Arbitrary Univariate Function Evaluation and Re-Encryption Protocols over Lifted-ElGamal Type Ciphertexts

Koji Nuida[1][2], Satsuya Ohata[2], Shigeo Mitsunari[3], Nuttapong Attrapadung[2]

[1] Graduate School of Information Science and Technology, The University of Tokyo, Japan
nuida@mist.i.u-tokyo.ac.jp
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan
[3] Cybozu Labs, Inc., Japan

Abstract

Homomorphic encryption (HE) is one of the main tools in secure multiparty computation (MPC), and the (elliptic-curve) lifted-ElGamal cryptosystem is certainly the most efficient among the existing HE schemes. However, the combination of MPC with this most efficient HE has rarely appeared in the literature. This is mainly because the major known techniques for (additively) HE-based MPC are not available for this scheme due to its typical restriction that only a plaintext in a small range can be efficiently decrypted.

In this paper, we resolve this problem. By our technique, a Server having a lifted-ElGamal ciphertext $[[m]]$ with unknown small plaintext m can obtain a ciphertext $[[\varphi(m)]]$ for an *arbitrary* function φ by just one-round communication with a semi-honest Client (and also two-rounds with a malicious Client) having a decryption key, where m is kept secret for both parties. This property enlarges much the variations of MPC based on the most efficient lifted-ElGamal cryptosystem. As an application, we implemented MPC for exact edit distance between two encrypted strings; our experiment for strings of length 1024 shows that the protocol takes only 45 seconds in LAN environments and about 3 minutes even in WAN environments. Moreover, our technique is also available with other "lifted-ElGamal type" HE schemes and admits different keys/schemes for the original and the resulting ciphertexts. For example, we can securely convert a level-2 (i.e., after multiplication) ciphertext for some two-level HE schemes into a level-1 (i.e., before multiplication) ciphertext, and securely apply arbitrary functions $\varphi(m)$ to encrypted plaintexts for some attribute-based HE schemes. This is the first result (even by using communication) on realizing these two functionalities.

1 Introduction

Secure multiparty computation (*MPC*) is a cryptographic technique that enables two or more parties to jointly evaluate a function for their local inputs while concealing the inputs from each other. Due to increasing social demands for privacy protection in information technology such as big-data analyses and personalized services, MPC has been a major research topic in cryptology itself as well as applications to other areas. Among various solutions to MPC depending on practical situations, one of the major settings is *client-server secure computation* using *homomorphic encryption* (HE), where a Client sends his/her input (e.g., a search query) to a Server in securely encrypted form and Server (who may also have his/her own input; e.g., a database) performs the necessary computations (e.g., determining the search result) in encrypted form, possibly with some communications to Client (where, of course, Server's secret input should be kept secret against Client). This paper focuses on this HE-based client-server model.

For the underlying HE schemes in this scenario, *fully homomorphic encryption* (*FHE*) [10] is an obvious option, which in principle enables Server to perform arbitrary operations over encrypted data without communication. However, the existing FHE schemes, though intensively improved in efficiency, are still not very efficient so far. On the other hand, *additively homomorphic encryption* (*AHE*) [9, 21] is the most efficient class among the practical HE schemes, which supports only addition (and scalar multiplication) over encrypted integers. Despite the restricted functionality, several techniques to achieve MPC using AHE have been developed as explained below. There is also an intermediate class, such as *two-level homomorphic encryption* (*2LHE*) [3, 8] which additionally supports a single multiplication over ciphertexts and is still relatively efficient.

For the known techniques for AHE-based MPC mentioned above, the fundamental underlying idea is "homomorphic masking" for encrypted values, which is outlined as follows (where $[[m]]$ denotes a ciphertext with plaintext m):

1. Given a ciphertext $[[x]]$, Server homomorphically generates $[[x + r]]$ with a random secret value r, and sends $[[x + r]]$ to Client.
2. Client decrypts it and obtains $x + r$, then generates some ciphertext(s) c depending on $x + r$ and sends it to Server. Here the value x is information-theoretically concealed for Client by virtue of the random secret value r.
3. Server generates the result by using $[[x + r]]$, c, and the knowledge of r.

This idea has been widely used in AHE-based MPC to realize various kinds of functionalities, e.g., arithmetic multiplication [13, 14] and division [28]; less-than comparison [16, 29]; and bit-decomposition [22, 23, 24].

However, this technique has the following serious drawback: Among the known AHE schemes, the "lifted" ElGamal cryptosystem [9] (where the plaintext is placed at the exponent of a group element) over elliptic curves is certainly the most efficient so far, but the technique above does not work when using the lifted ElGamal cryptosystem as the underlying AHE. This is because, in the cryptosystem we can decrypt a ciphertext only when its plaintext is sufficiently small, while the masking value r above must be chosen from the whole plaintext space for concealing x, therefore Client cannot decrypt $[[x + r]]$ in general (even if the original x is small). As a result, the most efficient lifted ElGamal cryptosystem has rarely been applied to MPC in the literature (except some results mentioned later), and we had to rely on other, less efficient AHE schemes such as Paillier cryptosystem [21]. We also note that, the known practical 2LHE schemes [3, 8] also have the same restriction on the decryptable plaintexts, therefore such schemes could not be used in MPC beyond the functionalities natively supported by 2LHE. To overcome this issue, we must establish a new methodology in MPC that is also applicable to those "lifted-ElGamal type" schemes.

1.1 Our Contributions

In this paper, we propose a new technique for AHE-based MPC that resolve the serious issue described above. More precisely, our technique achieves the following functionality even when the underlying scheme is of lifted-ElGamal type: Given a ciphertext $[[m]]$ with unknown plaintext m and an *arbitrary* univariate function $\varphi(x)$, Server can securely obtain a ciphertext $[[\varphi(m)]]$ with *one* round-trip (i.e., Server $\to$ Client $\to$ Server) communication to semi-honest Client having the decryption key. This is also extendible to the case of malicious Client by adding one more communication round as explained later. A remarkable advantage is that *the execution cost of our protocol is almost independent of the complexity of the function* φ (precisely, the communication cost and the number of required homomorphic operations are both independent of φ). We also note that our technique can be also extended to two-input functions $\varphi(x, y)$ (e.g., max/min and multiplication) under certain condition; see Section 6 for details. On the other hand, the main drawback of our technique is that the execution cost is proportional to the number of possible plaintexts m that can appear in the input ciphertext $[[m]]$ for Server (in short, we can handle ciphertexts with sufficiently small plaintexts only). Here we emphasize however that, despite this drawback, our technique is still applicable to some practical problems in MPC. For example, based on our technique, we implemented MPC for *exact edit distance* of encrypted strings, and our experiment shows that it takes only about 3 min. for strings of length 1024 even in WAN environments. See Section 7.2 for details of the experiments.

To explain the idea of our proposed technique, we revisit some "exceptional" previous results (e.g., [25, 26, 27]) on MPC based on the lifted-ElGamal cryptosystem (which are indeed very rare, as mentioned in the last paragraph before this subsection). Their first idea is that, when Client has an input x in a small range, say $[0, K-1]$, Client generates encrypted "unary vector" expressing x, that is, a list of ciphertexts $\mathsf{EU}(x) = ([[a_0]], \ldots, [[a_{K-1}]])$ where $a_i = 1$ if $x = i$ and $a_i = 0$ if $x \neq i$. If Client sends $\mathsf{EU}(x)$ to Server, then Server can compute $[[\varphi(x)]]$ by taking the inner product of the plain vector $(\varphi(0), \ldots, \varphi(K-1))$ and the encrypted vector $\mathsf{EU}(x)$ via the AHE functionality. Moreover, in their results, a *recursive* function evaluation is required, therefore Server and Client have to convert (without leaking the value $\varphi(x)$ to Client) the intermediate ciphertext $[[\varphi(x)]]$ into $\mathsf{EU}(\varphi(x))$ in order to compute the next function. They solved this problem by using the following trick: Server generates in the previous step a ciphertext $[[\varphi(x) + r \bmod K]]$ instead of $[[\varphi(x)]]$ where r is a random secret value. If this is sent to and decrypted by Client, then Client obtains $\varphi(x) + r \bmod K$, but the value of $\varphi(x)$ is still concealed by virtue of r. Moreover, when Client generates $\mathsf{EU}(\varphi(x) + r \bmod K)$, Server can obtain $\mathsf{EU}(\varphi(x))$ by just cyclically rotating the components in $\mathsf{EU}(\varphi(x) + r \bmod K)$ by r.

Based on the observation above, our problem can be restated as a problem of securely converting $[[m]]$ into $\mathsf{EU}(m)$ *when the ciphertext* $[[m]]$ *is given from the beginning.* This means that, now there is no "previous step" to modify the construction of $[[m]]$, hence the trick mentioned above is no longer available. To solve the problem, we introduce another trick as follows. Suppose that the plaintext m is unknown but is guaranteed to lie in a given range $[0, K-1]$. The key fact is that, for each $a \in [0, K-1]$, if Server generates $[[r_a \cdot (m-a)]]$ with random r_a (which is available via the AHE functionality), then the value of $r_a \cdot (m-a)$ becomes 0 if $m = a$, and a random value if $m \neq a$. By sending these ciphertexts to Client in random order, Client receives one $[[0]]$ and $K-1$ $[[\text{random}]]$'s in random order, which does not leak information on m. Now for each position of the ciphertext, if Client has received $[[0]]$ (respectively, $[[\text{random}]]$) then Client sends $[[1]]$ (respectively, $[[0]]$) to Server. This yields a permutation of the vector $\mathsf{EU}(m)$, and now Server can recover $\mathsf{EU}(m)$ from its permuted version, as desired. We note that the vector $\mathsf{EU}(m)$ obtained by Server can be reused for computing two or more functions simultaneously. See Section 3 for more details of our proposed protocol (with semi-honest Client). Here we note further that, one may feel a relation between our result and some other primitives such as the oblivious transfer (OT) and the private information retrieval (PIR). Although there might be some similarity to the techniques used in OT or PIR, we emphasize that the problem setting is totally different from those primitives, as now not only Server but also even Client do not know which is "the correct value" (i.e., m) to be retrieved by Client.

Although our explanation above focused on our functionality of computing a function over encrypted data, our proposed protocol also has the following remarkable advantage: *the key pairs, and even the underlying AHE schemes themselves, for the original ciphertext* $[[m]]$ *and for the resulting ciphertext* $[[\varphi(m)]]$ *may be different.* That is, our protocol enables to securely convert a ciphertext in some key/scheme into a ciphertext in another key/scheme. We investigate in detail the following two applications of this property:

Level reduction for 2LHE ciphertexts. For some 2LHE schemes [3, 8] having a structure similar to the lifted-ElGamal cryptosystem, our protocol enables to securely convert a level-2 ciphertext $[[m]]$ (i.e., after homomorphic multiplication) into a level-1 ciphertext (i.e., before homomorphic multiplication). This functionality is analogous to the bootstrapping procedure [10] for FHE schemes. Of course, such a functionality seems never achievable by a stand-alone use of just 2LHE schemes. Our protocol realizes this functionality by using one-round communication, and this is the first result (even when communication is allowed) on achieving this functionality. We implemented this protocol and performed experiments, which show that the conversion in the case $m \in [0, 2^8 - 1]$ takes only 0.08 sec. and 0.23 sec. in LAN and WAN environments, respectively. See Section 7.2 for details.

Computing over attribute-based encryption. In attribute-based encryption (ABE) for predicate P, a key associated to attribute x can be used to decrypt a ciphertext associated to attribute y if $P(x, y) = 1$. ABE is useful for fine-grained access control over encrypted data [12]. Adding homomorphic capabilities to ABE is a long-standing problem [6]. The most recent progress by Brakerski et al. [6] allows to construct the so-called targeted homomorphic ABE. However, their constructions perform in the same level of state-of-the-art (multi-key) FHE schemes [15, 7, 18] at best, which themselves are not very practical yet. In contrast, here we focus on more efficient pairing-based ABE. We observe that most of the existing pairing-based ABE schemes have the form of ElGamal-type ciphertexts; as a result, our proposed technique for function evaluation over ciphertexts is easily extended to those ABE (as long as the attributes for keys and ciphertexts are properly controlled). See Section 8 for details.

We also explain an outline of the extension of our proposed protocol from semi-honest Client to malicious Client. To force malicious Client to reply ciphertexts with correct plaintexts, a known technique used in the literature (e.g., [13]) is to let Server generate a "flag ciphertext" from Client's reply in a way that it has plaintext 0 if Client's choice of plaintexts is honest, while it has a non-zero plaintext with high probability if Client is dishonest.

To apply this framework in our situation, as the first step, we replace the encrypted unary vector sent from Client with an encrypted "unary matrix", where the plaintexts in the "true" column (corresponding to the candidate value $a \in [0, K-1]$ with $a = m$) have to be a small vector $\vec{\alpha}$ (depending on the column) chosen by Server (instead of the value 1 used in the semi-honest case) and the plaintexts in the other "false" columns have to be zero vectors $\vec{0}$. Server wants to verify that each column is either $\vec{\alpha}$ or $\vec{0}$ without knowing which is actually chosen (i.e., which is the true column). For the purpose, Server randomly chooses a vector $\vec{w}$ orthogonal to $\vec{\alpha}$, and homomorphically computes the inner product of $\vec{w}$ and the encrypted column. If Client is honest, then the resulting plaintext is 0 as $\vec{w} \cdot \vec{\alpha} = \vec{w} \cdot \vec{0} = 0$. On the other hand, for false columns, the vector $\vec{\alpha}$ is concealed from Client by the random masking, therefore the probability that Client can choose a plaintext vector different from the honest choice $\vec{0}$ and orthogonal to $\vec{w}$ is significantly small.

However, for the true column, as the honest choice is now $\vec{\alpha}$, Client can easily deviate from it while keeping the orthogonality to $\vec{w}$ by just choosing $\vec{0}$. In order to prevent this, we also introduce dummy columns containing ciphertexts, say with plaintexts $\vec{\beta}$, in a way that the vectors $\vec{\beta}$ are indistinguishable from the $\vec{\alpha}$ at the true column, and whenever Client deviates from choosing the $\vec{\beta}$ at some dummy column, the flag ciphertext will have non-zero plaintext with high probability. Now malicious Client has to guess the correct ciphertexts among the indistinguishable ciphertexts in the true and the dummy columns, which will be difficult if the length of the vector and the number of dummy columns are sufficiently large. See Section 4 for further details. We also note that, when Client is malicious and the underlying scheme involves invalid ciphertexts that are indistinguishable from valid ciphertexts, such as the 2LHE schemes in [3, 8], we have to care about the possibility that Client replies an invalid ciphertext in the protocol. For this case, we can modify (by utilizing the 2LHE functionality) the construction above in a way that the plaintext of the flag ciphertext will be non-zero with high probability as well when Client does not reply valid ciphertexts. See Section 5 for details.

2 Preliminaries

In this paper, $y \leftarrow \mathcal{A}(x)$ (or $\mathcal{A}(x; r)$, when specifying the randomness r) means that an algorithm $\mathcal{A}$ with input x outputs y, and $a \overset{R}{\leftarrow} X$ means that an element a is chosen from a (finite) set X uniformly at random. "PPT" means "probabilistic polynomial-time". The security parameter is usually denoted by λ. We say that a function $\varepsilon(\lambda)$ is *negligible* if, for any $k > 0$, there exists a λ_0 satisfying that $|\varepsilon(\lambda)| < \lambda^{-k}$ for any $\lambda > \lambda_0$. The statistical distance between two random variables X and Y is defined by $\Delta(X, Y) \overset{\text{def}}{=} \frac{1}{2} \sum_a |\Pr[X = a] - \Pr[Y = a]|$. For $a, b \in \mathbb{Z}$, we write $[a, b] \overset{\text{def}}{=} \{a, a+1, \dots, b-1, b\}$. We often write a ciphertext for plaintext m as $[[m]]$ and write the plaintext for a ciphertext c as $\mathsf{Pt}[c]$.

2.1 Homomorphic Encryption with Restricted Plaintext Range

In this paper, we deal with the following class of *additively homomorphic encryption* (*AHE*) *schemes* including the lifted-ElGamal cryptosystem [9].

Definition 1. An *AHE scheme* Π consists of PPT algorithms Gen, Enc, Dec, Rnd and polynomial-time computable operators $\boxplus$, $\boxminus$ satisfying the following:

- The key generation algorithm Gen outputs a public key pk and a secret key sk; $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Here the plaintext space $\mathcal{M}$ is an additive group and contains a subset $\mathcal{M}_{\text{eff}} \subseteq \mathcal{M}$ of *effective plaintexts*; and the ciphertext space $\mathcal{C}$ contains disjoint subsets $\mathcal{C}_m$ for $m \in \mathcal{M}$. We define the set of *valid ciphertexts* by $\mathcal{C}_{\text{val}} \overset{\text{def}}{=} \bigcup_{m \in \mathcal{M}} \mathcal{C}_m \subseteq \mathcal{C}$. Symbols pk and sk that are obvious from the context may be omitted.

  Note: Only the ciphertexts $c \in \mathcal{C}_m$ with $m \in \mathcal{M}_{\text{eff}}$ are guaranteed to be efficiently decrypted (see below).

- Given $m \in \mathcal{M}$, the encryption algorithm Enc outputs an element $c \in \mathcal{C}$; $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.

  Randomness Condition: We require that $\mathsf{Enc}_{\mathsf{pk}}(m; r)$ for uniform randomness r is uniformly random over $\mathcal{C}_m$.

- Given sk and $c \in \mathcal{C}$, the decryption algorithm Dec outputs either an element of $\mathcal{M}$ or the failure symbol $\perp$.

  Correctness Condition: We require that, for $c \in \mathcal{C}_m$, it holds with probability one that $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ when $m \in \mathcal{M}_{\text{eff}}$ and $\perp \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$ when $m \notin \mathcal{M}_{\text{eff}}$.

- ***(Homomorphic Addition)*** Given $c_1 \in \mathcal{C}_{m_1}$ and $c_2 \in \mathcal{C}_{m_2}$, the operation $c_1 \boxplus_{\mathsf{pk}} c_2$ yields an element of $\mathcal{C}_{m_1+m_2}$.

- ***(Homomorphic Negation)*** Given $c \in \mathcal{C}_m$, the operation $\boxminus_{\mathsf{pk}} c$ yields an element of $\mathcal{C}_{-m}$. We also write $c_1 \boxminus_{\mathsf{pk}} c_2 \overset{\text{def}}{=} c_1 \boxplus_{\mathsf{pk}} (\boxminus_{\mathsf{pk}} c_2)$.

- ***(Homomorphic Rerandomization)*** Given $c \in \mathcal{C}_m$, the algorithm Rnd outputs a uniformly random element $c' \in \mathcal{C}_m$; $c' \leftarrow \mathsf{Rnd}_{\mathsf{pk}}(c)$.

When $c \in \mathcal{C}_m$ ($m \in \mathcal{M}$) and k is a non-negative integer of polynomially bounded bit length, we can generate an element of $\mathcal{C}_{k \cdot m}$ by combining the operator $\boxplus_{\mathsf{pk}}$ with the standard binary method; we denote the resulting operation by $k \boxdot_{\mathsf{pk}} c$. When $k < 0$, we write $k \boxdot_{\mathsf{pk}} c \overset{\text{def}}{=} |k| \boxdot_{\mathsf{pk}} (\boxminus_{\mathsf{pk}} c) \in \mathcal{C}_{k \cdot m}$. On the other hand, for $c \in \mathcal{C}$ and $m' \in \mathcal{M}$, we may simply write $m' \boxplus c$ instead of $\mathsf{Enc}(m'; 0) \boxplus c$. Similar notations $c \boxplus m'$, $m' \boxminus c$, and $c \boxminus m'$ are also used.

