, \mathtt{MFHE.}\Pi_{\mathtt{Dec}})$ *of algorithms and multiparty protocols, so that the first four algorithms have the same definition as the ThFHE and the last protocol is defined as follows:*
***Decryption**: $m' \leftarrow \mathtt{MFHE.}\Pi_{\mathtt{Dec}}(\mathsf{ct}'; \mathsf{sk}_1, \ldots, \mathsf{sk}_N)$. Given a ciphertext ct, this protocol outputs a message m', under the following condition: When this protocol ask parties to send some information to other parties, there are at least T parties (called participants) who can do so within the required time.*

In this work, we focus on MFHE scheme with *arbitrary*-threshold (i.e., $1 \leq T \leq N$) and dynamic participants (i.e., the participants can be any T parties rather than from a pre-determined set). The recent work [15] establishes a reduction from RLWE-based MFHE to a novel primitive called *approximate secret sharing* (ApproxSS). In other words, given an arbitrary-threshold ApproxSS scheme satisfying specified requirements, one can directly obtain an arbitrary-threshold MFHE scheme. Below we review the definition of ApproxSS.

Definition 16 (ApproxSS). *For a message space $\mathcal{M}$, a T-out-of-N approximate secret sharing is a pair* $\mathtt{ApproxSS} = (\mathtt{ApproxSS.Share}, \mathtt{ApproxSS.}\Pi_{\mathtt{ApproxRec}})$ *defined as follows:*
***Share**: $\{s_i\}_{i\in[N]} \leftarrow \mathtt{ApproxSS.Share}(m)$. On input a message $m \in \mathcal{M}$, this algorithm outputs N shares $\{s_i\}_{i\in[N]}$;*
***Approximate Recovery**: $m' \leftarrow \mathtt{ApproxSS.}\Pi_{\mathtt{ApproxRec}}(\{s_i\}_{i\in[N]}, \chi)$. On input shares $\{s_i\}_{i\in[N]}$ and a distribution χ on $\mathcal{M}$, this protocol outputs a common approximate message $m' \in \mathcal{M}$ to all parties under the following condition: When this protocol ask parties to send some information to other parties, there are at least T parties (called participants) who can do so within the required time.*

Definition 17 (Approximate Correctness). *An ApproxSS satisfies $\mathcal{M}_B$-approximate correctness if and only if for the shares of any message $\{s_i\}_{i\in[N]} \leftarrow \mathtt{ApproxSS.Share}(m)$, the recovered $m' \leftarrow \mathtt{ApproxSS.}\Pi_{\mathtt{ApproxRec}}(\{s_i\}_{i\in[N]}, \chi)$ satisfies $m' - m \in \mathcal{M}_B$.*

Definition 18 (Approximate Security). *For an ApproxSS whose $\Pi_{\mathtt{ApproxRec}}$ consists of R rounds, it satisfies χ-approximate security if for any security parameter λ the following claim holds: There exists $R+1$ PPT simulator algorithms $(S_0, S_1, \ldots, S_R)$ such that for any PPT adversary $\mathcal{A}$, the following two experiments $\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, \chi)$ and $\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, \chi)$ are indistinguishable.*

$\mathsf{Expt}_{\mathcal{A},\mathsf{Real}}(1^\lambda, \chi)$:

1. On input the security parameter λ, $\mathcal{A}$ outputs $N, T, \mathcal{M}$, a message $m \in \mathcal{M}$, and a set $\mathcal{N}_A$ of $T-1$ corrupted clients.
2. The challenger runs $\{s_i\}_{i\in[N]} \leftarrow \mathtt{Share}(m)$ and sends $\{s_i\}_{i\in\mathcal{N}_A}$ to $\mathcal{A}$.

3. The $\Pi_{\texttt{ApproxRec}}$ protocol is performed as follows: In each round r, $\mathcal{A}$ selects a set $\mathcal{T}_r$ consisting of T participants, then the challenger runs the r-th round of the protocol for parties in $\mathcal{T}_r$ and provides the transcript Trans_r that can be observed by parties in $\mathcal{N}_A$ to $\mathcal{A}$.
4. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

$\mathsf{Expt}_{\mathcal{A},\mathsf{Ideal}}(1^\lambda, \chi)$:

1. On input the security parameter λ, $\mathcal{A}$ outputs $N, T, \mathcal{M}$, a message $m \in \mathcal{M}$, and a set $\mathcal{N}_A$ of $T-1$ corrupted clients.
2. The challenger runs $\{s_i\}_{i\in\mathcal{N}_A} \leftarrow S_0(N, T, \mathcal{M})$ and sends $\{s_i\}_{i\in\mathcal{N}_A}$ to $\mathcal{A}$.
3. The challenger samples $m_\chi \leftarrow m + x$ with $x \leftarrow\!\!\$\ \chi$ and then simulates the $\Pi_{\texttt{ApproxRec}}$ protocol as follows: In each round r, $\mathcal{A}$ selects a set $\mathcal{T}_r$ consisting of T participants, then the challenger generates transcript Trans'_r by running $\mathsf{Trans}'_r \leftarrow S_r(m_\chi, \{s_i\}_{i\in\mathcal{N}_A}, \mathcal{T}_r)$ and sends Trans'_r to $\mathcal{A}$
4. At the end of this experiment, $\mathcal{A}$ outputs a distinguishing bit b.