Remark 2. Here we do *not* claim that a party having a secret key sk can obtain no information on the plaintext for a given ciphertext c with $c \notin \bigcup_{m \in \mathcal{M}_{\text{eff}}} \mathcal{C}_m$. We emphasize that our proposed protocols are secure even in this situation.

2.2 Two-Level Homomorphic Encryption

In this paper, by a *two-level homomorphic encryption* (*2LHE*) *scheme* we mean an AHE scheme in which a single multiplication between encrypted plaintexts is also possible. For some existing schemes [3, 8], the set of *level*-1 *ciphertexts* (i.e., those before multiplication) can be viewed as an AHE scheme, and similarly for *level*-2 *ciphertexts* (i.e., those after multiplication). We formalize this as follows.

Definition 3. A *2LHE scheme* Π consists of two AHE schemes $\Pi^{\langle 1 \rangle}, \Pi^{\langle 2 \rangle}$ with the same plaintext space $\mathcal{M}$ which is a ring, together with a polynomial-time computable operator $\boxtimes^{\langle 1 \rangle}$. We call $\Pi^{\langle i \rangle}$ for $i \in \{1, 2\}$ the *level-i part* of Π; below we indicate the objects associated to the part $\Pi^{\langle i \rangle}$ by superscript "$\langle i \rangle$".

More precisely, for a key pair $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ of Π, pk involves public keys $\mathsf{pk}^{\langle 1 \rangle}$ for $\Pi^{\langle 1 \rangle}$ and $\mathsf{pk}^{\langle 2 \rangle}$ for $\Pi^{\langle 2 \rangle}$, where the two associated plaintext spaces $\mathcal{M}^{\langle 1 \rangle}$ and $\mathcal{M}^{\langle 2 \rangle}$ are equal, denoted by $\mathcal{M}$; and sk involves the corresponding secret keys $\mathsf{sk}^{\langle 1 \rangle}$ for $\Pi^{\langle 1 \rangle}$ and $\mathsf{sk}^{\langle 2 \rangle}$ for $\Pi^{\langle 2 \rangle}$. (On the other hand, the subsets of effective plaintexts $\mathcal{M}_{\text{eff}}^{\langle 1 \rangle}$ and $\mathcal{M}_{\text{eff}}^{\langle 2 \rangle}$ may be different.) Moreover:

- ***(Homomorphic Multiplication)*** Given $c_1 \in \mathcal{C}_{m_1}^{\langle 1 \rangle}$ and $c_2 \in \mathcal{C}_{m_2}^{\langle 1 \rangle}$, the operation $c_1 \boxtimes_{\mathsf{pk}}^{\langle 1 \rangle} c_2$ yields an element of $\mathcal{C}_{m_1 \cdot m_2}^{\langle 2 \rangle}$.

We also use a notation $m' \boxtimes^{\langle 1 \rangle} c$ with $c \in \mathcal{C}^{\langle 1 \rangle}$ and $m' \in \mathcal{M}$ as an abbreviation of $\mathsf{Enc}^{\langle 1 \rangle}(m'; 0) \boxtimes^{\langle 1 \rangle} c$, and similarly for $c \boxtimes^{\langle 1 \rangle} m'$. In particular, the operation $1 \boxtimes^{\langle 1 \rangle} c$ yields a conversion of a level-1 ciphertext c into a level-2 ciphertext with the same plaintext.

2.3 Two-Party Computation in the Malicious Model

In the paper, we deal with a two-party protocol between *Server* and *Client*. For the security definition in the malicious model, we basically follow Section 7.2.3 of [11] with slight notation changes, and perform the following simplification owing to the specific situation in this paper:

- Only Server obtains an output (Client has no output).
- We only consider one-sided security against possibly malicious Client; see the first paragraph of Section 3.4 for the reason.
- We do not consider any modification of the input by Client. This is because, in our proposed protocols Client is not supposed to send the input to Server, therefore the input modification is just a part of Client's local behavior.

Let $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ denote the functionality for Server's output to be computed, where pp denotes a public parameter known to both parties and x_{S} and x_{C} denote local inputs for Server and Client, respectively.

For a real protocol execution, let Client's *transcript* (with PPT algorithm $\mathcal{A}$) be the list of messages sent from Server to Client during the protocol; and let Client's *view* $\mathsf{view}_{\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}})$ consist of the transcript together with pp, x_{C}, and Client's internal randomness r_{C}, where r_{S} is Server's internal randomness. Let $\mathcal{A}_{\mathsf{out}}$ denote Client's algorithm to make some guess about Server's secret after the protocol. On the other hand, let $\mathsf{out}_{\mathsf{S},\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}})$ denote Server's output by the protocol, which is defined to be $\perp$ when the protocol is aborted. Then the output pair for the two parties is

$$\mathsf{out}_{\mathsf{real},\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}}) \overset{\mathrm{def}}{=} (\mathsf{out}_{\mathsf{S},\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}}), \mathcal{A}_{\mathsf{out}}(\mathsf{view}_{\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}}))) \enspace .$$

On the other hand, an ideal protocol execution with Client's PPT algorithm $\mathcal{B}$ is defined in the following manner. First, the public parameters and local inputs for two parties are sent to the trusted party (TP). Secondly, TP asks Client whether aborting the protocol or not. Then:

- If Client's reply $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}})$ is to abort (denoted by `abort`), then TP tells Server that the protocol is aborted; now Server outputs $\perp$, while Client outputs an object given by $\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}})$. Hence the output pair is
$$\mathsf{out}_{\mathsf{ideal},\mathcal{B}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{C}}) \overset{\mathrm{def}}{=} (\perp, \mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}})) \enspace .$$
- Otherwise (denoted by `proceed`), TP sends $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ to Server; now Server outputs it, while Client outputs an object given by $\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}})$. Hence the output pair is
$$\mathsf{out}_{\mathsf{ideal},\mathcal{B}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{C}}) \overset{\mathrm{def}}{=} (F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}), \mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}})) \enspace .$$

Now the security against malicious Client is defined as follows. The case of semi-honest Client is its special case where Client does not deviate from the protocol in the real execution nor abort the protocol in the ideal execution.

Definition 4. In the setting above, we say that a protocol is *secure* against malicious Client with statistical distance at most ε if, for any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$ satisfying that, for any pp, x_{S}, and x_{C}, the probability distributions for $\mathsf{out}_{\mathsf{real},\mathcal{A}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{S}}, r_{\mathsf{C}})$ with uniformly random $(r_{\mathsf{S}}, r_{\mathsf{C}})$ and for $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}}; r_{\mathsf{C}})$ with uniformly random r_{C} (and uniform internal randomness for the functionality F) have statistical distance at most ε.

3 Our Protocol with Semi-Honest Client

3.1 Setting

Let Π and $\Pi^\dagger$ be AHE schemes with key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$, respectively. We will indicate objects associated to the scheme $\Pi^\dagger$ by superscript '$\dagger$'. We suppose that the plaintext spaces are $\mathcal{M} = \mathbb{F}_p$ and $\mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$ for publicly known primes p and $p^\dagger$, respectively. Let $\mathcal{M}_{\mathrm{eff}} \subseteq \mathcal{M}$ be the subset of effective plaintexts for Π (see Definition 1). We suppose that $0 \in \mathcal{M}_{\mathrm{eff}}$.

In the protocol, two parties' inputs are specified as follows:

Server's local input: Server has a ciphertext c for Π associated to the key pk, where the plaintext $\mathsf{Pt}[c]$ is supposed to be in a given subset $S \subseteq \mathcal{M} = \mathbb{F}_p$. Server also has polynomially many functions $\varphi_h \colon S \to \mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$, $h \in [1, L]$.

Client's local input: Client has the secret key sk for Π.

Common information: Both parties know pk, $\mathsf{pk}^\dagger$, and the cardinality $|S|$ of the subset S. Note that $|S|$ must be polynomially bounded.

Our protocol has one-sided output for Server, i.e., Client outputs nothing. The functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ for Server's output is defined as follows, where pp, x_{S}, and x_{C} denote the common information, Server's local input, and Client's local input, respectively:

- Parsing pp, x_{S}, and x_{C} as above, $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ is a list of uniformly random ciphertexts $c_h^\dagger$ $(h \in [1, L])$ for plaintexts $\varphi_h(\mathsf{Pt}[c])$ in $\Pi^\dagger$ associated to $\mathsf{pk}^\dagger$.

We emphasize that the schemes Π and $\Pi^\dagger$, as well as the key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$, may be different. Therefore, our protocol can be used for converting a ciphertext in one scheme (e.g., a level-2 ciphertext in a 2LHE scheme) into a ciphertext in another scheme (e.g., a level-1 ciphertext in the 2LHE scheme).

3.2 Description of the Protocol

1. [Server $\to$ Client] Server executes the following:
   (a) For each $j \in S$, choose $\gamma_j \xleftarrow{R} \mathbb{F}_p$ and set $c_j \leftarrow \mathsf{Rnd}\big((\gamma_j \boxdot c) \boxminus (\gamma_j \cdot j)\big)$.
   (b) Send, to Client, all the ciphertexts c_j in a uniformly random order.
2. [Client $\to$ Server] Let $\overline{c}_1, \dots, \overline{c}_{|S|}$ denote the ciphertexts for Π sent from Server at the previous step. Client executes the following:
   (a) For each $\overline{i} \in [1, |S|]$, compute $\overline{m}_{\overline{i}} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\overline{c}_{\overline{i}})$. If the number of $\overline{i} \in [1, |S|]$ satisfying $\overline{m}_{\overline{i}} = 0$ is not equal to 1, then abort the protocol.
   (b) For each $\overline{i} \in [1, |S|]$, set
   $$\overline{\overline{c}}_{\overline{i}} \leftarrow \begin{cases} \mathsf{Enc}^\dagger(1) & \text{if } \overline{m}_{\overline{i}} = 0 \ , \\ \mathsf{Enc}^\dagger(0) & \text{if } \overline{m}_{\overline{i}} \neq 0 \ . \end{cases}$$
   (c) Send $\overline{\overline{c}}_1, \dots, \overline{\overline{c}}_{|S|}$ to Server in this order.
3. [Server's output] Given the $|S|$ ciphertexts sent from Client at the previous step, Server first permutes the ciphertexts in the reverse way of Step 1b, yielding ciphertexts $\widetilde{c}_j$ $(j \in S)$. Then Server executes the following:
   (a) For each $h \in [1, L]$, set $\overline{c}_h^\dagger \leftarrow \boxplus^\dagger_{j \in S}(\varphi_h(j) \boxdot^\dagger \widetilde{c}_j)$ and $c_h^\dagger \leftarrow \mathsf{Rnd}^\dagger(\overline{c}_h^\dagger)$.
   (b) Then output the $c_h^\dagger$ for all $h \in [1, L]$.

3.3 Correctness

Theorem 5. *Suppose that both Server and Client honestly execute the protocol. Then the protocol is aborted with probability at most $\varepsilon_{\mathrm{cor}} = (|S| - 1)/p$. Conditioned on that the protocol is not aborted, Server's output distribution is identical to the distribution of the functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$.*

Proof. For Step 1, we have $\mathsf{Pt}[c_j] = \gamma_j \cdot \mathsf{Pt}[c] - \gamma_j \cdot j = \gamma_j(\mathsf{Pt}[c] - j)$, which is $0 \in \mathcal{M}_{\mathrm{eff}}$ when $j = \mathsf{Pt}[c]$ and is uniformly random over $\mathcal{M} = \mathbb{F}_p$ when $j \neq \mathsf{Pt}[c]$ as now $\mathsf{Pt}[c] - j \in \mathbb{F}_p{}^{\times}$ and $\gamma_j \overset{R}{\leftarrow} \mathbb{F}_p$. By the hypothesis $\mathsf{Pt}[c] \in S$, the condition $j = \mathsf{Pt}[c]$ is satisfied for precisely one $j \in S$. Therefore, for Step 2, the number of $\bar{i}$'s with $\mathsf{Pt}[\overline{c}_{\bar{i}}] = 0$ is at least 1, and it exceeds 1 (hence Client aborts) only when some of the remaining $|S| - 1$ random plaintexts becomes 0, which occurs with probability at most $\varepsilon_{\mathrm{cor}}$. Moreover, for Step 3 (where the protocol has not been aborted), the argument above implies that $\mathsf{Pt}[\widetilde{c}_{\mathsf{Pt}[c]}] = 1$ and $\mathsf{Pt}[\widetilde{c}_j] = 0$ for $j \neq \mathsf{Pt}[c]$. Therefore, for $h \in [1, L]$, we have

$$\mathsf{Pt}[\overline{c}_h^{\dagger}] = \varphi_h(\mathsf{Pt}[c]) \cdot 1 + \sum_{j \in S \setminus \{\mathsf{Pt}[c]\}} \varphi_h(j) \cdot 0 = \varphi_h(\mathsf{Pt}[c]) \tag{1}$$

and the output $c_h^{\dagger}$ is a uniformly random ciphertext for the same plaintext by the property of $\mathsf{Rnd}^{\dagger}$. Hence Theorem 5 holds. □

3.4 Security

When Server honestly executes the protocol and the AHE schemes Π and $\Pi^{\dagger}$ are IND-CPA, it is natural to expect that the protocol does not leak Client's secret information (e.g., plaintext for c) to Server, as all messages sent to Server are encrypted and the condition to abort the protocol is independent of Client's secret. Here we do not formalize this argument due to the technical difficulty that, the public keys are now regarded as *fixed* values involved in the input for the protocol, while the security of the AHE schemes is defined with respect to *random* keys. Nevertheless, when our protocol is included as a subprotocol of an "ordinary" protocol, the CPA security of the AHE schemes would indeed imply that Client's secret is protected against Server during this subprotocol.

On the other hand, we have the following result on the security of our protocol against semi-honest Client.

Theorem 6. *There exists a PPT algorithm (simulator) $\mathcal{S}$ satisfying that, given Client's input objects (either local or common to Server) as input, the output distribution of $\mathcal{S}$ is identical to the distribution of Client's view during an honest execution of the protocol.*

Proof. The argument in the proof of Theorem 5 shows that, among the $|S|$ ciphertexts sent from Server, which have been perfectly rerandomized (by Rnd) and permuted, precisely one of them have plaintext 0, while the plaintexts for the remaining ciphertexts are independent and uniformly random over $\mathcal{M}$. This transcript can be perfectly simulated by PPT $\mathcal{S}$ by generating a single ciphertext of 0 and $|S| - 1$ ciphertexts of random plaintexts and then permuting them, while the other part of Client's view can be trivially simulated as well. Hence Theorem 6 holds. □

4 Our Protocol with Malicious Client

In this section, we extend the protocol in Section 3 to the case of malicious Client. Note that Server is still supposed to honestly follow the protocol.

First we observe that, considering a batch execution of our protocol against semi-honest Client (in Section 3) for many, say N input ciphertexts, we do not know any strategy better than just executing N protocols in parallel; hence the total complexity is proportional to N. On the other hand, for the current case of malicious Client, our argument below will show that the complexity per one ciphertext of a batch execution for N ciphertexts can be reduced when N increases. Intuitively, this is because our protocol against malicious Client uses some dummy ciphertexts, and gathering all the N collections of dummy ciphertexts yields stronger security than using the N collections of dummies individually. By this reason, below we consider such a batch protocol execution.

4.1 Setting

In the same way as Section 3, let $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$ be key pairs for AHE schemes Π and $\Pi^\dagger$ with prime-order plaintext spaces $\mathcal{M} = \mathbb{F}_p$ and $\mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$, respectively. We suppose that the set $\mathcal{M}_{\mathrm{eff}}$ of effective plaintexts for Π (see Definition 1) is a subset of $[0, p^\dagger - 1]$, hence any non-zero element of $\mathcal{M}_{\mathrm{eff}}$ (which is invertible in $\mathcal{M} = \mathbb{F}_p$) is also invertible in $\mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$. Let $\mathcal{M}^\dagger_{\mathrm{eff}}$ denote the set of effective plaintexts for $\Pi^\dagger$.

Considering a batch protocol execution with N input ciphertexts as discussed above, now two parties' inputs are specified as follows:

Server's local input: Server has N ciphertexts $c_1, \dots, c_N$ for Π associated to the key pk, where each plaintext $\mathsf{Pt}[c_i]$ is in a given subset $S_i \subseteq \mathcal{M}$, and also has polynomially many functions $\varphi_{i,h} \colon S_i \to \mathcal{M}^\dagger$ with $i \in [1, N]$, $h \in [1, L_i]$.

Client's local input: Client has the secret keys sk and $\mathsf{sk}^\dagger$ for Π and $\Pi^\dagger$.

Common information: Both parties know pk, $\mathsf{pk}^\dagger$, N, $N_S \overset{\mathrm{def}}{=} \sum_{i=1}^{N} |S_i|$, and additional parameters μ and ν. Note that all of N_S, μ, and ν must be polynomially bounded.

Similarly to Section 3.1, our protocol here also has the one-sided output functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ for Server defined as follows:

- Parsing pp, x_{S}, and x_{C} as above, $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$ is a list of uniformly random ciphertexts $c^\dagger_{i,h}$ (for $i \in [1, N]$, $h \in [1, L_i]$) for plaintexts $\varphi_{i,h}(\mathsf{Pt}[c_i])$ in $\Pi^\dagger$ associated to $\mathsf{pk}^\dagger$.

4.2 Description of the Protocol

1. [Server $\to$ Client (1)] Server executes the following:

   (a) For each $i \in [1, N]$ and each $j \in S_i$, choose vectors
   $$\vec{\alpha}_{i,j} = (\alpha_{i,j,1}, \dots, \alpha_{i,j,\mu}) \overset{R}{\leftarrow} (\mathcal{M}_{\mathrm{eff}} \setminus \{0\})^\mu \ ,$$
   $$(\gamma_{i,j,1}, \dots, \gamma_{i,j,\mu}) \overset{R}{\leftarrow} (\mathbb{F}_p)^\mu \ ,$$
   and set $c_{i,j,k} \leftarrow \mathsf{Rnd}\big((\gamma_{i,j,k} \boxdot c_i) \boxplus (\alpha_{i,j,k} - \gamma_{i,j,k} \cdot j)\big)$ for each $k \in [1, \mu]$.

   (b) Choose $\alpha_{\perp,1}, \dots, \alpha_{\perp,\mu} \overset{R}{\leftarrow} \mathcal{M}_{\mathrm{eff}} \setminus \{0\}$ and set $c_{\perp,k} \leftarrow \mathsf{Enc}(\alpha_{\perp,k})$ for each $k \in [1, \mu]$.

   (c) Send, to Client, all the ciphertexts $c_{i,j,k}$ and $c_{\perp,k}$ in a uniformly random order.

2. [Client $\to$ Server (1)] Let $\overline{c}_1, \dots, \overline{c}_{\overline{N}}$ (where $\overline{N} \overset{\mathrm{def}}{=} N_S\mu + \mu$) denote the ciphertexts for Π sent from Server at the previous step. Client executes the following:

   (a) For each $\overline{i} \in [1, \overline{N}]$, compute $\overline{m}_{\overline{i}} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\overline{c}_{\overline{i}})$. If the number of $\overline{i} \in [1, \overline{N}]$ satisfying $\overline{m}_{\overline{i}} \neq \perp$ is not equal to $N\mu + \mu$, then abort the protocol.

   (b) For each $\overline{i} \in [1, \overline{N}]$, set
   $$\overline{\overline{c}}_{\overline{i}} \leftarrow \begin{cases} \mathsf{Enc}^\dagger(\overline{m}_{\overline{i}}) & \text{if } \overline{m}_{\overline{i}} \neq \perp \ , \\ \mathsf{Enc}^\dagger(0) & \text{if } \overline{m}_{\overline{i}} = \perp \ . \end{cases}$$

   (c) Send $\overline{\overline{c}}_1, \dots, \overline{\overline{c}}_{\overline{N}}$ to Server in this order.

3. [Server $\to$ Client (2)] If the message sent from Client at the previous step does not consist of $\overline{N}$ valid ciphertexts for $\Pi^\dagger$ with respect to the key $\mathsf{pk}^\dagger$, then Server aborts the protocol. Otherwise, Server first permutes the $\overline{N}$ ciphertexts in the reverse way of Step 1c, yielding ciphertexts $\widetilde{c}_{i,j,k}$ ($i \in [1, N]$, $j \in S_i$, $k \in [1, \mu]$) and $\widetilde{c}_{\perp,k}$ ($k \in [1, \mu]$). Then Server executes the following:
   (a) For each $i \in [1, N]$ and $j \in S_i$, take a vector $\vec{w}_{i,j} = (w_{i,j,1}, \dots, w_{i,j,\mu}) \in (\mathbb{F}_{p^\dagger})^\mu$ uniformly at random subject to the condition
   $$\vec{w}_{i,j} \cdot \vec{\alpha}_{i,j} = \sum_{k=1}^{\mu} w_{i,j,k} \alpha_{i,j,k} = 0 \text{ in } \mathbb{F}_{p^\dagger} \quad . \tag{2}$$
   (b) Choose $w_{\perp,1}, \dots, w_{\perp,\mu} \xleftarrow{R} \mathbb{F}_{p^\dagger}$.
   (c) Set
   $$c^\dagger_{\mathrm{flag}} \leftarrow \left(\boxplus^\dagger_{i\in[1,N], j\in S_i, k\in[1,\mu]} (w_{i,j,k} \boxdot^\dagger \widetilde{c}_{i,j,k}) \right) \boxplus^\dagger \left(\boxplus^\dagger_{k\in[1,\mu]} (w_{\perp,k} \boxdot^\dagger (\widetilde{c}_{\perp,k} \boxminus^\dagger \alpha_{\perp,k})) \right) \quad . \tag{3}$$
   (d) For $h \in [1, \nu]$, choose $\alpha_{\mathrm{chk},h} \xleftarrow{R} \mathcal{M}^\dagger_{\mathrm{eff}}$ and $\gamma_{\mathrm{chk},h} \xleftarrow{R} \mathbb{F}_{p^\dagger}$, and set
   $$\overline{c}^\dagger_{\mathrm{chk},h} \leftarrow \alpha_{\mathrm{chk},h} \boxplus^\dagger (\gamma_{\mathrm{chk},h} \boxdot^\dagger c^\dagger_{\mathrm{flag}})$$
   and $c^\dagger_{\mathrm{chk},h} \leftarrow \mathsf{Rnd}^\dagger(\overline{c}^\dagger_{\mathrm{chk},h})$. Then send $c^\dagger_{\mathrm{chk},1}, \dots, c^\dagger_{\mathrm{chk},\nu}$ to Client.

4. [Client $\to$ Server (2)] Given the ciphertexts $c^\dagger_{\mathrm{chk},1}, \dots, c^\dagger_{\mathrm{chk},\nu}$, for each $h \in [1, \nu]$, Client computes $m_{\mathrm{chk},h} \leftarrow \mathsf{Dec}^\dagger_{\mathsf{sk}^\dagger}(c^\dagger_{\mathrm{chk},h})$, and aborts the protocol if $m_{\mathrm{chk},h} = \perp$. Otherwise, Client sends $m_{\mathrm{chk},1}, \dots, m_{\mathrm{chk},\nu}$ to Server.

5. [Server's output] For the message $m_{\mathrm{chk},1}, \dots, m_{\mathrm{chk},\nu}$ sent from Client, Server first checks if $m_{\mathrm{chk},h} = \alpha_{\mathrm{chk},h}$ for all $h \in [1, \nu]$. If this is not satisfied, then Server aborts the protocol. Otherwise, for each $i \in [1, N]$ and $h \in [1, L_i]$, Server sets
   $$\overline{c}^\dagger_{i,h} \leftarrow \boxplus^\dagger_{j\in S_i} \left(({\alpha_{i,j,1}}^{-1} \varphi_{i,h}(j)) \boxdot^\dagger \widetilde{c}_{i,j,1} \right)$$
   where the inverse of each $\alpha_{i,j,1}$ is taken in $\mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$, and $c^\dagger_{i,h} \leftarrow \mathsf{Rnd}^\dagger(\overline{c}^\dagger_{i,h})$. Then Server outputs all the $c^\dagger_{i,h}$'s.

Remark 7. Step 3 of our protocol implicitly assumed that Server (without the secret key) can efficiently verify that an object sent from possibly malicious Client is a valid ciphertext for the AHE scheme $\Pi^\dagger$. This is in fact not possible for 2LHE schemes in [3, 8], but we can modify (as in Section 5) our protocol for such schemes in a way that the verifiability for valid ciphertexts is not needed.

Remark 8. When Π and $\Pi^\dagger$ are the same scheme and the key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$ are equal as well, the ciphertext $\overline{\overline{c}}_{\overline{i}}$ in Step 2b for the case $\overline{m}_{\overline{i}} \neq \perp$ can be generated by just rerandomizing the corresponding ciphertext $\overline{c}_{\overline{i}}$ instead of encrypting $\overline{m}_{\overline{i}}$ from scratch. This slightly improves the Client-side efficiency.

4.3 Correctness with Honest Client

Theorem 9. *Suppose that both Server and Client honestly execute the protocol. Then the protocol is aborted with probability at most $\varepsilon_{\mathrm{cor}} = (N_S - N)\mu \cdot |\mathcal{M}_{\mathrm{eff}}|/p$. Conditioned on that the protocol is not aborted, Server's output distribution is identical to the distribution of the functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$.*

Proof. For Steps 1 and 2, we have $\mathsf{Pt}[c_{\perp,k}] = \alpha_{\perp,k} \in \mathcal{M}_{\mathrm{eff}}$. On the other hand, essentially the same argument as Theorem 5 implies that $\mathsf{Pt}[c_{i,\mathsf{Pt}[c_i],k}] = \mathsf{Pt}[\widetilde{c}_{i,\mathsf{Pt}[c_i],k}] = \alpha_{i,\mathsf{Pt}[c_i],k} \in \mathcal{M}_{\mathrm{eff}}$ and $\mathsf{Pt}[c_{i,j,k}]$ is uniformly random over $\mathcal{M} = \mathbb{F}_p$ when $j \neq \mathsf{Pt}[c_i]$. Client aborts in Step 2 only when some of the latter $(N_S - N)\mu$ random plaintexts belongs to $\mathcal{M}_{\mathrm{eff}}$, which occurs with probability at most $\varepsilon_{\mathrm{cor}}$.

From now, we assume that Client did not abort in Step 2. Server does not abort in Step 3, as now Client has honestly sent the valid ciphertexts. On the other hand, the argument above implies that the vector $(\mathsf{Pt}[\widetilde{c}_{i,j,1}], \dots, \mathsf{Pt}[\widetilde{c}_{i,j,\mu}])$ is either $\vec{\alpha}_{i,j}$ or the zero vector. In any case, the first term of Eq.(3) has plaintext 0 due to Eq.(2), while the second term of Eq.(3) also has plaintext 0 by the argument above. Hence we have $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = 0$ and $\mathsf{Pt}[c^{\dagger}_{\mathrm{chk},h}] = \alpha_{\mathrm{chk},h} \in \mathcal{M}^{\dagger}_{\mathrm{eff}}$ for each h. Therefore, in Step 4, Client does not abort and sends $m_{\mathrm{chk},h} = \alpha_{\mathrm{chk},h}$ back to Server, which lets Server not abort in Step 5. Finally, an argument similar to Theorem 5 implies that Server's output $c^{\dagger}_{i,h}$ is a random ciphertext of $\varphi_{i,h}(\mathsf{Pt}[c_i])$, where the term $\varphi_h(\mathsf{Pt}[c]) \cdot 1$ in Eq.(1) is replaced by ${\alpha_{i,\mathsf{Pt}[c_i],1}}^{-1} \varphi_{i,h}(\mathsf{Pt}[c_i]) \cdot \alpha_{i,\mathsf{Pt}[c_i],1}$. Hence Theorem 9 holds. □

4.4 Security against Malicious Client

By the same reason as explained in the first paragraph of Section 3.4, here we consider the security against malicious Client only.

Theorem 10. *Suppose that Server honestly executes the protocol. Let*

$$\varepsilon_{\mathrm{sec}} = \varepsilon_{\mathrm{sec},1} + \varepsilon_{\mathrm{sec},2} + \max(\varepsilon_{\mathrm{sec},3}, \varepsilon_{\mathrm{sec},4}) + 1/p^{\dagger} \ ,$$

where

$$\varepsilon_{\mathrm{sec},1} \overset{\mathrm{def}}{=} \frac{(N_S - N)\mu \cdot |\mathcal{M}_{\mathrm{eff}}|}{p} \ , \ \varepsilon_{\mathrm{sec},2} \overset{\mathrm{def}}{=} \frac{1}{|\mathcal{M}^{\dagger}_{\mathrm{eff}}|^{\nu}} \ ,$$
$$\varepsilon_{\mathrm{sec},3} \overset{\mathrm{def}}{=} \frac{1}{(|\mathcal{M}_{\mathrm{eff}}| - 1)^{\mu - 1}} \ , \ \varepsilon_{\mathrm{sec},4} \overset{\mathrm{def}}{=} N \cdot \binom{(N+1)\mu}{\mu}^{-1} \ .$$

Then our protocol is secure against malicious Client with statistical distance at most $\varepsilon_{\mathrm{sec}}$ (see Definition 4 for the terminology).

Proof. Let $\mathcal{A}$ be Client's PPT algorithm in a real protocol execution. We define a PPT algorithm $\mathcal{B}$ for Client in an ideal protocol execution. The internal randomness r_{C} for $\mathcal{B}$ consists of two parts $r_{\mathsf{C},1}$ and $r_{\mathsf{C},2}$ where $r_{\mathsf{C},1}$ is the randomness for running $\mathcal{A}$. Given the common part pp of the input and Client's local input x_{C}, $\mathcal{B}$ proceeds as follows:

1. By using the randomness $r_{\mathsf{C},2}$, $\mathcal{B}$ independently generates (by using pk) $N\mu + \mu$ uniformly random ciphertexts for uniformly random plaintexts in $\mathcal{M}_{\mathrm{eff}} \setminus \{0\}$ and $\overline{N} - (N\mu + \mu)$ uniformly random ciphertexts for uniformly random plaintexts in $\mathcal{M} \setminus \mathcal{M}_{\mathrm{eff}}$. $\mathcal{B}$ permutes these ciphertexts uniformly at random, and then executes Step 2 of $\mathcal{A}$ with randomness $r_{\mathsf{C},1}$ where the permuted ciphertexts $\overline{c}_{\mathrm{sim},\overline{i}}$ ($\overline{i} \in [1, \overline{N}]$) are given to $\mathcal{A}$ as the message from Server.

2. When the $\mathcal{A}$ aborts before outputting a message to Server, $\mathcal{B}$ decides that $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{abort}$ and
$$\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathcal{A}_{\mathsf{out}}(\mathsf{pp}, x_{\mathsf{C}}, r_{\mathsf{C},1}, (\mathsf{msg}_1))$$
where msg_1 is the list of the $\overline{c}_{\mathrm{sim},\overline{i}}$'s. Otherwise, $\mathcal{B}$ checks the message returned by $\mathcal{A}$, and if it does not consist of $\overline{N}$ valid ciphertexts for $\Pi^{\dagger}$, then $\mathcal{B}$ decides that
$$\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{abort}$$
and
$$\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathcal{A}_{\mathsf{out}}(\mathsf{pp}, x_{\mathsf{C}}, r_{\mathsf{C},1}, (\mathsf{msg}_1)) \ .$$

3. In the other case where $\mathcal{A}$ has returned $\overline{N}$ valid ciphertexts $\overline{\overline{c}}_{\mathrm{sim},\overline{i}}$ ($\overline{i} \in [1, \overline{N}]$), $\mathcal{B}$ checks if the following condition is satisfied:

$$\mathsf{Dec}_{\mathsf{sk}}(\overline{c}_{\mathrm{sim},\overline{i}}) = \mathsf{Dec}^{\dagger}_{\mathsf{sk}^{\dagger}}(\overline{\overline{c}}_{\mathrm{sim},\overline{i}}) \neq \bot \text{ or } (\mathsf{Dec}_{\mathsf{sk}}(\overline{c}_{\mathrm{sim},\overline{i}}), \mathsf{Dec}^{\dagger}_{\mathsf{sk}^{\dagger}}(\overline{\overline{c}}_{\mathrm{sim},\overline{i}})) = (\bot, 0) \ .$$

   We call the case *valid* if the condition is satisfied for all $\overline{i} \in [1, \overline{N}]$, and *invalid* otherwise.

4. In the invalid case, by using the randomness $r_{\mathsf{C},2}$, $\mathcal{B}$ generates (by using $\mathsf{pk}^{\dagger}$) ν uniformly random ciphertexts $c^{\dagger}_{\mathrm{sim,chk},h}$ ($h \in [1, \nu]$) for uniformly random plaintexts in $\mathcal{M}^{\dagger}$. Then $\mathcal{B}$ decides that $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{abort}$ and

$$\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathcal{A}_{\mathsf{out}}(\mathsf{pp}, x_{\mathsf{C}}, r_{\mathsf{C},1}, (\mathsf{msg}_1, \mathsf{msg}_2))$$

   where msg_2 is the list of the ciphertexts $c^{\dagger}_{\mathrm{sim,chk},h}$.

5. On the other hand, in the valid case, by using the randomness $r_{\mathsf{C},2}$, $\mathcal{B}$ generates $\alpha_{\mathrm{sim,chk},h} \xleftarrow{R} \mathcal{M}^{\dagger}_{\mathrm{eff}}$ and $c^{\dagger}_{\mathrm{sim,chk},h} \leftarrow \mathsf{Enc}^{\dagger}(\alpha_{\mathrm{sim,chk},h})$ for $h \in [1, \nu]$. $\mathcal{B}$ decides that

$$\mathcal{B}(\mathsf{out}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathcal{A}_{\mathsf{out}}(\mathsf{pp}, x_{\mathsf{C}}, r_{\mathsf{C},1}, (\mathsf{msg}_1, \mathsf{msg}_2))$$

   where msg_2 is the list of the ciphertexts $c^{\dagger}_{\mathrm{sim,chk},h}$. Then $\mathcal{B}$ executes Step 4 of $\mathcal{A}$ with randomness $r_{\mathsf{C},1}$ where msg_2 is given to $\mathcal{A}$ as the message from Server. If $\mathcal{A}$ does not return a message consisting of ν values $m_{\mathrm{sim,chk},h}$ ($h \in [1, \nu]$), including the case where $\mathcal{A}$ aborts the protocol, then $\mathcal{B}$ decides that $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{abort}$.

6. When the $\mathcal{A}$ has returned ν values $m_{\mathrm{sim,chk},h}$ ($h \in [1, \nu]$), $\mathcal{B}$ checks whether $m_{\mathrm{sim,chk},h} = \alpha_{\mathrm{sim,chk},h}$ holds for all $h \in [1, \nu]$. If the condition is not satisfied, then $\mathcal{B}$ decides that $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{abort}$. On the other hand, if the condition is satisfied, then $\mathcal{B}$ decides that $\mathcal{B}(\mathsf{reply}; \mathsf{pp}, x_{\mathsf{C}}; r_{\mathsf{C}}) = \mathtt{proceed}$.

Before comparing the distributions of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$ and $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}$ as in Definition 4, we modify the behavior of Server in the real protocol as follows. We first note that, by the proof of Theorem 9, the ciphertext $c_{i,j,k}$ in Step 1 of the protocol is a uniformly random ciphertext for plaintext $\alpha_{i,\mathsf{Pt}[c_i],k}$ if $j = \mathsf{Pt}[c_i]$ and for a uniformly random plaintext in $\mathcal{M} = \mathbb{F}_p$ if $j \neq \mathsf{Pt}[c_i]$. Then we modify the protocol in a way that Server directly generates $c_{i,j,k} \leftarrow \mathsf{Enc}(\beta_{i,j,k})$ where $\beta_{i,\mathsf{Pt}[c_i],k} \leftarrow \alpha_{i,\mathsf{Pt}[c_i],k}$ and $\beta_{i,j,k} \xleftarrow{R} \mathbb{F}_p$ if $j \neq \mathsf{Pt}[c_i]$. (We note that, though the resulting Server is no longer PPT, this does not matter in the proof.) This does not change the distribution of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$. Secondly, we modify the protocol in a way that, for each (i, j, k) with $j \neq \mathsf{Pt}[c_i]$, $\beta_{i,j,k}$ is chosen as $\beta_{i,j,k} \xleftarrow{R} \mathbb{F}_p \setminus \mathcal{M}_{\mathrm{eff}}$ instead of $\beta_{i,j,k} \xleftarrow{R} \mathbb{F}_p$. This yields a statistical distance for the distribution of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$ at most $|\mathcal{M}_{\mathrm{eff}}|/p$ per each (i, j, k), hence at most $(N_S - N)\mu \cdot |\mathcal{M}_{\mathrm{eff}}|/p = \varepsilon_{\mathrm{sec},1}$ in total. Summarizing, the statistical distance between the distributions of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$ in the original and the modified protocols is at most $\varepsilon_{\mathrm{sec},1}$.

From now, we compare the distributions of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$ for the modified protocol and $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}$. For the modified protocol, the argument above shows that the distribution of the ciphertexts sent from Server to Client in Step 1c (which is independent of Client's randomness) is identical to the distribution of the ciphertexts $\overline{c}_{\mathrm{sim},\overline{i}}$ ($\overline{i} \in [1, \overline{N}]$) in Step 1 of $\mathcal{B}$. Now we divide the situation in the modified protocol into the following two disjoint cases:

(R1) Client in Step 2 does not return $\overline{N}$ valid ciphertexts for $\Pi^{\dagger}$ (including the case where Client aborts the protocol);

(R2) Otherwise, i.e., Client in Step 2 returns $\overline{N}$ valid ciphertexts for $\Pi^{\dagger}$.