With the above definition, the existing work [15] demonstrates the following lemma to establish the properties of MFHE over those of ApproxSS.

Lemma 1. *Suppose that χ is a B_0-bounded distribution and the error in MFHE's ciphertext is bounded by B_{CT}. If there exists a T-out-of-N linear ApproxSS scheme with χ-approximate security and $\mathcal{M}_B$-approximate correctness, then there exists an MFHE scheme with correctness and simulation-based security if $B_{\mathsf{CT}} + T \cdot B_0 < \Delta/2$ and $B_{\mathsf{CT}}/B_0 \in \mathsf{negl}$.*

5.2 ApproxSS via Threshold Lattice-Exp-ElGamal

Below we provide a ThLHE-based ApproxSS construction, in which we use $\mathcal{T}_r$ to denote the set of participants in the r-th round. We note that $\mathcal{T}_r$ is just a notation used to mark those actual participants (could be any T parties) and Construction 4 does not require parties to learn or agree on $\mathcal{T}_r$ in advance.

Construction 4 *Suppose that* $\texttt{ThLHE} = (\texttt{ThLHE.Setup}, \texttt{ThLHE.KGen}, \texttt{ThLHE.Enc}, \texttt{ThLHE.Eval}, \texttt{ThLHE.PartDec}, \texttt{ThLHE.FinDec})$ *is a T-out-of-N ThLHE scheme over message space $\mathcal{M}$. We construct a T-out-of-N ThLHE-based ApproxSS scheme, denoted by* $\texttt{TASS} = (\texttt{TASS.Share}, \texttt{TASS.}\Pi_{\texttt{ApproxRec}})$*, as follows:*

- $\{s_i\}_{i\in[N]} \leftarrow \texttt{TASS.Share}(m)$*: On input a message m, this algorithm sequentially executes* $\mathsf{pp} \leftarrow \texttt{ThLHE.Setup}(1^\lambda)$, $\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]} \leftarrow \texttt{ThLHE.KGen}()$*, and* $\mathsf{ctm} \leftarrow \texttt{ThLHE.Enc}(m; \mathsf{pk})$*. It then outputs $s_i = (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}_i, \mathsf{ctm})$.*
- $m' \leftarrow \texttt{TASS.}\Pi_{\texttt{ApproxRec}}(\{s_i\}_{i\in[N]}, \chi)$*: On input N shares and a distribution χ, this protocol outputs m' with two communication rounds.*
  - ◇ ***Round 1****: Each party $i \in \mathcal{T}_1$ samples a $n_i \leftarrow\!\!\$\ \chi$ and executes* $\mathsf{ctn}_i \leftarrow \texttt{ThLHE.Enc}(n_i; \mathsf{pk})$*. It then sends ctn_i to every parties.*
  - ◇ ***Round 2****: Each party $j \in \mathcal{T}_2$ runs* $\mathsf{ctm}' \leftarrow \texttt{ThLHE.Eval}(f; \mathsf{ctm}, \{\mathsf{ctn}_i\}_{i\in\mathcal{T}_1})$ *with $f(\{x_i\}_{i\in[T+1]}) = \sum_{i\in[T+1]} x_i$ and* $\mathsf{ds}_i \leftarrow \texttt{ThLHE.PartDec}(\mathsf{ctm}'; \mathsf{sk}_i)$*. It then sends ctm' to every parties.*

◇ ***Finalization****: Outputs* $m' \leftarrow \mathtt{ThLHE.FinDec}(\mathsf{ctm}'; \{\mathsf{ds}_i\}_{i\in\mathcal{T}})$.

We state the correctness and security theorem of TASS as follows (see the full version [14] for their formal proofs).