We also divide the situation in Step 2 of the algorithm $\mathcal{B}$ into two disjoint cases (I1) and (I2) in a similar way. We write $\mathsf{out}_{\mathsf{real},\mathcal{A}}|_E$ to denote the conditional distribution of $\mathsf{out}_{\mathsf{real},\mathcal{A}}$ conditioned on a case E, and write $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_E$ similarly. Now the argument above implies that the probabilities of Case (R1) and of Case (I1) are equal, so are (R2) and (I2). Hence it follows that

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}) \leq \max_{\mathrm{x}=1,2} \Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{Rx}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{Ix}}) \enspace .$$

In Case (R1), the modified protocol is aborted before Server sends a message in Step 3. Now Client's transcript has the same distribution as (msg_1) appeared in Step 2 of $\mathcal{B}$ for Case (I1), therefore the conditional distributions $\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R1}}$ and $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I1}}$ are identical to each other. Hence we have

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}) \leq \Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R2}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I2}}) \enspace .$$

To evaluate the statistical distance in the right-hand side, we note that the conditional distribution of ciphertexts $\overline{\overline{c}}_{\mathrm{sim},\overline{i}}$ ($\overline{i} \in [1,\overline{N}]$) in Step 3 of $\mathcal{B}$ conditioned on Case (I2) is identical to the conditional distribution of ciphertexts $\overline{\overline{c}}_{\overline{i}}$ ($\overline{i} \in [1,\overline{N}]$) sent from $\mathcal{A}$ to Server in Step 2 of the modified protocol conditioned on Case (R2). Now let π denote the permutation for ciphertexts applied in Step 1c of the modified protocol. Namely, π is a bijection from the index set

$$\{(i,j,k) \mid i \in [1,N], j \in S_i, k \in [1,\mu]\} \cup \{(\perp,k) \mid k \in [1,\mu]\}$$

to the index set $[1,\overline{N}]$. Now the ciphertexts sent from Server to Client in Step 1c satisfy that, $\overline{c}_{\overline{i}}$ ($\overline{i} \in [1,\overline{N}]$) is a uniformly random ciphertext for plaintext given by

$$\mathsf{Pt}[\overline{c}_{\overline{i}}] = \begin{cases} \alpha_{i,\mathsf{Pt}[c_i],k} \xleftarrow{R} \mathcal{M}_{\mathrm{eff}} \setminus \{0\} \\ \qquad \text{if } \pi^{-1}(\overline{i}) = (i,\mathsf{Pt}[c_i],k) \text{ with } i \in [1,N], k \in [1,\mu] \enspace , \\ \alpha_{\perp,k} \xleftarrow{R} \mathcal{M}_{\mathrm{eff}} \setminus \{0\} \\ \qquad \text{if } \pi^{-1}(\overline{i}) = (\perp,k) \text{ with } k \in [1,\mu] \enspace , \\ \beta_{i,j,k} \xleftarrow{R} \mathbb{F}_p \setminus \mathcal{M}_{\mathrm{eff}} \\ \qquad \text{if } \pi^{-1}(\overline{i}) = (i,j,k) \text{ with } i \in [1,N], j \in S_i \setminus \{\mathsf{Pt}[c_i]\}, k \in [1,\mu] \enspace . \end{cases}$$

In particular, the distributions of the $\overline{c}_{\overline{i}}$'s for the first two cases are identical to each other, and those for the third case are also identical to each other. Let I and J denote the sets of the indices $\overline{i}$ in the first two cases and the third case above, respectively. We moreover divide the set I into subsets $I_1,\dots,I_N$ and $I_\perp$ where

$$I_i \overset{\mathrm{def}}{=} \{\pi(i,\mathsf{Pt}[c_i],k) \mid k \in [1,\mu]\} \text{ for } i \in [1,N]$$

and

$$I_\perp \overset{\mathrm{def}}{=} \{\pi(\perp,k) \mid k \in [1,\mu]\} \enspace .$$

We also define

$$I' \overset{\mathrm{def}}{=} \{\overline{i} \in I \mid \mathsf{Pt}[\overline{\overline{c}}_{\overline{i}}] \neq \mathsf{Pt}[\overline{c}_{\overline{i}}]\} \text{ and } J' \overset{\mathrm{def}}{=} \{\overline{i} \in J \mid \mathsf{Pt}[\overline{\overline{c}}_{\overline{i}}] \neq 0\} \enspace .$$

Now we subdivide Case (R2) into two disjoint subcases; (R-valid) $I' = J' = \emptyset$; and (R-invalid) $I' \neq \emptyset$ or $J' \neq \emptyset$. Note that Case (R-valid) corresponds precisely to the valid case in the algorithm $\mathcal{B}$, called (I-valid); similarly Case (R-invalid) corresponds to the invalid case in $\mathcal{B}$, called (I-invalid). Hence we have

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}) \leq \max_{\mathrm{x}=\mathrm{valid},\mathrm{invalid}} \Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-x}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-x}}) \enspace .$$

We show that the conditional distributions $\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-valid}}$ and $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-valid}}$ are identical. Indeed, the same argument as Theorem 9 implies that $c^\dagger_{\mathrm{chk},h}$ ($h \in [1,\nu]$) in Step 3 of the modified protocol for Case (R-valid) is a uniformly random ciphertext for plaintext $\alpha_{\mathrm{chk},h} \xleftarrow{R} \mathcal{M}^\dagger_{\mathrm{eff}}$. These objects have distributions identical to $c^\dagger_{\mathrm{sim},\mathrm{chk},h}$ and $\alpha_{\mathrm{sim},\mathrm{chk},h}$ in Step 5 of $\mathcal{B}$ for Case (I-valid). In particular, the pair $(\mathsf{msg}_1, \mathsf{msg}_2)$ in Step 5 of $\mathcal{B}$ has a conditional distribution identical to Client's transcript in the modified protocol. It also follows that, the conditional probabilities that Step 4 of $\mathcal{A}$ does not return ν values (including the case where $\mathcal{A}$ aborts the protocol) are equal for Cases (R-valid) and (I-valid). Moreover, when the $\mathcal{A}$ returns ν values, the conditional distribution of the values $m_{\mathrm{sim,chk},h}$ in Step 6 of $\mathcal{B}$ (conditioned on the $\alpha_{\mathrm{sim,chk},h}$'s and $c^{\dagger}_{\mathrm{sim,chk},h}$'s) is also identical to the conditional distribution of the values $m_{\mathrm{chk},h}$ in Step 4 of $\mathcal{A}$ in the modified protocol (conditioned on the $\alpha_{\mathrm{chk},h}$'s and $c^{\dagger}_{\mathrm{chk},h}$'s). This implies that, the conditional probability that the protocol is not aborted, which is equivalent to $m_{\mathrm{chk},h} = \alpha_{\mathrm{chk},h}$ for all $h \in [1,\nu]$ in the modified protocol and to $m_{\mathrm{sim,chk},h} = \alpha_{\mathrm{sim,chk},h}$ for all $h \in [1,\nu]$ in $\mathcal{B}$, respectively, is also equal in the two cases. Moreover, when the protocol is not aborted, the same argument as Theorem 9 implies that Server's conditional output distribution in the modified protocol is identical to the distribution of $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$. Summarizing the arguments, it follows that $\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-valid}}$ and $\mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-valid}}$ are identical, as desired. Hence we have

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}) \le \Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-invalid}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-invalid}}) \enspace .$$

In order to evaluate the right-hand side, we further subdivide Case (R-invalid) into the following two disjoint subcases:

(R-success) $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = 0$;

(R-failure) $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] \in (\mathbb{F}_{p^{\dagger}})^{\times}$.

Then we have

$$\begin{aligned}&\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}})\\ &\le \Pr[\text{ R-success } | \text{ R-invalid }] + \Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-failure}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-invalid}}) \enspace .\end{aligned} \tag{4}$$

We first consider the second term in the right-hand side of Eq.(4). In this case, each $c^{\dagger}_{\mathrm{chk},h}$ is a perfectly random ciphertext independent of $\alpha_{\mathrm{chk},h}$ due to the property of $\mathsf{Rnd}^{\dagger}$ and the choice $\gamma_{\mathrm{chk},h} \overset{R}{\leftarrow} \mathbb{F}_{p^{\dagger}}$, therefore the conditional distribution of Client's transcript is identical to that of $(\mathsf{msg}_1, \mathsf{msg}_2)$ in $\mathcal{B}$ for Case (I-invalid). Moreover, this property of $c^{\dagger}_{\mathrm{chk},h}$ implies that Client in Step 4 can correctly choose $m_{\mathrm{chk},h} = \alpha_{\mathrm{chk},h}$ for all $h \in [1,\nu]$ (hence preventing Server to abort) with probability at most $\varepsilon_{\mathrm{sec},2}$. Hence we have

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}|_{\mathrm{R-failure}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}|_{\mathrm{I-invalid}}) \le \varepsilon_{\mathrm{sec},2} \enspace .$$

In order to evaluate the first term in the right-hand side of Eq.(4), we consider the two disjoint cases $J' = \emptyset$ and $J' \neq \emptyset$. From now, we abbreviate "R-invalid" and "R-success" into "R-i" and "R-s", respectively. We have

$$\begin{aligned}\Pr[\text{ R-s } | \text{ R-i }] &= \Pr[\text{ R-s} \wedge J' = \emptyset \mid \text{R-i }] + \Pr[\text{ R-s} \wedge J' \neq \emptyset \mid \text{R-i }]\\ &= \Pr[J' = \emptyset \mid \text{R-i }] \cdot \Pr[\text{ R-s } | \text{ R-i } \wedge J' = \emptyset] + \Pr[J' \neq \emptyset \mid \text{R-i }] \cdot \Pr[\text{ R-s } | \text{ R-i } \wedge J' \neq \emptyset]\\ &= \Pr[J' = \emptyset \mid \text{R-i }] \cdot \Pr[\text{ R-s } | \; I' \neq \emptyset = J'] + \Pr[J' \neq \emptyset \mid \text{R-i }] \cdot \Pr[\text{ R-s } | \; J' \neq \emptyset] \enspace .\end{aligned} \tag{5}$$

Write $\vec{P}_{i,j} \overset{\mathrm{def}}{=} (\mathsf{Pt}[\widetilde{c}_{i,j,1}], \dots, \mathsf{Pt}[\widetilde{c}_{i,j,\mu}])$. For the second term in the right-hand side of Eq.(5), we assume that $J' \neq \emptyset$, and take an element $\bar{i} = \pi(i,j,k) \in J'$, hence $j \neq \mathsf{Pt}[c_i]$ and $\mathsf{Pt}[\overline{c}_{\bar{i}}] = \mathsf{Pt}[\widetilde{c}_{i,j,k}] \neq 0$. In particular, $\vec{P}_{i,j} \in (\mathbb{F}_{p^{\dagger}})^{\mu}$ is a non-zero vector. Moreover, as $j \neq \mathsf{Pt}[c_i]$ and hence Client's view is independent of the vector $\vec{\alpha}_{i,j}$ by the construction of the modified protocol, distributions of $\vec{P}_{i,j}$ and $\vec{\alpha}_{i,j}$ are independent as well. Among the $|\mathcal{M}_{\mathrm{eff}} \setminus \{0\}|^{\mu}$ choices of $\vec{\alpha}_{i,j}$, at most $|\mathcal{M}_{\mathrm{eff}} \setminus \{0\}|$ of them is linearly dependent with $\vec{P}_{i,j}$, which occurs with probability at most $|\mathcal{M}_{\mathrm{eff}} \setminus \{0\}|^{-(\mu-1)} \le (|\mathcal{M}_{\mathrm{eff}}| - 1)^{-(\mu-1)} = \varepsilon_{\mathrm{sec},3}$. On the other hand, when $\vec{P}_{i,j}$ and $\vec{\alpha}_{i,j}$ are linearly independent, for the uniformly random vector $\vec{w}_{i,j} \in (\mathbb{F}_{p^{\dagger}})^{\mu}$ orthogonal to $\vec{\alpha}_{i,j}$, the inner product $\vec{w}_{i,j} \cdot \vec{P}_{i,j}$ takes a uniformly random value over $\mathbb{F}_{p^{\dagger}}$. Now one of the terms in Eq.(3) has perfectly random plaintext, so does $c^{\dagger}_{\mathrm{flag}}$ as well, therefore $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = 0$ holds with probability $1/p^{\dagger}$. Summarizing, we have

$$\Pr[\text{ R-s } | \; J' \neq \emptyset] \le \varepsilon_{\mathrm{sec},3} + \frac{1}{p^{\dagger}} \enspace .$$

For the first term in the right-hand side of Eq.(5), we assume that $I' \neq \emptyset = J'$. We define the following three disjoint events:

(E_1) $I' \cap I_\perp \neq \emptyset$;

(E_2) $I' \cap I_\perp = \emptyset$ and $\emptyset \neq I' \cap I_{i^*} \neq I_{i^*}$ for some $i^* \in [1, N]$;

(E_3) None of the above is satisfied, i.e., I' is the disjoint union of at least one sets $I_{i_1}, \dots, I_{i_\delta}$ with $1 \leq i_1 < \cdots < i_\delta \leq N$.

Then we have

$$\Pr[\text{ R-s} \mid I' \neq \emptyset = J'] = \sum_{x=1}^{3} \Pr[E_x \mid I' \neq \emptyset = J'] \cdot \Pr[\text{ R-s} \mid I' \neq \emptyset = J' \wedge E_x] \ .$$

In the case E_1, we can take an element $\bar{i} = \pi(\perp, k) \in I' \cap I_\perp$, hence $\mathsf{Pt}[\widetilde{c}_{\perp,k}] = \mathsf{Pt}[\overline{\overline{c}}_{\bar{i}}] \neq \mathsf{Pt}[\overline{c}_{\bar{i}}] = \alpha_{\perp,k}$. Now $\mathsf{Pt}[\widetilde{c}_{\perp,k} \boxminus^\dagger \alpha_{\perp,k}] \in (\mathbb{F}_{p^\dagger})^\times$, while $w_{\perp,k}$ is uniformly random over $\mathbb{F}_{p^\dagger}$. This implies that one of the terms in Eq.(3) has perfectly random plaintext, so does $c^\dagger_{\mathrm{flag}}$ as well, therefore $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}] = 0$ holds with probability $1/p^\dagger$. On the other hand, in the case E_2, we can take elements $\bar{i}_1 = \pi(i^*, \mathsf{Pt}[c_{i^*}], k_1) \in I' \cap I_{i^*}$ and $\bar{i}_2 = \pi(i^*, \mathsf{Pt}[c_{i^*}], k_2) \in I_{i^*} \setminus I'$. Now we have

$$\mathsf{Pt}[\widetilde{c}_{i^*,\mathsf{Pt}[c_{i^*}],k_1}] = \mathsf{Pt}[\overline{\overline{c}}_{\bar{i}_1}] \neq \mathsf{Pt}[\overline{c}_{\bar{i}_1}] = \alpha_{i^*,\mathsf{Pt}[c_{i^*}],k_1}$$

and

$$\mathsf{Pt}[\widetilde{c}_{i^*,\mathsf{Pt}[c_{i^*}],k_2}] = \mathsf{Pt}[\overline{\overline{c}}_{\bar{i}_2}] = \mathsf{Pt}[\overline{c}_{\bar{i}_2}] = \alpha_{i^*,\mathsf{Pt}[c_{i^*}],k_2} \neq 0 \ .$$

This implies that $\vec{P}_{i^*,\mathsf{Pt}[c_{i^*}]}$ and $\vec{\alpha}_{i^*,\mathsf{Pt}[c_{i^*}]}$ are linearly independent, therefore the same argument as the case $J' \neq \emptyset$ above implies that $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}]$ is uniformly random and becomes 0 with probability $1/p^\dagger$. Hence we have

$$\begin{aligned}\Pr[\text{ R-s} \mid I' \neq \emptyset = J'] &\leq \sum_{x=1}^{2} \Pr[E_x \mid I' \neq \emptyset = J'] \cdot \frac{1}{p^\dagger} + \Pr[E_3 \mid I' \neq \emptyset = J'] \\ &\leq \frac{1}{p^\dagger} + \Pr[E_3 \mid I' \neq \emptyset = J'] \ .\end{aligned}$$

Finally, we evaluate the probability $\Pr[E_3 \mid I' \neq \emptyset = J']$. When $|I'|$ is not a multiple of μ, the event E_3 never occurs. From now, we consider the case that $|I'| = \delta\mu$ for an integer $\delta \geq 1$. Now the cases for different choices of δ are disjoint; we evaluate the maximal probability of the event for various choices of δ. We recall that the distributions of ciphertexts $\overline{c}_{\bar{i}}$ with $\bar{i} \in I$ are identical to each other. Therefore, from Client's view, the permutation π is perfectly random subject to the condition that π maps the set

$$\{(i, \mathsf{Pt}[c_i], k) \mid i \in [1, N], k \in [1, \mu]\} \cup \{(\perp, k) \mid k \in [1, \mu]\}$$

onto I. Now for each $(i_1, \dots, i_\delta)$, π maps the subset $\{(i_\rho, \mathsf{Pt}[c_{i_\rho}], k) \mid \rho \in [1, \delta], k \in [1, \mu]\}$ onto the subset I' of I with probability $\binom{(N+1)\mu}{\delta\mu}^{-1}$. As there are $\binom{N}{\delta}$ choices of $(i_1, \dots, i_\delta)$, the event E_3 occurs for this δ with probability $f(\delta)/((N+1)\mu)!$ where

$$f(\delta) \overset{\mathrm{def}}{=} \binom{N}{\delta} (\delta\mu)!((N+1-\delta)\mu)! \ .$$

Now we have $f(N+1-\delta) = (\delta/(N+1-\delta)) \cdot f(\delta)$, which implies that $f(\delta)$ is not the maximum value of f if $\delta > (N+1)/2$. Moreover, for $2 \leq \delta \leq (N+1)/2$, we have (as $N+1-\delta \geq \delta - 1$)

$$\begin{aligned} f(\delta-1) &= \frac{\delta}{N+1-\delta} \frac{((\delta-1)\mu)!}{(\delta\mu)!} \frac{((N+2-\delta)\mu)!}{((N+1-\delta)\mu)!} f(\delta) \\ &= \frac{\delta}{N+1-\delta} \frac{(N+1-\delta)\mu+1}{(\delta-1)\mu+1} \cdots \frac{(N+1-\delta)\mu+\mu}{(\delta-1)\mu+\mu} f(\delta) \\ &\geq \frac{\delta}{N+1-\delta} \cdot 1 \cdots 1 \cdot \frac{N+2-\delta}{\delta} f(\delta) = \frac{N+2-\delta}{N+1-\delta} f(\delta) \geq f(\delta) \enspace . \end{aligned}$$

This implies that the maximum value of $f(\delta)$ is attained at $\delta = 1$. Therefore, we have

$$\Pr[E_3 \mid I' \neq \emptyset = J'] \leq \frac{f(1)}{((N+1)\mu)!} = N \cdot \frac{\mu!(N\mu)!}{((N+1)\mu)!} = \varepsilon_{\mathrm{sec},4}$$

and hence

$$\Pr[\text{ R-s} \mid I' \neq \emptyset = J'] \leq \frac{1}{p^\dagger} + \varepsilon_{\mathrm{sec},4} \enspace .$$

Now Eq.(5) becomes

$$\begin{aligned} \Pr[\text{ R-s} \mid \text{R-i }] &\leq \Pr[J' = \emptyset \mid \text{R-i }] \left(\varepsilon_{\mathrm{sec},4} + \frac{1}{p^\dagger}\right) + \Pr[J' \neq \emptyset \mid \text{R-i }] \left(\varepsilon_{\mathrm{sec},3} + \frac{1}{p^\dagger}\right) \\ &\leq \max\left(\varepsilon_{\mathrm{sec},3} + \frac{1}{p^\dagger}, \varepsilon_{\mathrm{sec},4} + \frac{1}{p^\dagger}\right) = \max(\varepsilon_{\mathrm{sec},3}, \varepsilon_{\mathrm{sec},4}) + \frac{1}{p^\dagger} \enspace . \end{aligned}$$

By Eq.(4) and the arguments above, we have

$$\Delta(\mathsf{out}_{\mathsf{real},\mathcal{A}}, \mathsf{out}_{\mathsf{ideal},\mathcal{B}}) \leq \max(\varepsilon_{\mathrm{sec},3}, \varepsilon_{\mathrm{sec},4}) + \frac{1}{p^\dagger} + \varepsilon_{\mathrm{sec},2}$$

as desired. This completes the proof of Theorem 10. □

4.5 Parameters for Batch Executions

The complexity of our proposed protocol against malicious Client is almost proportional to the parameter μ and is also dependent slightly on the other parameter ν. Table 1 shows examples of the parameters μ and ν that make the bound $\varepsilon_{\mathrm{sec}}$ in Theorem 10 at most 2^{-128} for various numbers N of Server's input ciphertexts. Here we consider the case of 10-bit encrypted values, i.e., $|S_i| = 2^{10} = 1024$. We assume the use of the lifted-ElGamal cryptosystem, where the spaces $\mathcal{M}_{\mathrm{eff}}$ and $\mathcal{M}_{\mathrm{eff}}^\dagger$ of effective plaintexts have to be fairly small for keeping decryption efficiency. Here we take $|\mathcal{M}_{\mathrm{eff}}| = |\mathcal{M}_{\mathrm{eff}}^\dagger| = 10000$. On the other hand, the orders $p, p^\dagger$ of plaintext spaces (i.e., orders of the underlying groups) are significantly longer than 128 bits for the sake of security of the cryptosystems. Accordingly, we can ignore the terms $\varepsilon_{\mathrm{sec},1}$ and $1/p^\dagger$ in $\varepsilon_{\mathrm{sec}}$. Then a calculation shows that, when N is not very large, the term $\varepsilon_{\mathrm{sec},4}$ is dominant in $\varepsilon_{\mathrm{sec}}$; and as N increases, the value of μ satisfying the bound is decreased and hence the communication overhead per one input ciphertext is reduced. On the other hand, the value of μ satisfying the bound is lower bounded due to the term $\varepsilon_{\mathrm{sec},3}$ (precisely, $\mu \geq 11$ in the current case), and when N is large, $\varepsilon_{\mathrm{sec},3}$ becomes dominant rather than $\varepsilon_{\mathrm{sec},4}$ and hence the lower bound for μ is attained. This argument shows that, a batch execution of our protocol improves the efficiency, but the efficiency improvement is saturated when N attains a certain threshold.

5 Our Protocols over 2LHE Schemes

In this section, we describe a way of modifying our protocol with malicious Client (in Section 4) to resolve the problem of invalid ciphertexts mentioned in Remark 7 when the level-1 part of the 2LHE scheme in [3] is used as the AHE scheme $\Pi^\dagger$. A similar idea might be also applicable to another 2LHE scheme in [8].

Table 1: Examples of parameters μ and ν for our proposed protocol against malicious Client to attain the bound $\varepsilon_{\mathrm{sec}} \leq 2^{-128}$ in Theorem 10

N	1	10	100	1000	10000
μ	66	28	18	13	11
ν	10	10	10	10	10

5.1 The 2LHE Scheme

Here we summarize some properties of the 2LHE scheme in [3] relevant to our protocol construction (see Appendix A for the other details). The plaintext space for the scheme is $\mathcal{M} = \mathbb{F}_p$ with (large) prime p. The set $\mathcal{C}^{\langle 1\rangle}$ of level-1 ciphertexts is the union of disjoint subsets $\mathcal{C}^{\langle 1\rangle}_{m,m'}$ parameterized by two elements m, m' of $\mathcal{M}$; and $\mathcal{C}^{\langle 1\rangle}_{m} = \mathcal{C}^{\langle 1\rangle}_{m,m}$ is the set of valid level-1 ciphertexts for plaintext m, hence $\mathcal{C}^{\langle 1\rangle}_{\mathrm{val}} = \bigcup_{m \in \mathcal{M}} \mathcal{C}^{\langle 1\rangle}_{m,m}$. On the other hand, any level-2 ciphertext is valid; $\mathcal{C}^{\langle 2\rangle} = \mathcal{C}^{\langle 2\rangle}_{\mathrm{val}} = \bigcup_{m \in \mathcal{M}} \mathcal{C}^{\langle 2\rangle}_{m}$. Now the level-1 homomorphic operations are extended also to invalid ciphertexts as follows: given $c_1 \in \mathcal{C}^{\langle 1\rangle}_{m_1,m'_1}$ and $c_2 \in \mathcal{C}^{\langle 1\rangle}_{m_2,m'_2}$, we have $c_1 \boxplus^{\langle 1\rangle} c_2 \in \mathcal{C}^{\langle 1\rangle}_{m_1+m_2,m'_1+m'_2}$ and $\boxminus^{\langle 1\rangle} c_1 \in \mathcal{C}^{\langle 1\rangle}_{-m_1,-m'_1}$, and $\mathsf{Rnd}^{\langle 1\rangle}(c_1)$ outputs a uniformly random element of $\mathcal{C}^{\langle 1\rangle}_{m_1,m'_1}$. Moreover, the homomorphic multiplication operation is also extended to invalid ciphertexts in the following manner:

$$\mathcal{C}^{\langle 1\rangle}_{m_1,m'_1} \boxtimes^{\langle 1\rangle} \mathcal{C}^{\langle 1\rangle}_{m_2,m'_2} \subseteq \mathcal{C}^{\langle 2\rangle}_{m_1 \cdot m'_2} \text{ for any } m_1, m'_1, m_2, m'_2 \in \mathcal{M} \ . \tag{6}$$

5.2 The Modified Protocol

The setting here is the same as Section 4.1 (in particular, Client may be malicious) except that now the level-1 part $\Pi^{\dagger\langle 1\rangle}$ of the 2LHE scheme $\Pi^{\dagger}$ in Section 5.1 plays the role of $\Pi^{\dagger}$ in the original setting. Then we modify Step 3 of the protocol in Section 4.2 in the following manner:

- At the beginning of the step, if the message sent from Client at the previous step does not consist of $\overline{N}$ elements in the union of $\mathcal{C}^{\dagger\langle 1\rangle}_{m,m'}$ over all $m, m' \in \mathcal{M}^{\dagger} = \mathbb{F}_{p^{\dagger}}$, then Server aborts the protocol.

- At Step 3c, we rename the original ciphertext $c^{\dagger}_{\mathrm{flag}}$ as $c^{\dagger\dagger}_{\mathrm{flag}}$, and then newly define $c^{\dagger}_{\mathrm{flag}}$ by choosing $\widehat{w}_{i,j,k} \xleftarrow{R} \mathbb{F}_{p^{\dagger}}$ for each $i \in [1,N]$, $j \in S_i$, and $k \in [1,\mu]$, choosing $\widehat{w}_{\perp,k} \xleftarrow{R} \mathbb{F}_{p^{\dagger}}$ for each $k \in [1,\mu]$, and setting

$$\begin{aligned} c^{\dagger}_{\mathrm{flag}} \leftarrow (c^{\dagger\dagger}_{\mathrm{flag}} \boxtimes^{\dagger\langle 1\rangle} 1) \boxplus^{\dagger\langle 2\rangle} &\left(\boxplus^{\dagger\langle 2\rangle}_{i \in [1,N], j \in S_i, k \in [1,\mu]} (\widehat{w}_{i,j,k} \boxdot^{\dagger\langle 2\rangle} \mathsf{Val}(\widetilde{c}_{i,j,k})) \right) \\ \boxplus^{\dagger\langle 2\rangle} &\left(\boxplus^{\dagger\langle 2\rangle}_{k \in [1,\mu]} (\widehat{w}_{\perp,k} \boxdot^{\dagger\langle 2\rangle} \mathsf{Val}(\widetilde{c}_{\perp,k})) \right) \end{aligned} \tag{7}$$

  where we define

$$\mathsf{Val}(c) \overset{\mathrm{def}}{=} (c \boxtimes^{\dagger\langle 1\rangle} 1) \boxplus^{\dagger\langle 2\rangle} (-1 \boxtimes^{\dagger\langle 1\rangle} c) \ . \tag{8}$$

  Hence $c^{\dagger}_{\mathrm{flag}}$ is now a level-2 (rather than level-1) ciphertext, therefore the following procedures relevant to the objects $\alpha_{\mathrm{chk},h}$, $\overline{c}^{\dagger}_{\mathrm{chk},h}$, and $c^{\dagger}_{\mathrm{chk},h}$ are proceeded by using the level-2 part of $\Pi^{\dagger}$ instead of the level-1 part.

Intuitively, this modification intends to let the ciphertext $c^{\dagger}_{\mathrm{flag}}$ (originally used for detecting Client's dishonest choice of plaintexts) also play the role of checking if Client's ciphertexts are valid. In more detail, for any $c \in \mathcal{C}^{\dagger\langle 1\rangle}_{m,m'}$, Eq.(6) implies that $\mathsf{Val}(c) \in \mathcal{C}^{\dagger\langle 2\rangle}_{m-m'}$. In other words, we have $\mathsf{Pt}[\mathsf{Val}(c)] = 0$ if and only if c is valid. This implies that, if some of the ciphertexts $\widetilde{c}_{i,j,k}$ and $\widetilde{c}_{\perp,k}$ is invalid, then at least one of the terms in Eq.(7) has a uniformly random plaintext, so does $c^{\dagger}_{\mathrm{flag}}$, therefore $\Pr[\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = 0] = 1/p^{\dagger}$. On the other hand, if all of $\widetilde{c}_{i,j,k}$ and $\widetilde{c}_{\perp,k}$ are valid, then $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = \mathsf{Pt}[c^{\dagger\dagger}_{\mathrm{flag}}]$ and hence this additional procedure does not affect the result of the protocol. Owing to this property, essentially the same arguments as our original protocol imply the following results.

Theorem 11. *Suppose that both Server and Client honestly execute the protocol. Then the protocol is aborted with probability at most $\varepsilon_{\mathrm{cor}}$, where $\varepsilon_{\mathrm{cor}}$ is the same as in Theorem 9. If the protocol is not aborted, then the output distribution for Server is identical to the distribution of the functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$.*

Theorem 12. *Suppose that Server honestly executes the protocol. Then our protocol is secure against malicious Client with statistical distance at most $\varepsilon_{\mathrm{sec}}$ (see Definition 4 for the terminology), where $\varepsilon_{\mathrm{sec}}$ is the same as in Theorem 10.*

5.3 An Improved Protocol for Binary Plaintexts

Here we consider the setting of Section 5.2 (i.e., $\Pi^\dagger$ is the 2LHE scheme in [3] and Client may be malicious) and suppose moreover that $S_i = \{0,1\}$ for every i; i.e., each input ciphertext for Server has plaintext 0 or 1. In this case, by using the two-level functionality of $\Pi^\dagger$, we can construct a more efficient protocol. We note that the same idea is also applicable to any other case with $|S_i| = 2$ by using a homomorphic affine transformation that maps S_i onto $\{0,1\}$.

We recall some settings for readers' convenience. Server has N ciphertexts $c_1, \dots, c_N$ for AHE scheme Π with plaintexts in $\{0,1\}$. Server's goal is, roughly speaking, to obtain level-1 ciphertexts (in the scheme $\Pi^\dagger$) of plaintexts $\varphi_{i,h}(m_i)$ for $i \in [1,N]$ and $h \in [1, L_i]$ (without having the secret keys $\mathsf{sk}, \mathsf{sk}^\dagger$ held by Client) where $\varphi_{i,h}$ is a function $\{0,1\} \to \mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$ and m_i denotes the unknown ciphertext of c_i. Now the protocol is described as follows.

1. [Server $\to$ Client (1)] Server executes the following:
   (a) For each $i \in [1,N]$, $j \in \{0,1\}$, and $k \in [1,\mu]$, choose $\alpha_{i,j,k} \xleftarrow{R} \mathcal{M}_{\mathrm{eff}} \setminus \{0\}$ and $\gamma_{i,j,k} \xleftarrow{R} \mathbb{F}_p$, and set $c_{i,j,k} \leftarrow \mathsf{Rnd}\big((\gamma_{i,j,k} \boxdot c_i) \boxplus (\alpha_{i,j,k} - \gamma_{i,j,k} \cdot j)\big)$.
   (b) For each $i \in [1,N]$, send, to Client, two collections
   $$(c_{i,0,k})_{k\in[1,\mu]} \text{ and } (c_{i,1,k})_{k\in[1,\mu]}$$
   of ciphertexts in an independent and uniformly random order (i.e., swapping them with probability 1/2).
2. [Client $\to$ Server (1)] For each $i \in [1,N]$, let $(\overline{c}_{i,0,k})_{k\in[1,\mu]}$ and $(\overline{c}_{i,1,k})_{k\in[1,\mu]}$ denote the i-th pair of two collections of ciphertexts for Π sent from Server at the previous step. Client executes the following:
   (a) For each $i \in [1,N]$, $j \in \{0,1\}$, and $k \in [1,\mu]$, compute $\overline{m}_{i,j,k} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\overline{c}_{i,j,k})$. If, for each $i \in [1,N]$, there exists an index $j_0 = j_0(i) \in \{0,1\}$ satisfying that $\overline{m}_{i,j_0,k} \neq \perp$ and $\overline{m}_{i,1-j_0,k} = \perp$ for any $k \in [1,\mu]$, then proceed the protocol; otherwise abort the protocol.
   (b) For each $i \in [1,N]$, $j \in \{0,1\}$, and $k \in [1,\mu]$, set
   $$\overline{\overline{c}}_{i,j,k} \leftarrow \begin{cases} \mathsf{Enc}^{\dagger\langle 1\rangle}(\overline{m}_{i,j,k}) & \text{if } \overline{m}_{i,j,k} \neq \perp \ , \\ \mathsf{Enc}^{\dagger\langle 1\rangle}(0) & \text{if } \overline{m}_{i,j,k} = \perp \ . \end{cases}$$
   (c) For each $i \in [1,N]$, send two collections $(\overline{\overline{c}}_{i,0,k})_{k\in[1,\mu]}$ and $(\overline{\overline{c}}_{i,1,k})_{k\in[1,\mu]}$ to Server.
3. [Server $\to$ Client (2)] If the message sent from Client at the previous step does not consist of N pairs of two collections $(\overline{\overline{c}}_{i,0,k})_{k\in[1,\mu]}$ and $(\overline{\overline{c}}_{i,1,k})_{k\in[1,\mu]}$ ($i \in [1,N]$) of elements in the union of $\mathcal{C}^{\dagger\langle 1\rangle}_{m,m'}$ over all $m, m' \in \mathcal{M}^\dagger$, then Server aborts the protocol. Otherwise, Server first permutes the two collections in each of the N pairs in the reverse way of Step 1b, yielding collections $(\widetilde{c}_{i,0,k})_{k\in[1,\mu]}$ and $(\widetilde{c}_{i,1,k})_{k\in[1,\mu]}$ of (possibly invalid) level-1 ciphertexts in the scheme $\Pi^\dagger$ for $i \in [1,N]$. Then Server executes the following:

(a) Choose $u_{i,j,k} \xleftarrow{R} \mathbb{F}_{p^\dagger}$ for each $i \in [1,N]$, $j \in \{0,1\}$, and $k \in [1,\mu]$, and set

$$c^\dagger_{\mathrm{flag},1} \leftarrow \boxplus^{\dagger\langle 2\rangle}_{i\in[1,N],j\in\{0,1\},k\in[1,\mu]} (u_{i,j,k} \boxdot^{\dagger\langle 2\rangle} \mathsf{Val}(\widetilde{c}_{i,j,k}))$$

where $\mathsf{Val}(\cdot)$ is defined as in Eq.(8).

(b) Choose $v^{(0)}_{i,k}, \dots, v^{(4)}_{i,k} \xleftarrow{R} \mathbb{F}_{p^\dagger}$ for each $i \in [1,N]$ and $k \in [1,\mu]$, and set

$$c^\dagger_{\mathrm{flag},2} \leftarrow \boxplus^{\dagger\langle 2\rangle}_{\substack{i\in[1,N]\\ k\in[1,\mu]}} \Big(v^{(0)}_{i,k} \boxdot^{\dagger\langle 2\rangle} \Big(\mathsf{Chk}^{(0)}_{i,k,k}(v^{(1)}_{i,k}, v^{(2)}_{i,k}) \boxtimes^{\dagger\langle 1\rangle} \mathsf{Chk}^{(1)}_{i,k,k}(v^{(3)}_{i,k}, v^{(4)}_{i,k}) \Big) \Big)$$

where we define, for $i \in [1,N]$, $k, k' \in [1,\mu]$, and $u, u' \in \mathbb{F}_{p^\dagger}$,

$$\mathsf{Chk}^{(0)}_{i,k,k'}(u,u') \overset{\mathrm{def}}{=} \big(u \boxdot^{\dagger\langle 1\rangle} (\widetilde{c}_{i,0,k} \boxminus^{\dagger\langle 1\rangle} \alpha_{i,0,k})\big) \boxplus^{\dagger\langle 1\rangle} \big(u' \boxdot^{\dagger\langle 1\rangle} \widetilde{c}_{i,1,k'}\big) \ ,$$
$$\mathsf{Chk}^{(1)}_{i,k,k'}(u,u') \overset{\mathrm{def}}{=} \big(u \boxdot^{\dagger\langle 1\rangle} \widetilde{c}_{i,0,k}\big) \boxplus^{\dagger\langle 1\rangle} \big(u' \boxdot^{\dagger\langle 1\rangle} (\widetilde{c}_{i,1,k'} \boxminus^{\dagger\langle 1\rangle} \alpha_{i,1,k'})\big) \ .$$

(c) Choose $w^{(0)}_{i,k}, \dots, w^{(4)}_{i,k} \xleftarrow{R} \mathbb{F}_{p^\dagger}$ for each $i \in [1,N]$ and $k \in [2,\mu]$, and set

$$c^\dagger_{\mathrm{flag},3} \leftarrow \boxplus^{\dagger\langle 2\rangle}_{\substack{i\in[1,N]\\ k\in[2,\mu]}} \Big(w^{(0)}_{i,k} \boxdot^{\dagger\langle 2\rangle} \Big(\mathsf{Chk}^{(0)}_{i,1,k}(w^{(1)}_{i,k}, w^{(2)}_{i,k}) \boxtimes^{\dagger\langle 1\rangle} \mathsf{Chk}^{(1)}_{i,1,k}(w^{(3)}_{i,k}, w^{(4)}_{i,k}) \Big) \Big) \ .$$

(d) Set $c^\dagger_{\mathrm{flag}} \leftarrow c^\dagger_{\mathrm{flag},1} \boxplus^{\dagger\langle 2\rangle} c^\dagger_{\mathrm{flag},2} \boxplus^{\dagger\langle 2\rangle} c^\dagger_{\mathrm{flag},3}$.

(e) For $h \in [1,\nu]$, choose $\alpha_{\mathrm{chk},h} \xleftarrow{R} \mathcal{M}^{\dagger\langle 2\rangle}_{\mathrm{eff}}$ and $\gamma_{\mathrm{chk},h} \xleftarrow{R} \mathbb{F}_{p^\dagger}$, and set

$$\overline{c}^\dagger_{\mathrm{chk},h} \leftarrow \alpha_{\mathrm{chk},h} \boxplus^{\dagger\langle 2\rangle} (\gamma_{\mathrm{chk},h} \boxdot^{\dagger\langle 2\rangle} c^\dagger_{\mathrm{flag}})$$

and $c^\dagger_{\mathrm{chk},h} \leftarrow \mathsf{Rnd}^{\dagger\langle 2\rangle}(\overline{c}^\dagger_{\mathrm{chk},h})$. Then send $c^\dagger_{\mathrm{chk},1}, \dots, c^\dagger_{\mathrm{chk},\nu}$ to Client.

4. [Client $\to$ Server (2)] Given the $c^\dagger_{\mathrm{chk},1}, \dots, c^\dagger_{\mathrm{chk},\nu}$, for each $h \in [1,\nu]$, Client computes $m_{\mathrm{chk},h} \leftarrow \mathsf{Dec}^{\dagger\langle 2\rangle}_{\mathsf{sk}^\dagger}(c^\dagger_{\mathrm{chk},h})$, and aborts the protocol if $m_{\mathrm{chk},h} = \perp$. Otherwise, Client sends $m_{\mathrm{chk},1}, \dots, m_{\mathrm{chk},\nu}$ to Server.