Theorem 10 (Correctness and Security of Construction 4). *Suppose that* ThLHE *satisfies correctness and simulation-based security (Definitions 7 and 8). If* χ *is a* B_0*-bounded distribution, then* TASS *is an ApproxSS scheme that satisfies* χ*-approximate security under RLWE hardness assumption and* $\mathcal{M}_B$*-approximate correctness with* $\mathcal{M}_B = \{n \mid \|n\| \leq T \cdot B_0\}$ *(Definitions 17 and 18).*

Based on this TASS scheme as well as its correctness and security, we directly obtain a correct and secure MFHE scheme as a natural result of Lemma 1. In the interest of space, we put the detailed description of this MFHE scheme as well as the theorems on its properties in the full version [14]. Next, we mainly demonstrate that the complexity of $\mathtt{TASS}.\Pi_{\mathtt{ApproxRec}}$ is reduced with the help of our Lattice-Exp-ElGamal. Particularly, we list the computation and communication complexities of $\mathtt{TASS}.\Pi_{\mathtt{ApproxRec}}$, along with the comparison with existing ApproxSS schemes, in Table 2. The detailed analysis behind Table 2 is put in the full version [14]. Since existing work [15] has shown that the complexity of $\mathtt{MFHE}.\Pi_{\mathtt{Dec}}$ is the same as $\mathtt{TASS}.\Pi_{\mathtt{ApproxRec}}$, according to Table 2, we can claim that the MFHE from TASS is the first concretely-efficient one that achieves quasi-linear computation complexity and constant communication complexity, under arbitrary threshold and dynamic participants.

Table 2. Comparison between ThLHE-based ApproxSS (from Lattice-Exp-ElGamal) and existing ApproxSS schemes in terms of $\Pi_{\mathtt{ApproxRec}}$ protocol's complexity.

	$\{0,1\}$-ApproxSS [7]	Shamir-based ApproxSS			ThLHE-based ApproxSS
		Scheme 1 [7]	Scheme 2 [52]	Scheme 3 [15]	
Computation Complexity	$\mathcal{O}(N^{5.2})$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N \log N)$
Communication Complexity	$\mathcal{O}(N^{4.2})$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(1)$
Round Number	1	1	2	2	2

Acknowledgements. We sincerely thank all anonymous reviewers for their insightful reviews and constructive comments, which are of great help for improving this paper. Xinyi Huang is supported in part by the National Natural Science Foundation of China (No. 62425205). Rongmao Chen is supported in part by the National Natural Science Foundation of China (No. 62122092). Chao Lin is supported in part by National Natural Science Foundation of China (No. U21A20466 and No. 62372108). Songze Li is supported in part by the Fundamental Research Funds for the Central Universities (Grant No. 2242025K30025).

References

1. Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., Lokam, S., Micciancio, D., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Homomorphic encryption standard. Cryptology ePrint Archive, Paper 2019/939 (2019), `https://eprint.iacr.org/2019/939`
2. Applebaum, B., Nir, O., Pinkas, B.: How to recover a secret with $o(n)$ additions. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 236–262. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_8
3. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Berlin, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_29
4. Bell, J., Gascón, A., Lepoint, T., Li, B., Meiklejohn, S., Raykova, M., Yun, C.: ACORN: Input validation for secure aggregation. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 4805–4822. USENIX Association (Aug 2023)
5. Benaloh, J.C., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 27–35. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34799-2_3
6. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1175–1191. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133982
7. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 565–596. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_19
8. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19571-6_16
9. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 387–416. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_14
10. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277 (2011), `https://eprint.iacr.org/2011/277`
11. Braun, L., Damgård, I., Orlandi, C.: Secure multiparty computation from threshold encryption based on class groups. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 613–645. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_20
12. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Cham (Apr 2015). https://doi.org/10.1007/978-3-319-16715-2_26

13. Castagnos, G., Laguillaumie, F., Tucker, I.: Threshold linearly homomorphic encryption on $\mathbf{Z}/2^k\mathbf{Z}$. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 99–129. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_4
14. Chang, Y., Chen, R., Lin, C., Li, S., Huang, X.: Resolving the efficiency-utility dilemma of threshold linearly homomorphic encryption via message-space adapter. Cryptology ePrint Archive, Report 2025/*** (2025), https://eprint.iacr.org/2025/***
15. Chang, Y., Li, S.: Arbitrary-threshold fully homomorphic encryption with lower complexity. Cryptology ePrint Archive, Report 2025/084 (2025), https://eprint.iacr.org/2025/084
16. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70694-8_15
17. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (Jan 2020). https://doi.org/10.1007/s00145-019-09319-x
18. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Berlin, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44987-6_18
19. Cramer, R., Fehr, S.: Optimal black-box secret sharing over arbitrary Abelian groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 272–287. Springer, Berlin, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_18
20. Cramer, R., Fehr, S., Stam, M.: Black-box secret sharing from primitive sets in algebraic number fields. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 344–360. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_21
21. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 103–118. Springer, Berlin, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_9
22. Cramer, R., Xing, C.: Blackbox secret sharing revisited: A coding-theoretic approach with application to expansionless near-threshold schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 499–528. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_18
23. Dahl, M.: Efficient sharing with the fast fourier transform. https://mortendahl.dk/2017/06/24/secret-sharing-part2/ (2017), accessed: 2025–02-03
24. Dahl, M., Demmler, D., El Kazdadi, S., Meyre, A., Orfila, J.B., Rotaru, D., Smart, N.P., Tap, S., Walter, M.: Noah's ark: Efficient threshold-FHE using noise flooding. Cryptology ePrint Archive, Report 2023/815 (2023), https://eprint.iacr.org/2023/815
25. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Berlin, Heidelberg (Feb 2001). https://doi.org/10.1007/3-540-44586-2_9
26. Damgård, I., Thorbek, R.: Linear integer secret sharing and distributed exponentiation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 75–90. Springer, Berlin, Heidelberg (Apr 2006). https://doi.org/10.1007/11745853_6

27. Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of paillier's public-key system with applications to electronic voting. International Journal of Information Security **9**(6), 371–385 (Dec 2010)
28. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28
29. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_24
30. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4), 469–472 (1985). https://doi.org/10.1109/TIT.1985.1057074
31. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), `https://eprint.iacr.org/2012/144`
32. Fouque, P.A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Berlin, Heidelberg (Feb 2001). https://doi.org/10.1007/3-540-45472-1_7
33. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal resilience proactive public-key cryptosystems. In: 38th FOCS. pp. 384–393. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646127
34. Friedman, O., Marmor, A., Mutzari, D., Scaly, Y.C., Spiizer, Y., Yanai, A.: Tiresias: Large scale, uc-secure threshold paillier. In: ASIACRYPT. Springer-Verlag (2024)
35. Genise, N., Micciancio, D., Polyakov, Y.: Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 655–684. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17656-3_23
36. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press (May / Jun 2009). https://doi.org/10.1145/1536414.1536440
37. Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J.B., Rabin, T., Yakoubov, S.: YOSO: You only speak once - secure MPC with stateless ephemeral roles. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 64–93. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_3
38. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Berlin, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_32
39. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
40. Katz, J., Yung, M.: Threshold cryptosystems based on factoring. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer, Berlin, Heidelberg (Dec 2002). https://doi.org/10.1007/3-540-36178-2_12
41. Küsters, R., Liedtke, J., Müller, J., Rausch, D., Vogt, A.: Ordinos: A verifiable tally-hiding e-voting system. In: Proceedings of IEEE EuroS&P. pp. 216–235 (2020)
42. Mouchet, C., Bertrand, E., Hubaux, J.P.: An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. Journal of Cryptology **36**(2) (Mar 2023). https://doi.org/10.1007/s00145-023-09452-8

43. Mouchet, C., Chatel, S., Pyrgelis, A., Troncoso, C.: Helium: Scalable mpc among lightweight participants and under churn. In: Proceedings of ACM CCS. p. 3038–3052. ACM, New York, NY, USA (2024). https://doi.org/10.1145/3658644.3670346
44. Mouchet, C., Troncoso-Pastoriza, J., Hubaux, J.P.: Multiparty homomorphic encryption: From theory to practice. Cryptology ePrint Archive, Report 2020/304 (2020), https://eprint.iacr.org/2020/304
45. Mouchet, C., Troncoso-Pastoriza, J.R., Bossuat, J.P., Hubaux, J.P.: Multiparty homomorphic encryption from ring-learning-with-errors. PoPETs **2021**(4), 291–311 (Oct 2021). https://doi.org/10.2478/popets-2021-0071
46. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: Gong, L., Reiter, M.K. (eds.) ACM CCS 98. pp. 59–66. ACM Press (Nov 1998). https://doi.org/10.1145/288090.288106
47. Nakamoto, S.: A peer-to-peer electronic cash system (2009), https://bitcoin.org/bitcoin.pdf
48. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 308–318. Springer, Berlin, Heidelberg (May / Jun 1998). https://doi.org/10.1007/BFb0054135
49. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_16
50. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Berlin, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_15
51. Singh, A., Patranabis, S., Mukhopadhyay, D.: Threshold OPRF from threshold additive HE. Cryptology ePrint Archive, Report 2024/1032 (2024), https://eprint.iacr.org/2024/1032
52. Tian, H., Wen, Y., Zhang, F., Shao, Y., Li, B.: A distributed threshold additive homomorphic encryption for federated learning with dropout resiliency based on lattice. In: Proceedings of International Symposium on Cyberspace Safety and Security. p. 277–292. Springer-Verlag, Berlin, Heidelberg (2022)
53. Vempala, S.: Fast fourier transform. https://sites.cc.gatech.edu/classes/AY2010/cs6505_spring/lectures/FFT.pdf (2010), accessed: 2025–02-03
54. Wong, H.W.H., Ma, J.P.K., Chow, S.S.M.: Secure multiparty computation of threshold signatures made more efficient. In: NDSS 2024. The Internet Society (Feb / Mar 2024)