5. [Server's output] For the message $m_{\mathrm{chk},1}, \dots, m_{\mathrm{chk},\nu}$ sent from Client, Server first checks if $m_{\mathrm{chk},h} = \alpha_{\mathrm{chk},h}$ for all $h \in [1,\nu]$. If this is not satisfied, then Server aborts the protocol. Otherwise, for each $i \in [1,N]$ and $h \in [1,L_i]$, Server sets

$$\overline{c}^\dagger_{i,h} \leftarrow \boxplus^{\dagger\langle 1\rangle}_{j=0,1}((\alpha_{i,j,1}{}^{-1}\varphi_{i,h}(j)) \boxdot^{\dagger\langle 1\rangle} \widetilde{c}_{i,j,1})$$

where the inverse of $\alpha_{i,j,1}$ is taken modulo $p^\dagger$, and $c^\dagger_{i,h} \leftarrow \mathsf{Rnd}^{\dagger\langle 1\rangle}(\overline{c}^\dagger_{i,h})$. Then Server outputs all the $c^\dagger_{i,h}$'s.

We analyze the behavior of this protocol. The main difference of the protocol from the protocol in Section 5.2 is at the construction of $c^\dagger_{\mathrm{flag}}$. It is intended that, if Client (as well as Server) honestly executes the protocol then $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}] = 0$; while if Client does not choose the plaintexts in Step 2 honestly then we will have $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}] = 0$ only with a sufficiently small probability.

First, we focus on the construction of $c^\dagger_{\mathrm{flag},1}$. The argument in Section 5.2 implies that $\mathsf{Pt}[\mathsf{Val}(\widetilde{c}_{i,j,k})] = 0$ if and only if $\widetilde{c}_{i,j,k}$ is a valid level-1 ciphertext. Hence, $\mathsf{Pt}[c^\dagger_{\mathrm{flag},1}] = 0$ if all $\widetilde{c}_{i,j,k}$ are valid. On the other hand, if some $\widetilde{c}_{i,j,k}$ is invalid, then the plaintext for $u_{i,j,k} \boxdot^{\dagger\langle 2\rangle} \mathsf{Val}(\widetilde{c}_{i,j,k})$ becomes uniformly at random over $\mathbb{F}_{p^\dagger}$, therefore $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}]$ becomes uniformly random as well (whatever the plaintexts for $c^\dagger_{\mathrm{flag},2}$ and $c^\dagger_{\mathrm{flag},3}$ are). In this case, the conditional probability that $\mathsf{Pt}[c^\dagger_{\mathrm{flag}}] = 0$ is $1/p^\dagger$.

From now, we suppose that all ciphertexts $\widetilde{c}_{i,j,k}$ are valid. We focus on the components in the constructions of $c^\dagger_{\mathrm{flag},2}$ and $c^\dagger_{\mathrm{flag},3}$ relevant to any fixed index $i \in [1,N]$. Write $\widetilde{m}_{i,j,k} = \mathsf{Pt}[\widetilde{c}_{i,j,k}]$. We prepare the following lemma:

Lemma 13. *In the current situation, the following two conditions are equivalent:*

1. *Either* $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (\alpha_{i,0,k}, 0)$ *for any* $k \in [1, \mu]$*, or* $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (0, \alpha_{i,1,k})$ *for any* $k \in [1, \mu]$*.*
2. *Both of the following properties hold:*
   (a) *For each* $k \in [1, \mu]$*, we have* $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (\alpha_{i,0,k}, 0)$ *or* $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (0, \alpha_{i,1,k})$*.*
   (b) *For each* $k \in [2, \mu]$*, we have* $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,k}) = (\alpha_{i,0,1}, 0)$ *or* $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,k}) = (0, \alpha_{i,1,k})$*.*

Proof. It is straightforward to verify that Condition 1 implies the two properties in Condition 2. We consider the converse direction. By Condition 2a for $k = 1$, we have either $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,1}) = (\alpha_{i,0,1}, 0)$ or $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,1}) = (0, \alpha_{i,1,1})$ (we note that at most one of them is satisfied, since the values $\alpha_{i,j,k}$ are non-zero). Now if $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,1}) = (\alpha_{i,0,1}, 0)$, then Condition 2b implies that $\widetilde{m}_{i,1,k} = 0$ for any $k \in [2, \mu]$, and then Condition 2a implies that $\widetilde{m}_{i,0,k} = \alpha_{i,0,k}$ for any $k \in [2, \mu]$. Hence it is in the former case of Condition 1. Similarly, if $(\widetilde{m}_{i,0,1}, \widetilde{m}_{i,1,1}) = (0, \alpha_{i,1,1})$, then Condition 2b implies that $\widetilde{m}_{i,1,k} = \alpha_{i,1,k}$ for any $k \in [2, \mu]$, and then Condition 2a implies that $\widetilde{m}_{i,0,k} = 0$ for any $k \in [2, \mu]$. Hence it is in the latter case of Condition 1. Therefore Lemma 13 holds. □

Based on Lemma 13, we evaluate the probability that $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ when ($\widetilde{c}_{i,j,k}$ are valid ciphertexts but) Client did not honestly choose the plaintexts $\widetilde{m}_{i,j,k}$ for $\widetilde{c}_{i,j,k}$ in Step 2 of the protocol. First we note that, for any index $i \in [1, N]$, if $\mathsf{Pt}[c_i] = 0$, then there is an index $j_0 \in \{0, 1\}$ with the property that $\overline{c}_{i,j_0,k}$ is a uniformly random ciphertext of $\alpha_{i,0,k} \neq 0$ and $\overline{c}_{i,1-j_0,k}$ is a uniformly random ciphertext of a uniformly random plaintext for every $k \in [1, \mu]$. Now the condition $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (\alpha_{i,0,k}, 0)$ will be satisfied for every $k \in [1, \mu]$ if and only if Client honestly sets the plaintexts for $\overline{\overline{c}}_{i,j_0,k}$ and $\overline{\overline{c}}_{i,1-j_0,k}$ to be $\alpha_{i,0,k}$ and 0, respectively, for every $k \in [1, \mu]$. On the other hand, the condition $(\widetilde{m}_{i,0,k}, \widetilde{m}_{i,1,k}) = (0, \alpha_{i,1,k})$ will be satisfied only when Client sets the plaintext for $\overline{\overline{c}}_{i,1-j_0,k}$ to be $\alpha_{i,1,k}$. As the ciphertext $\overline{c}_{i,1-j_0,k}$ received by Client is independent of $\alpha_{i,1,k}$, Client can succeed the aforementioned choice for all of $\overline{\overline{c}}_{i,1-j_0,k}$ ($k \in [1, \mu]$) with probability at most $(|\mathcal{M}_{\text{eff}}| - 1)^{-\mu}$. The situation is similar in the other case where $\mathsf{Pt}[c_i] = 1$. Summarizing, when Client did not honestly choose the plaintexts in Step 2, the probability that Condition 1 in Lemma 13 is satisfied for any $i \in [1, N]$ with probability at most $(|\mathcal{M}_{\text{eff}}| - 1)^{-\mu}$.

From now, we consider the other case where Condition 1 in Lemma 13 (hence Condition 2 as well, by the lemma) is not satisfied for some $i \in [1, N]$. First we suppose that Condition 2a in the lemma is not satisfied for some $k \in [1, \mu]$. In this case, at least one of the two terms $\widetilde{c}_{i,0,k} \boxminus^{\dagger\langle 1\rangle} \alpha_{i,0,k}$ and $\widetilde{c}_{i,1,k}$ in $\mathsf{Chk}^{(0)}_{i,k,k}(v^{(1)}_{i,k}, v^{(2)}_{i,k})$ has a non-zero plaintext; as $v^{(1)}_{i,k}$ and $v^{(2)}_{i,k}$ are uniformly random over $\mathbb{F}_{p^{\dagger}}$, it follows that now the plaintext for $\mathsf{Chk}^{(0)}_{i,k,k}(v^{(1)}_{i,k}, v^{(2)}_{i,k})$ becomes 0 with probability $1/p^{\dagger}$. The same property also holds for $\mathsf{Chk}^{(1)}_{i,k,k}(v^{(3)}_{i,k}, v^{(4)}_{i,k})$. Moreover, when both $\mathsf{Chk}^{(0)}_{i,k,k}(v^{(1)}_{i,k}, v^{(2)}_{i,k})$ and $\mathsf{Chk}^{(1)}_{i,k,k}(v^{(3)}_{i,k}, v^{(4)}_{i,k})$ have non-zero plaintexts, the plaintext for the term

$$v^{(0)}_{i,k} \boxdot^{\dagger\langle 2\rangle} \left(\mathsf{Chk}^{(0)}_{i,k,k}(v^{(1)}_{i,k}, v^{(2)}_{i,k}) \boxtimes^{\dagger\langle 1\rangle} \mathsf{Chk}^{(1)}_{i,k,k}(v^{(3)}_{i,k}, v^{(4)}_{i,k})\right)$$

is uniformly random over $\mathbb{F}_{p^{\dagger}}$, therefore (whatever the other terms in $c^{\dagger}_{\text{flag},1}$, $c^{\dagger}_{\text{flag},2}$, and $c^{\dagger}_{\text{flag},3}$ are) $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ with probability $1/p^{\dagger}$. Hence, in the current case, the conditional probability that $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ is at most $1/p^{\dagger} + 1/p^{\dagger} + 1/p^{\dagger} = 3/p^{\dagger}$.

Similarly, when Condition 2b in the lemma is not satisfied for some $k \in [2, \mu]$, by focusing on the term

$$w^{(0)}_{i,k} \boxdot^{\dagger\langle 2\rangle} \left(\mathsf{Chk}^{(0)}_{i,1,k}(w^{(1)}_{i,k}, w^{(2)}_{i,k}) \boxtimes^{\dagger\langle 1\rangle} \mathsf{Chk}^{(1)}_{i,1,k}(w^{(3)}_{i,k}, w^{(4)}_{i,k})\right)$$

in $c^{\dagger}_{\text{flag},3}$ instead, it follows that the conditional probability that $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ is at most $3/p^{\dagger}$ as well. Therefore, when Condition 1 in Lemma 13 is not satisfied, the conditional probability that $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ is at most $3/p^{\dagger}$.

By the previous three paragraphs, when all ciphertexts $\widetilde{c}_{i,j,k}$ are valid but Client did not honestly choose the plaintexts, the conditional probability that $\mathsf{Pt}[c^{\dagger}_{\text{flag}}] = 0$ is at most $(|\mathcal{M}_{\text{eff}}| - 1)^{-\mu} + 3/p^{\dagger}$. This bound is larger than the bound for the conditional probability (evaluated above) for the case of invalid ciphertexts $\widetilde{c}_{i,j,k}$. Hence, the probability that a malicious Client can set $\mathsf{Pt}[c^{\dagger}_{\mathrm{flag}}] = 0$ is at most $(|\mathcal{M}_{\mathrm{eff}}| - 1)^{-\mu} + 3/p^{\dagger}$. Owing to the property, essentially the same arguments as Theorem 9 and Theorem 10 imply (by using the fact $|S_i| = 2$) the following results.

Theorem 14. *Suppose that both Server and Client honestly execute the protocol. Then the protocol is aborted with probability at most $\varepsilon_{\mathrm{cor}} = N\mu \cdot |\mathcal{M}_{\mathrm{eff}}|/p$. If the protocol is not aborted, then the output distribution for Server is identical to the distribution of the functionality $F(\mathsf{pp}, x_{\mathsf{S}}, x_{\mathsf{C}})$.*

Theorem 15. *Suppose that Server honestly executes the protocol. Let $\varepsilon_{\mathrm{sec}} = \varepsilon_{\mathrm{sec},1} + \varepsilon_{\mathrm{sec},2} + \varepsilon_{\mathrm{sec},3}$, where*

$$\varepsilon_{\mathrm{sec},1} \overset{\mathrm{def}}{=} \frac{N\mu \cdot |\mathcal{M}_{\mathrm{eff}}|}{p} \ , \ \varepsilon_{\mathrm{sec},2} \overset{\mathrm{def}}{=} \frac{1}{|\mathcal{M}^{\dagger}_{\mathrm{eff}}|^{\nu}} \ , \ \varepsilon_{\mathrm{sec},3} \overset{\mathrm{def}}{=} \frac{1}{(|\mathcal{M}_{\mathrm{eff}}| - 1)^{\mu}} + \frac{3}{p^{\dagger}} \ .$$

Then our protocol is secure against malicious Client with statistical distance at most $\varepsilon_{\mathrm{sec}}$ (see Definition 4 for the terminology).

In this protocol, the number (two) of communication rounds is the same as our protocol in Section 5.2. In the first round, Server sends and receives $2N\mu + \mu$ ciphertexts in the protocol in Section 5.2 (specialized to the current case $S_i = \{0, 1\}$), while the number is reduced to $2N\mu$ (i.e., dummy ciphertexts $c_{\perp,k}$ are not used) in the current protocol. Moreover, the term $\varepsilon_{\mathrm{sec},4}$, which was dominant (when N is not very large) in the bound for the statistical distance in Theorem 10, has been removed from the bound in the current Theorem 15; as a result, the parameter μ can be smaller than the case of Section 5.2. This shows that, from the viewpoint of communication costs, the current protocol is more efficient than the one in Section 5.2.

6 Extensions to Two-Input Functions

Our proposed protocols natively support univariate (one-input) functions $\varphi(x) = \varphi_{i,h}(x)$ only. In this section, we investigate some possible ways of extending our protocols to two-input functions $\varphi(x, y)$ for encrypted values x, y held by Server. Here we suppose that $x \in I_x$ and $y \in I_y$ for some known subsets $I_x, I_y \subseteq \mathcal{M}$. We write $[[z]]$ to denote a ciphertext with plaintext z.

First we consider an easy case where, there exist a univariate function $\psi(z)$ and constants α, β, γ for which we have

$$\varphi(x, y) = \psi(\alpha x + \beta y + \gamma) \text{ for any } x \in I_x \text{ and } y \in I_y \ . \tag{9}$$

In this case, given ciphertexts $[[x]]$ and $[[y]]$, Server can homomorphically generate $[[\alpha x + \beta y + \gamma]]$ and then obtain $[[\psi(\alpha x + \beta y + \gamma)]] = [[\varphi(x, y)]]$ by applying our proposed protocol. A simple but useful application of this technique is secure comparison of encrypted integers x, y; Server wants to obtain $[[\chi]]$ where $\chi = 1$ if $x \geq y$ and $\chi = 0$ if $x < y$. This is achieved by using the relation $\chi = \mathsf{sign}(x - y)$ where sign is defined by $\mathsf{sign}(z) \overset{\mathrm{def}}{=} 1$ if $z \geq 0$ and $\mathsf{sign}(z) \overset{\mathrm{def}}{=} 0$ if $z < 0$.

The technique above also yields a generic (though not very efficient) solution for this problem. Namely, we suppose (by shifting the domain I_y of y if necessary) that $I_y \subseteq [0, K - 1]$ for an integer K. Then Server can obtain $[[\varphi(x, y)]]$ by using the relation $\varphi(x, y) = \psi(K \cdot x + y)$ for $x \in I_x$ and $y \in I_y$ where ψ is defined by $\psi(z) \overset{\mathrm{def}}{=} \varphi(\lfloor z/K \rfloor, z \bmod K)$.

Besides the simplest case in Eq.(9), some other two-input functions can be evaluated more efficiently than the generic case in the last paragraph. For example, suppose that $x \in [0, N_x - 1]$ and $y \in [0, N_y - 1]$ for some integers N_x and N_y. Then Server can obtain a ciphertext $[[xy]]$ for the product of x and y from ciphertexts $[[x]]$ and $[[y]]$ in the following manner:

1. Server homomorphically generates $[[x + y]]$.
2. From $[[x]]$, $[[y]]$, and $[[x + y]]$, Server obtains $[[x^2]]$, $[[y^2]]$, and $[[(x + y)^2]]$ simultaneously by using our proposed protocol.

3. Finally, Server homomorphically computes

$$2^{-1} \boxdot ([[(x+y)^2]] \boxminus [[x^2]] \boxminus [[y^2]]) = [[xy]]$$

where the inverse of 2 is taken in the plaintext space $\mathbb{F}_p$.

As $x+y \in [0, N_x+N_y-2]$, this procedure has to handle only $N_x+N_y+(N_x+N_y-1) = 2N_x+2N_y-1$ ciphertexts (for semi-honest Client) for our proposed protocol, while the generic technique above has to handle N_xN_y ciphertexts. Therefore, the technique here is much more efficient than the generic case.

When a 2LHE scheme can be used as the scheme $\Pi^\dagger$, another strategy also works as follows. Given ciphertexts $[[x]]$ and $[[y]]$ for the scheme Π, Server first obtains (by using our protocol) a level-1 ciphertext $[[\delta(x,a)]]^{\dagger\langle 1\rangle}$ for each $a \in I_x$ and a level-1 ciphertext $[[\varphi(a',y)]]^{\dagger\langle 1\rangle}$ for each $a' \in I_x$ in the scheme $\Pi^\dagger$, where $\delta(x,a)$ denotes the Kronecker Delta (i.e., it takes 1 if $x=a$ and takes 0 if $x \neq a$). Then Server can locally compute the following:

$$\begin{aligned} \boxplus^{\dagger\langle 2\rangle}_{a\in I_x}\left([[\delta(x,a)]]^{\dagger\langle 1\rangle} \boxtimes^{\dagger\langle 1\rangle} [[\varphi(a,y)]]^{\dagger\langle 1\rangle}\right) &= [[\sum_{a\in I_x} \delta(x,a)\cdot\varphi(a,y)]]^{\dagger\langle 2\rangle} \\ &= [[\varphi(x,y)]]^{\dagger\langle 2\rangle} \ . \end{aligned}$$

We also note that, when the values $\varphi(x,y)$ of φ are not too large, from the two ciphertexts $[[\delta(x,a)]]^\dagger$ and $[[\varphi(a,y)]]^\dagger$ appeared in the last paragraph, Server can also obtain $[[\delta(x,a)\cdot\varphi(a,y)]]^\dagger$ by using the techniques described above without homomorphic multiplication functionality for $\Pi^\dagger$. Hence, by homomorphically adding them, Server can obtain the $[[\varphi(x,y)]]^\dagger$.

7 Experimental Results

7.1 Secure Computation of Exact Edit Distance

Based on our proposed protocol (against semi-honest Client), we implemented a secure protocol to compute the edit distance between two (character-wise) encrypted strings held by Server. Our computation is based on the standard technique using dynamic programming; that is, given two strings $a = a_1a_2\cdots a_L$ and $b = b_1b_2\cdots b_L$ (of equal lengths), we recursively construct a matrix $(D[i][j])_{i,j}$ in a way that $D[0][i] = D[i][0] = i$ and

$$D[i][j] = \min\{D[i-1][j]+1, D[i][j-1]+1, D[i-1][j-1]+e(i,j)\} \qquad (10)$$

where $e(i,j) = 1$ if $a_i \neq b_j$ and $e(i,j) = 0$ if $a_i = b_j$. Then the value of $D[L][L]$ becomes the edit distance between a and b. Now note that, when each character is encoded by an integer, $e(i,j)$ is in fact a univariate function of $a_i - b_j$, therefore the value of $e(i,j)$ can be securely computed from ciphertexts $[[a_i]]$ and $[[b_j]]$ by using our proposed protocol. On the other hand, for integers A, B, and C, we have $\min\{A,B,C\} = \min\{\min\{A,B\},C\}$ and $\min\{A,B\} = A - \mathsf{ReLU}(A-B)$ where $\mathsf{ReLU}(x) \overset{\text{def}}{=} \max\{0,x\}$. Hence the value of Eq.(10) can also be securely computed by a combination of our proposed protocol and the additively-homomorphic functionality of the underlying cryptosystem. Moreover, it is known that $D[i-1][j]-D[i][j-1] \in [-2,2]$, $D[i-1][j]-D[i-1][j-1] \in [-1,1]$, and $D[i][j-1]-D[i-1][j-1] \in [-1,1]$. Therefore, the encrypted input values for ReLU to compute Eq.(10) are all within a significantly small range, which makes our protocol more efficient.

Table 2 shows the experimental results on the execution times for our protocol of edit distance computation. Here, the string lengths L varied as $L = 128$, 256, 512, and 1024. Our protocol is based on the elliptic-curve lifted-ElGamal cryptosystem using the standard elliptic curve secp256k1 with 128-bit security. Our experiments were performed over a real network environment (instead of using the value of theoretically simulated network specifications) and mainly used Wide Area Network (WAN) rather than Local Area Network (LAN). In our experiments over WAN, we used MacBook Pro Core i5 2.3GHz and a desktop PC Core i7 8700 3.2GHz for implementing Client and Server, respectively.

Table 2: Experimental results on online total execution times (sec.) for secure two-party computation of exact edit distance (here "SS" and "HE" stand for Secret Sharing and Homomorphic Encryption, respectively)

Protocols	Primitive	Network	String Length L			
			128	256	512	1024
[19], §V-C	SS	WAN (simulated)	41.8	84.9	174.5	367.0
Ours	HE	WAN (real)	8.68	24.36	58.38	203.41
		LAN (real)	1.81	4.27	13.22	44.72

Table 3: Experimental results on total execution times (sec.) for secure conversion from level-2 into level-1 ciphertexts in 2LHE scheme

Network	Plaintext Bit-Length $\log_2 \lvert S \rvert$								
(real)	8	9	10	11	12	13	14	15	16
LAN	0.08	0.10	0.13	0.24	0.35	0.57	1.11	1.98	3.74
WAN	0.23	0.39	0.68	1.52	2.18	3.89	8.02	16.16	32.89

Table 2 also shows a comparison with previous work. Here we emphasize that our protocol is for computing *exact* edit distance, while most of the previous results dealt with *approximate* edit distance in order to improve the efficiency. Moreover, many existing results supposed to use *LAN*, in contrast to our present work using *WAN*. Among such situations, in a recent paper [19], experimental results on secure two-party computation of the *exact* edit distance over *WAN* are reported. The values shown in Table 2 are quoted from Table VI of that paper. The protocol in [19] is based on the 2-out-of-2 secret sharing (SS); and the estimated execution times in that paper are based on theoretical simulation of network environment (instead of using a real network) where it is noted that they assumed WAN environment in their simulation. Although a fair comparison between our result and theirs is difficult due to the difference of computer/network environments, we can still say that our experimental results on the protocol execution times are at least comparable to their results (and it might be considerable for some practical applications). We also note that, the protocol in [19] adopted the so-called client-aided model for SS-based MPC where, in addition to the execution times in Table 2, some other party also performs somewhat heavy pre-computation (e.g., 2262 sec. for $L = 1024$) to provide certain auxiliary inputs for the protocol. In contrast, our protocol here does not need any such heavy pre-computation.

As a reference, Table 2 also shows our experimental results using a real LAN environment instead of WAN, where Client and Server were both implemented by using Xeon Platinum 8280 2.7GHz.

7.2 Level Reduction for Ciphertexts in 2LHE Schemes

As mentioned at the end of Section 3.1, our proposed protocol can be used for securely converting a level-2 ciphertext for a 2LHE scheme into a level-1 ciphertext with the same plaintext by using just one-round computation (for semi-honest Client; and two-rounds for malicious Client). Of course, there seems no hope to achieve such a "level reduction" functionality by a stand-alone use of 2LHE schemes without fully (or somewhat) homomorphic property. We also implemented this level reduction protocol (with semi-honest Client). Table 3 shows the experimental results on the execution times of our protocol. Here, the sizes of the plaintext range $|S|$ for the original level-2 ciphertext held by Server varied from 2^8 to 2^{16}. We used the 2LHE scheme by Attrapadung et al. [3]; precisely, we used a publicly available library [17] for this 2LHE scheme based on BLS12-381 curves with 128-bit security, and developed the interfaces by C++ (Clang 7.0). The computer and the network environments for both LAN and WAN settings are the same as our experiment in Section 7.1. We note that, our protocol with semi-honest Client is correctly executable as long as the subset $\mathcal{M}_{\mathrm{eff}}$ of effective plaintexts satisfies $0 \in \mathcal{M}_{\mathrm{eff}}$; here we set $\mathcal{M}_{\mathrm{eff}} = \{0\}$ for the sake of efficiency.

The experimental results show that the time complexity of the implemented protocol is almost proportional to $|S|$ (especially for larger $|S|$), hence exponential in the bit length $\log_2 |S|$, which is consistent to the protocol construction. Although our protocol is not very practical when $|S|$ is large, it is significantly efficient for smaller $|S|$; e.g., when $|S| = 2^8$, it took only 0.23 sec. even in WAN environments. We also emphasize that, such efficient results are still obtained even though the 2LHE scheme used here is pairing-based and hence is less efficient than the lifted-ElGamal cryptosystem.

8 Computing over ABE Ciphertexts

Recall that our proposed protocols in this paper are applicable to any underlying encryption scheme that has all necessary properties common to the lifted-ElGamal cryptosystem (Definition 1). In this section, we explore a further extension of our protocols to MPC over ciphertexts of attribute-based encryption (ABE) schemes. The key observation is that, many pairing-based ABE schemes in the literature have ciphertext structures similar to the ElGamal cryptosystem, therefore such schemes can be equipped with the additively homomorphic functionality by slight modifications (as long as the ciphertexts have some linear structure in their exponents). The resulting schemes, which we call *additively homomorphic ABE* (*AH-ABE*) *schemes*, can then be used as the underlying scheme for our protocols (with some necessary adjustment to ABE settings).

Here one may be curious that, while MPC usually supposes a situation where no fully trusted party exists (otherwise such a party might be able to generate all parties' outputs with trust), the design of ABE frequently assumes a practical system operation where each user's secret key is generated by a trusted third party (TTP) having the master secret key. For this viewpoint we note that, even if there is a TTP to generate secret keys, it is practically not obvious that this TTP can also work with MPC between some users by the following reasons. First, when the TTP is a kind of official key issuance center, the center may not be given the authority to do other work such as the support to users' MPC. Secondly, in contrast to the secret key generation which might be not frequently requested per each user, MPC between users may occur very frequently and ubiquitously, therefore resolving all demands for MPC will easily exceed the capacity of just the single TTP. Hence it is certainly meaningful in practice to combine MPC with ABE.

8.1 Additively Homomorphic ABE: Definition

We first briefly describe the standard ABE definition for generic predicates (see e.g. [2]), followed by our definition for AH-ABE. In the following, for a PPT algorithm $\mathcal{A}$, we denote by $[\mathcal{A}(x)]$ the set of all possible outputs of $\mathcal{A}(x)$.

Predicate Family. Let $P = \{\, P_\kappa \colon \mathbb{X}_\kappa \times \mathbb{Y}_\kappa \to \{0,1\} \mid \kappa \in \mathcal{K} \,\}$ be a predicate family where $\mathbb{X}_\kappa$ and $\mathbb{Y}_\kappa$ denote "key attribute" and "ciphertext attribute" spaces, respectively. The index or "parameter" κ denotes a list of some parameters such as the universes of attributes and/or bounds on some quantities, hence its domain $\mathcal{K}$ will depend on that predicate. We will often omit κ when the context is clear.

Definition 16. An *ABE scheme* Π for predicate P consists of PPT algorithms Setup, Gen, Enc, Dec with the following properties.

- Given security parameter 1^λ and an index $\kappa \in \mathcal{K}$, the setup algorithm Setup generates a pair of a public key pk and a master secret key msk; $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \kappa)$.
- Given msk and a key attribute $x \in \mathbb{X}_\kappa$, the key generation algorithm Gen generates a secret key sk; $\mathsf{sk} \leftarrow \mathsf{Gen}(\mathsf{msk}, x)$. We suppose that sk contains x.
- Given pk, a ciphertext attribute $y \in \mathbb{Y}_\kappa$, and a plaintext $m \in \mathcal{M}$, the encryption algorithm Enc generates a ciphertext c; $c \leftarrow \mathsf{Enc}_{\mathsf{pk},y}(m)$. We suppose that c contains y.
- Given sk associated to $x \in \mathbb{X}_\kappa$ and a ciphertext c, the decryption algorithm Dec outputs either a plaintext $\tilde{m} \in \mathcal{M}$ or $\perp$; $\tilde{m}$ or $\perp \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$.

The standard correctness is as follows: for all $\lambda \geq 1$, $\kappa \in \mathcal{K}$, $(\mathsf{pk}, \mathsf{msk}) \in [\mathsf{Setup}(1^\lambda, \kappa)]$, $x \in \mathbb{X}_\kappa$, $y \in \mathbb{Y}_\kappa$, $\mathsf{sk} \in [\mathsf{Gen}(\mathsf{msk}, x)]$, $m \in \mathcal{M}$, $c \in [\mathsf{Enc}_{\mathsf{pk},y}(m)]$, if $P(x, y) = 1$, then $\mathsf{Dec}_{\mathsf{sk}}(c) = m$. We consider the standard security notions, namely, selective or adaptive IND-CPA; we refer to e.g. [2]. We will also require the following *public verifiability* as defined in [32], in order to deal with malicious Client where validity of ciphertexts have to be checked by Server.

Definition 17. We say that an ABE scheme Π for predicate P is *publicly verifiable*, if there exists a PPT algorithm Verify that takes as inputs a public key pk, a possible ciphertext $c \in \{0,1\}^*$, a ciphertext attribute $y \in \mathbb{Y}_\kappa$, and outputs 0 or 1 as follows: for all $\lambda \geq 1$, $\kappa \in \mathcal{K}$, $(\mathsf{pk}, \mathsf{msk}) \in [\mathsf{Setup}(1^\lambda, \kappa)]$, $y \in \mathbb{Y}_\kappa$, and $c \in \{0,1\}^*$, we have

$$\mathsf{Verify}(\mathsf{pk}, c, y) = 1 \text{ if and only if } c \in [\mathsf{Enc}_{\mathsf{pk},y}(m)] \text{ for some } m \in \mathcal{M}.$$

We also say that such a c is a valid ciphertext with attribute y.

Definition 18. An *AH-ABE* Π for predicate P is a publicly verifiable ABE scheme albeit with the correctness below with a set of effectively decryptable plaintexts $\mathcal{M}_{\mathrm{eff}} \subseteq \mathcal{M}$, together with an additional PPT algorithm Rnd, and polynomial-time computable operators $\boxplus$, $\boxminus$ as follows:

- Given pk and a valid ciphertext $c \in [\mathsf{Enc}_{\mathsf{pk},y}(m)]$, the rerandomization algorithm Rnd outputs a uniformly random $\tilde{c} \in [\mathsf{Enc}_{\mathsf{pk},y}(m)]$; $\tilde{c} \leftarrow \mathsf{Rnd}_{\mathsf{pk}}(c)$.
- Given pk, $c_1 \in [\mathsf{Enc}_{\mathsf{pk},y}(m_1)]$, and $c_2 \in [\mathsf{Enc}_{\mathsf{pk},y}(m_2)]$ with a common attribute y, the homomorphic addition $c_1 \boxplus_{\mathsf{pk}} c_2$ yields an element of $[\mathsf{Enc}_{\mathsf{pk},y}(m_1 + m_2)]$.
- Given pk and $c \in [\mathsf{Enc}_{\mathsf{pk},y}(m)]$, the homomorphic negation $\boxminus_{\mathsf{pk}} c$ yields an element of $[\mathsf{Enc}_{\mathsf{pk},y}(-m)]$.

The correctness condition is as follows: for any $m \in \mathcal{M}$, a uniformly random ciphertext $c \in [\mathsf{Enc}_{\mathsf{pk},y}(m)]$ satisfies the following except for negligible probability:

$$\mathsf{Dec}_{\mathsf{sk}}(c) = \begin{cases} m & \text{if } m \in \mathcal{M}_{\mathrm{eff}} \text{ and } P(x,y) = 1 \\ \perp & \text{otherwise.} \end{cases},$$

8.2 Protocols for Computing over AH-ABE: Settings

Due to the homomorphic functionality and the correctness condition regarding effective plaintexts for AH-ABE mentioned above, our proposed protocol in Section 3 with semi-honest Client is extended to the current case of AH-ABE. The points to be taken care of are the followings. In the protocol, the attributes for keys and ciphertexts are regarded as non-secret information. The key pair $(\mathsf{pk}, \mathsf{sk})$ for Server's input ciphertext should satisfy the relation $P(x, y) = 1$. Moreover, the error probability of the protocol is slightly increased from the original due to the non-perfect correctness in the formulation of AH-ABE, but the increase is negligible as well as the error probability for the underlying AH-ABE schemes.

For the sake of simplicity, here we only consider the case of semi-honest Client. When considering malicious Client, the case of a single input ciphertext (i.e., $N = 1$) has no special difficulty owing to the public verifiability of AH-ABE. On the other hand, in the case of batch execution for $N \geq 2$ input ciphertexts with *non-identical attributes*, a new difficulty arises as true ciphertexts and dummy ciphertexts generated by Server *with different attributes* can be easily distinguished by Client. To avoid this issue, a batch execution has to be performed only for a set of ciphertexts with the same attribute.

A concrete description of the current setting is as follows. Let Π and $\Pi^\dagger$ be AH-ABE schemes for predicates P and $P^\dagger$, respectively, and let $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}^\dagger, \mathsf{sk}^\dagger)$ be their key pairs. We suppose that the plaintext spaces are $\mathcal{M} = \mathbb{F}_p$ and $\mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$ for publicly known primes p and $p^\dagger$, respectively. Let $\mathcal{M}_{\mathrm{eff}} \subseteq \mathcal{M}$ be the subset of effective plaintexts for Π. We suppose that $0 \in \mathcal{M}_{\mathrm{eff}}$. Moreover, as mentioned above, we do not consider attribute hiding, which is inherited from the similar setting for the underlying AH-ABE. Hence, we simply let the attributes of Server's input and output ciphertexts be included in the common information. Now two parties' inputs are specified as follows:

Server's local input: Server has a valid ciphertext c with ciphertext attribute y in the scheme Π associated to the key pk, where the plaintext for c is supposed to be in a given subset $S \subseteq \mathcal{M} = \mathbb{F}_p$. Server also has a polynomially bounded number of functions $\varphi_h \colon S \to \mathcal{M}^\dagger = \mathbb{F}_{p^\dagger}$ with $h \in [1, L]$.

Client's local input: Client has the secret key sk associated to key attribute x in the scheme Π. This attribute must satisfy $P(x, y) = 1$.

Common information: Both parties know pk, $\mathsf{pk}^\dagger$ (including their indices κ, $\kappa^\dagger$), x, y, the cardinality $|S|$ of the subset S, and the attribute $y^\dagger \in \mathbb{Y}^\dagger_{\kappa^\dagger}$ of Server's output ciphertexts. Note that $|S|$ must be polynomially bounded.

Our protocol has one-sided output for Server, i.e., Client outputs nothing. Let pp, x_S, and x_C be the common information, Server's local input, and Client's local input, respectively. The functionality $F(\mathsf{pp}, x_\mathsf{S}, x_\mathsf{C})$ for Server's output is defined as a list of uniformly random valid ciphertexts $c_h^\dagger \in [\mathsf{Enc}^\dagger_{\mathsf{pk}^\dagger, y^\dagger}(\varphi_h(m))]$ for $h = [1, L]$.

8.3 Our Protocol for Computing over ABE: Construction

Our protocol for computing over AH-ABE (with semi-honest Client) follows exactly the same procedure as the protocol with AHE in Section 3.2. The difference is only that ciphertext/key attributes are also taken into account in the description. A concrete description of the protocol is as follows:

1. [Server $\to$ Client] Server executes the following:
   (a) For $j \in S$, choose $\gamma_j \xleftarrow{R} \mathbb{F}_p$ and set $c_j \leftarrow \mathsf{Rnd}\big((\gamma_j \boxdot c) \boxminus \mathsf{Enc}_{\mathsf{pk},y}(\gamma_j \cdot j)\big)$.
   (b) Send, to Client, all the ciphertexts c_j in a uniformly random order.
2. [Client $\to$ Server] Let $\overline{c}_1, \dots, \overline{c}_{|S|}$ denote the ciphertexts for Π sent from Server at the previous step. Client executes the following:
   (a) For each $i \in [1, |S|]$, compute $\overline{m}_i \leftarrow \mathsf{Dec}_\mathsf{sk}(\overline{c}_i)$. If the number of $i \in [1, |S|]$ satisfying $\overline{m}_i = 0$ is not equal to 1, then abort the protocol.
   (b) For each $i \in [1, |S|]$, set
   $$\overline{\overline{c}}_i \leftarrow \begin{cases} \mathsf{Enc}^\dagger_{\mathsf{pk}^\dagger, y^\dagger}(1) & \text{if } \overline{m}_i = 0 \ , \\ \mathsf{Enc}^\dagger_{\mathsf{pk}^\dagger, y^\dagger}(0) & \text{if } \overline{m}_i \neq 0 \ . \end{cases}$$
   (c) Send $\overline{\overline{c}}_1, \dots, \overline{\overline{c}}_{|S|}$ to Server in this order.
3. [Server's output] Given the $|S|$ ciphertexts sent from Client at the previous step, Server first permutes the ciphertexts in the reverse way of Step 1b, yielding ciphertexts $\widetilde{c}_j$ ($j \in S$). Then Server executes the following:
   (a) For each $h \in [1, L]$, set $\overline{c}_h^\dagger \leftarrow \boxplus^\dagger_{j \in S}(\varphi_h(j) \boxdot^\dagger \widetilde{c}_j)$ and set $c_h^\dagger \leftarrow \mathsf{Rnd}^\dagger(\overline{c}_h^\dagger)$.
   (b) Then output the $c_h^\dagger$ for all $h \in [1, L]$.

Correctness. Correctness of the protocol above follows from essentially the same argument as Section 3.3 for the AHE case. The only obvious difference is that we use the correctness of AH-ABE here, instead of AHE as previously; consequently, the error probability is affected with negligible increase, as the correctness condition for AH-ABE allows negligible decryption error probability.

Security. Security also follows from the argument in Section 3.4; that is, security against Client is information-theoretic and statistical, while security against Server is computational and is based directly on IND-CPA of ABE here. Note that, for the latter, either selectively or adaptively secure ABE is fine, as the protocol setting above do not consider key queries by Server.

8.4 Constructions for AH-ABE

First we give a construction of an AH-ABE scheme with equality predicate ($P(x,y)=1$ if and only if $x=y$); i.e., the setting of identity-based encryption (IBE). We start with the Boneh–Boyen IBE [5] already having an ElGamal-like structure, and simply define its "lifted" version. We set $\mathbb{Y}=\mathbb{X}=\{0,1\}^*$. Let $e\colon \mathbb{G}\times\mathbb{G}\to\mathbb{G}_{\mathrm{T}}$ be a symmetric bilinear pairing over cyclic groups $\mathbb{G}=\langle g\rangle$ and $\mathbb{G}_{\mathrm{T}}$ of exponentially large prime order p. Let $T=e(g,g)$. The scheme is described as follows, where $\mathcal{M}=\mathbb{F}_p$ and $\mathcal{M}_{\text{eff}}$ is a subset of $\mathcal{M}$ with only polynomially many elements:

- $\mathsf{Setup}(1^\lambda)$: Pick $\alpha, b_1, b_2 \xleftarrow{R} \mathbb{F}_p$. Set $\mathsf{msk}\leftarrow\alpha$ and $\mathsf{pk}\leftarrow(g,g^{b_1},g^{b_2},T^\alpha)$.
- $\mathsf{Gen}(\mathsf{msk},x)$: Pick $r\xleftarrow{R}\mathbb{F}_p$. Set $d_1=g^r$, $d_2=g^{\alpha+r(b_1+b_2x)}$. Output $\mathsf{sk}\leftarrow(d_1,d_2)$.
- $\mathsf{Enc}_{\mathsf{pk},y}(m)$: Pick $s\xleftarrow{R}\mathbb{F}_p$. Set $c_0=T^m(T^\alpha)^s$, $c_1=g^{s(b_1+b_2y)}$, and $c_2=g^s$. Output $c\leftarrow(c_0,c_1,c_2)$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Search for an $\tilde{m}\in\mathcal{M}_{\text{eff}}$ satisfying $T^{\tilde{m}}=c_0\cdot e(c_1,d_1)\cdot e(c_2,d_2)^{-1}$. If such an $\tilde{m}$ is found, then output $\tilde{m}$. Otherwise, output $\perp$.
- Homomorphic addition and negation can be done by

$$(c_0,c_1,c_2)\boxplus(c_0',c_1',c_2')=(c_0c_0',c_1c_1',c_2c_2')\ ,$$
$$\boxminus(c_0,c_1,c_2)=(c_0^{-1},c_1^{-1},c_2^{-1})\ .$$

- $\mathsf{Rnd}(c)$: Output $c\boxplus\mathsf{Enc}_{\mathsf{pk},y}(0)$.
- $\mathsf{Verify}(\mathsf{pk},c,y)$: Parse $c=(c_0,c_1,c_2)$. Check if $e(g,c_1)\overset{?}{=}e(g^{b_1+b_2y},c_2)$. If this holds, output 1. Otherwise, output 0.

The technique above for constructing an AH-ABE scheme (with equality predicate) from the Boneh–Boyen IBE is also extendible to many other ABE schemes (with various predicates) in the literature. More precisely, we provide a generic conversion from a class of pairing-based ABE constructions that admits a specific structure to AH-ABE schemes. In the following, for a vector $\vec{v}=(v_1,\dots,v_n)$, we denote $g^{\vec{v}}=(g^{v_1},\dots,g^{v_n})$. As usual, we use a symmetric bilinear pairing $e\colon\mathbb{G}\times\mathbb{G}\to\mathbb{G}_{\mathrm{T}}$ over cyclic groups $\mathbb{G}=\langle g\rangle$ and $\mathbb{G}_{\mathrm{T}}$ of prime order p. Let $T=e(g,g)$.

Base ABE Template. We assume a base ABE scheme with the following template construction. Let n be some integer.

- $\mathsf{Setup}(1^\lambda,\kappa)$: Pick $\alpha\xleftarrow{R}\mathbb{F}_p$ and $\vec{b}=(b_1,\dots,b_n)\xleftarrow{R}(\mathbb{F}_p)^n$. Set $\mathsf{msk}\leftarrow\alpha$ and $\mathsf{pk}\leftarrow(g,g^{\vec{b}},T^\alpha)$.
- $\mathsf{Enc}_{\mathsf{pk},y}(M)$: Let w be an integer that can depend on y. Pick a vector $\vec{s}=(s_0,s_1,\dots,s_w)\xleftarrow{R}(\mathbb{F}_p)^{w+1}$. Set

$$c_0=M\cdot(T^\alpha)^{s_0}\ ,\ c_1=g^{f_y(\vec{s},\vec{b})}\ ,\ c_2=g^{\vec{s}}\ ,$$

  where $f_y(\vec{s},\vec{b})$ is a vector of linear combinations of monomials s_ib_j for $i\in[0,w]$, $j\in[1,n]$, of which coefficients can depend on y. Note that these coefficients are publicly known. Then output $c\leftarrow(c_0,c_1,c_2)$.

Note that we do not pose any requirements for Gen and Dec for a base ABE scheme.

AH-ABE from Base ABE Template. From any ABE scheme Π in the form of the template construction described above, we can construct an AH-ABE scheme $\widetilde{\Pi}$ for the same predicate as follows. Algorithms Setup and Gen are exactly the same as those of Π. The other algorithms are as follows.

- $\widetilde{\Pi}.\mathsf{Enc}_{\mathsf{pk},y}(m)$: Output $\Pi.\mathsf{Enc}_{\mathsf{pk},y}(T^m)$.
- $\widetilde{\Pi}.\mathsf{Dec}_{\mathsf{sk}}(c)$: Search for an $\tilde{m} \in \mathcal{M}_{\mathrm{eff}}$ with $T^{\tilde{m}} = \Pi.\mathsf{Dec}_{\mathsf{sk}}(c)$. If such an $\tilde{m}$ is found, then output $\tilde{m}$. Otherwise, output $\perp$.
- Homomorphic addition in $\widetilde{\Pi}$ is done by
$$(c_0, c_1, c_2) \boxplus (c_0', c_1', c_2') = (c_0 \cdot c_0',\ c_1 \cdot c_1',\ c_2 \cdot c_2') \tag{11}$$
where $\cdot$ is the component-wise multiplication.
- Homomorphic negation in $\widetilde{\Pi}$ is done by
$$\boxminus(c_0, c_1, c_2) = (c_0^{-1},\ c_1^{-1},\ c_2^{-1}) \tag{12}$$
where the inverse is taken in the component-wise manner.
- $\widetilde{\Pi}.\mathsf{Rnd}(c)$: Output $c \boxplus \widetilde{\Pi}.\mathsf{Enc}_{\mathsf{pk},y}(0)$.
- $\widetilde{\Pi}.\mathsf{Verify}(\mathsf{pk}, c, y)$: Parse $c = (c_0, c_1, c_2)$, $c_1 = (u_1, \dots, u_\ell)$ for some ℓ, and $c_2 = (v_1, \dots, v_w)$. Let $a_{\iota,i,j}$ be the coefficient of monomial $s_i b_j$ in the ι-th polynomial in the vector $f_y(\vec{s}, \vec{b})$. For each $\iota \in [1, \ell]$, check if
$$e(g, u_\iota) \overset{?}{=} \prod_{i=0}^{w} \prod_{j=1}^{n} e(g^{a_{\iota,i,j} b_j}, v_i) \ . \tag{13}$$
If this holds for every ι, then output 1. Otherwise, output 0.

Proposition 19. *Suppose that Π is a secure ABE scheme for predicate P. Then $\widetilde{\Pi}$ is a secure AH-ABE scheme for predicate P in the same sense and is, in particular, publicly verifiable.*

Proof. We note that the encryption algorithm in $\widetilde{\Pi}$ is essentially the same as Π except the expressions m and T^m of plaintexts. As the additional functionalities in AH-ABE which were not in ABE are all publicly available, these do not affect the security, therefore the security of $\widetilde{\Pi}$ is trivially reduced to the security of Π.

Due to the linearity of the exponents in c with respect to the variables s_i, the operators in Eq.(11) and Eq.(12) in fact realize the homomorphic addition and negation, respectively. The required property for the algorithm Rnd also holds obviously. It remains to prove the correctness of Verify. For each $i \in [0, w]$, put $v_i = g^{s_i}$ for the uniquely determined $s_i \in \mathbb{F}_p$. Then for each $\iota \in [1, \ell]$, the right-hand side of Eq.(13) is
$$\prod_{i=0}^{w} \prod_{j=1}^{n} e(g^{a_{\iota,i,j} b_j}, g^{s_i}) = \prod_{i=0}^{w} \prod_{j=1}^{n} T^{a_{\iota,i,j} b_j s_i} = T^{\sum_{i=0}^{w} \sum_{j=1}^{n} a_{\iota,i,j} s_i b_j} = T^{f_y(\vec{s},\vec{b})_\iota}$$
where $f_y(\vec{s}, \vec{b})_\iota$ denotes the ι-th polynomial in the vector $f_y(\vec{s}, \vec{b})$. This implies that, Eq.(13) holds for every $\iota \in [1, \ell]$ if and only if $c_1 = (g^{f_y(\vec{s},\vec{b})_1}, \dots, g^{f_y(\vec{s},\vec{b})_\ell}) = g^{f_y(\vec{s},\vec{b})}$, or equivalently, c is a valid ciphertext with attribute y and randomness $\vec{s}$. This completes the proof of Proposition 19. $\square$

Applicability. The ABE template introduced above covers many ABE constructions (for many predicates) in the literature. First, for concreteness, we can see that the additively homomorphic IBE scheme described above is obtained by applying our conversion to the Boneh–Boyen IBE [5]. Moreover, although we do not go into detailed descriptions, it is seen that ABE schemes of [12, 20, 30, 4, 31] in fact conform our base ABE template and hence can be all converted to AH-ABE. We note also that, while the argument above assumed symmetric pairing, more efficient schemes with asymmetric pairing can be obtained as well by folklore methods or a generic method in [1].

Acknowledgements

This research and development work was supported by the Ministry of Internal Affairs and Communications Grant Number 182103105, by JST CREST Grant Number JPMJCR19F6, and by JSPS KAKENHI Kiban-A Grant Number 19H01109. The authors thank Masashi Horikoshi at Intel to provide Intel Xeon Scalable Processor environments to evaluate our protocol.

References

[1] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Takeya Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 241–260. Springer, Heidelberg, August 2014.

[2] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.

[3] Nuttapong Attrapadung, Goichiro Hanaoka, Shigeo Mitsunari, Yusuke Sakai, Kana Shimizu, and Tadanori Teruya. Efficient two-level homomorphic encryption in prime-order bilinear groups and A fast implementation in webassembly. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 685–697, 2018.

[4] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Heidelberg, March 2011.

[5] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[6] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. Targeted homomorphic attribute-based encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 330–360. Springer, Heidelberg, October / November 2016.

[7] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015.

[8] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 44–61, 2010.

[9] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 10–18, 1984.

[10] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[11] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications.* Cambridge University Press, 2004.

[12] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

[13] Per A. Hallgren, Martín Ochoa, and Andrei Sabelfeld. Bettertimes - privacy-assured outsourced multiplications for additively homomorphic encryption on finite fields. In *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, pages 291–309, 2015.

[14] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. *Journal of Computer Security*, 21(2):283–315, 2013.

[15] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[16] Baptiste Vinh Mau and Koji Nuida. Correction of a secure comparison protocol for encrypted integers in IEEE WIFS 2012 (short paper). In *Advances in Information and Computer Security - 12th International Workshop on Security, IWSEC 2017, Hiroshima, Japan, August 30 - September 1, 2017, Proceedings*, pages 181–191, 2017.

[17] Shigeo Mitsunari. mcl.

[18] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[19] Satsuya Ohata and Koji Nuida. Towards high-throughput secure MPC over the internet: Communication-efficient two-party protocols and its application. *CoRR*, abs/1907.03415, 2019.

[20] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 195–203. ACM Press, October 2007.

[21] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[22] Tord Ingolf Reistad and Tomas Toft. Linear, constant-rounds bit-decomposition. In *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, pages 245–257, 2009.

[23] Bharath K. Samanthula, Chun Hu, and Wei Jiang. An efficient and probabilistic secure bit-decomposition. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 541–546, 2013.

[24] Berry Schoenmakers and Pim Tuyls. Efficient binary conversion for paillier encrypted values. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 522–537, 2006.

[25] Kana Shimizu, Koji Nuida, and Gunnar Rätsch. Efficient privacy-preserving string search and an application in genomics. *Bioinformatics*, 32(11):1652–1661, 2016.

[26] Hiroki Sudo, Koji Nuida, and Kana Shimizu. An efficient private evaluation of a decision graph. In *Information Security and Cryptology - ICISC 2018 - 21st International Conference, Seoul, South Korea, November 28-30, 2018, Revised Selected Papers*, pages 143–160, 2018.

[27] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Utku Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 519–528, 2007.

[28] Thijs Veugen. Encrypted integer division. In *2010 IEEE International Workshop on Information Forensics and Security, WIFS 2010, Seattle, WA, USA, December 12-15, 2010*, pages 1–6, 2010.

[29] Thijs Veugen. Improving the DGK comparison protocol. In *2012 IEEE International Workshop on Information Forensics and Security, WIFS 2012, Costa Adeje, Tenerife, Spain, December 2-5, 2012*, pages 49–54, 2012.

[30] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, Heidelberg, March 2011.

[31] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 275–292. Springer, Heidelberg, March 2014.

[32] Shota Yamada, Nuttapong Attrapadung, Bagus Santoso, Jacob C. N. Schuldt, Goichiro Hanaoka, and Noboru Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 243–261. Springer, Heidelberg, May 2012.

Appendix

A Detail for a 2LHE Scheme

Here we describe in detail the construction of the 2LHE scheme in [3] (with slight modifications that are not essential). Here $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_{\mathrm{T}}$ is a nondegenerate, Type 3 bilinear pairing with multiplicative cyclic groups $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_{\mathrm{T}}$ of the same prime order p. Given elements $h_1 \in \mathbb{G}_1$ and $h_2 \in \mathbb{G}_2$ as below, for $m_1, m_2, \rho, \sigma \in \mathbb{F}_p$, we write

$$[m_1, m_2; \rho, \sigma]^{\langle 1 \rangle} \overset{\mathrm{def}}{=} ({g_1}^{\rho}, {g_1}^{m_1} {h_1}^{\rho}, {g_2}^{\sigma}, {g_2}^{m_2} {h_2}^{\sigma}) \in {\mathbb{G}_1}^2 \times {\mathbb{G}_2}^2 \ .$$

We set

$$\mathcal{C}_{m,m'}^{\langle 1 \rangle} \overset{\mathrm{def}}{=} \{(1, [m, m'; \rho, \sigma]^{\langle 1 \rangle}) \mid \rho, \sigma \in \mathbb{F}_p\} \text{ for } m, m' \in \mathbb{F}_p$$

and write $\mathcal{C}_m^{\langle 1 \rangle} = \mathcal{C}_{m,m}^{\langle 1 \rangle}$. On the other hand, given elements z_1, z_2, z_3, z_4 of $\mathbb{G}_{\mathrm{T}}$ as below, for $m, \rho, \sigma, \tau \in \mathbb{F}_p$, we write

$$[m; \rho, \sigma, \tau]^{\langle 2 \rangle} \overset{\mathrm{def}}{=} ({z_1}^{\rho+\sigma-\tau}, {z_2}^{\rho}, {z_3}^{\sigma}, {z_1}^{m} {z_4}^{\tau}) \in {\mathbb{G}_{\mathrm{T}}}^4 \ .$$

We set

$$\mathcal{C}_m^{\langle 2 \rangle} \overset{\mathrm{def}}{=} \{(2, [m; \rho, \sigma, \tau]^{\langle 2 \rangle}) \mid \rho, \sigma, \tau \in \mathbb{F}_p\} \text{ for } m \in \mathbb{F}_p \ .$$

Then the construction of the scheme is as follows.

- $\mathsf{Gen}(1^\lambda)$: Choose $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, $\mathbb{G}_\mathrm{T}$ and e as above. Choose $s_1, s_2 \xleftarrow{R} \mathbb{F}_p^\times$ and set $h_1 \leftarrow {g_1}^{s_1}$ and $h_2 \leftarrow {g_2}^{s_2}$. Compute $z_1 \leftarrow e(g_1, g_2)$, $z_2 \leftarrow e(g_1, h_2)$, $z_3 \leftarrow e(h_1, g_2)$, $z_4 \leftarrow e(h_1, h_2)$, and output $(\mathsf{pk}, \mathsf{sk})$ where
$$\mathsf{pk} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, g_1, g_2, e, h_1, h_2, z_1, z_2, z_3, z_4)\, , \, \mathsf{sk} \leftarrow (s_1, s_2) \ .$$
We set $\mathcal{M} = \mathbb{F}_p$, and set $\mathcal{M}_\mathrm{eff}^{\langle 1 \rangle}$ and $\mathcal{M}_\mathrm{eff}^{\langle 2 \rangle}$ to be any subsets of $\mathcal{M}$ with polynomially bounded numbers of elements. Moreover, we set $\mathcal{C}^{\langle 1 \rangle} = \bigsqcup_{m \in \mathcal{M}} \mathcal{C}_m^{\langle 1 \rangle}$ and $\mathcal{C}^{\langle 2 \rangle} = \bigsqcup_{m \in \mathcal{M}} \mathcal{C}_m^{\langle 2 \rangle}$.
- $\mathsf{Enc}^{\langle 1 \rangle}(m) = (1, [m, m; \rho, \sigma]^{\langle 1 \rangle})$ with $\rho, \sigma \xleftarrow{R} \mathbb{F}_p$.
- $\mathsf{Enc}^{\langle 2 \rangle}(m) = (2, [m; \rho, \sigma, \tau]^{\langle 2 \rangle})$ with $\rho, \sigma, \tau \xleftarrow{R} \mathbb{F}_p$.
- $\mathsf{Dec}^{\langle 1 \rangle}(1, (c_1, c_2, c_3, c_4))$: Search an $m \in \mathcal{M}_\mathrm{eff}^{\langle 1 \rangle}$ satisfying $c_1^{-s_1} c_2 = {g_1}^m$. If such an m is found then output the m; otherwise output $\perp$.
- $\mathsf{Dec}^{\langle 2 \rangle}(2, (c_1, c_2, c_3, c_4))$: Search an $m \in \mathcal{M}_\mathrm{eff}^{\langle 2 \rangle}$ satisfying $c_1^{s_1 s_2} {c_2}^{-s_1} {c_3}^{-s_2} c_4 = {z_1}^m$. If such an m is found then output the m; otherwise output $\perp$.
- $(i, (c_1, c_2, c_3, c_4)) \boxplus^{\langle i \rangle} (i, (c'_1, c'_2, c'_3, c'_4))$ for $i \in \{1, 2\}$: Output $(i, (c_1 c'_1, c_2 c'_2, c_3 c'_3, c_4 c'_4))$.
- $\boxminus^{\langle i \rangle}(i, (c_1, c_2, c_3, c_4)) = (i, ({c_1}^{-1}, {c_2}^{-1}, {c_3}^{-1}, {c_4}^{-1}))$ for $i \in \{1, 2\}$.
- $\mathsf{Rnd}^{\langle i \rangle}(i, c)$ for $i \in \{1, 2\}$: Output $(i, c) \boxplus^{\langle i \rangle} \mathsf{Enc}^{\langle i \rangle}(0)$.
- $(1, (c_1, c_2, c_3, c_4)) \boxtimes^{\langle 1 \rangle} (1, (c'_1, c'_2, c'_3, c'_4))$: First we define
$$(x_1, x_2) \times (y_1, y_2) \overset{\mathrm{def}}{=} (e(x_1, y_1), e(x_1, y_2), e(x_2, y_1), e(x_2, y_2))$$
for $(x_1, x_2) \in {\mathbb{G}_1}^2$ and $(y_1, y_2) \in {\mathbb{G}_2}^2$. Then the operator outputs $(2, (c_1, c_2) \times (c'_3, c'_4))$.

The argument in [3] shows that the scheme correctly satisfies the conditions for a 2LHE scheme in Definition 3. We note that, for invalid ciphertexts (i.e., elements of $\mathcal{C}_{m,m'}^{\langle 1 \rangle}$ with $m \neq m'$), we have
$$\mathcal{C}_{m_1, m'_1}^{\langle 1 \rangle} \boxtimes^{\langle 1 \rangle} \mathcal{C}_{m_2, m'_2}^{\langle 1 \rangle} \subseteq \mathcal{C}_{m_1 \cdot m'_2}^{\langle 2 \rangle} \text{ for any } m_1, m'_1, m_2, m'_2 \in \mathcal{M} \ .$$