Mis-operation Resistant Searchable Homomorphic Encryption

Keita Emura
National Institute of Information and Communications Technology (NICT)
k-emura@nict.go.jp

Takuya Hayashi
National Institute of Information and Communications Technology (NICT)
takuya.hayashi@nict.go.jp

Noboru Kunihiro
The University of Tokyo
JST CREST
kunihiro@k.u-tokyo.ac.jp

Jun Sakuma
University of Tsukuba
JST CREST
RIKEN Center for AIP
jun@cs.tsukuba.ac.jp

ABSTRACT

Let us consider a scenario that a data holder (e.g., a hospital) encrypts a data (e.g., a medical record) which relates a keyword (e.g., a disease name), and sends its ciphertext to a server. We here suppose not only the data but also the keyword should be kept private. A receiver sends a query to the server (e.g., average of body weights of cancer patients). Then, the server performs the homomorphic operation to the ciphertexts of the corresponding medical records, and returns the resultant ciphertext. In this scenario, the server should NOT be allowed to perform the homomorphic operation against ciphertexts associated with different keywords. If such a mis-operation happens, then medical records of different diseases are unexpectedly mixed. However, in the conventional homomorphic encryption, there is no way to prevent such an unexpected homomorphic operation, and this fact may become visible after decrypting a ciphertext, or as the most serious case it might be never detected.

To circumvent this problem, in this paper, we propose mis-operation resistant homomorphic encryption, where even if one performs the homomorphic operations against ciphertexts associated with keywords ω' and ω, where $\omega \neq \omega'$, the evaluation algorithm detects this fact. Moreover, even if one (intentionally or accidentally) performs the homomorphic operations against such ciphertexts, a ciphertext associated with a random keyword is generated, and the decryption algorithm rejects it. So, the receiver can recognize such a mis-operation happens in the evaluation phase. In addition to mis-operation resistance, we additionally adopt secure search functionality for keywords since it is desirable when one would like to delegate homomorphic operations to a third party. So, we call the proposed primitive mis-operation resistant searchable homomorphic encryption (MR-SHE).

We also give our implementation result of inner products of encrypted vectors. In the case when both vectors are encrypted, the running time of the receiver is millisecond order for relatively small-dimensional (e.g., 2^6) vectors. In the case when one vector is encrypted, the running time of the receiver is approximately 5 msec even for relatively high-dimensional (e.g., 2^{13}) vectors.

ASIA CCS '17, April 02-06, 2017, Abu Dhabi, United Arab Emirates

DOI: http://dx.doi.org/10.1145/3052973.3053015

Keywords

Homomorphic Encryption (HE); Searchable Encryption; Mis-operation Resistance

1. INTRODUCTION

1.1 Research Background

Let us consider a scenario that a data holder encrypts a data and sends its ciphertext to a server. The server computes some statistical values of data without decrypting ciphertexts, and sends the ciphertext of the final statistical value to a decryptor. The decryptor obtains the statistical value by decryption. This is a typical scenario of homomorphic encryption (HE) such as the Paillier encryption scheme [28]. Though the Paillier encryption supports additive homomorphic operation only, after seminal works by Gentry [20], several fully homomorphic encryption (FHE) schemes have been proposed so far. Due to the progress of (F)HE area, several applications of (F)HE also have been proposed.

On the other hand, a security drawback of HE also has been widely recognized so far. That is, anyone can freely perform homomorphic operations inevitably, and this means ciphertexts are malleable. Especially, a standard security level of Public Key Encryption (PKE), which we call security against adaptive chosen-ciphertext attack (CCA), is never achieved in HE. By changing the model of HE, Emura et al. [15, 14] proposed a way to achieve the CCA security and the homomorphic property simultaneously, where it is CCA secure against an adversary who does not have

the homomorphic operation key[1] (and the decryption key also), and simultaneously the homomorphic operation is allowed to a user who has the homomorphic operation key. This primitive is called keyed homomorphic public key encryption (KH-PKE), and several constructions have been proposed so far, e.g., KH-PKE with public verifiability [23, 27], keyed FHE [26], and keyed homomorphic identity-based encryption (KH-IBE) [14].

1.2 Our Target: Mis-operation Resistant Searchable Homomorphic Encryption

Though we can somewhat control who will be able to perform the homomorphic operation by employing KH-PKE, there is room for argument on the controllability of the homomorphic operation. For example, let us consider a scenario that a data holder (e.g., a hospital) encrypts a data (e.g., a medical record) which relates to a keyword (e.g., a disease name), and sends its ciphertext to a server. We here suppose not only the data but also the keyword should be kept private. For example, when the keyword indicates rare diseases, the keyword might cause identification of the patient even if the records are encrypted. In case of genetic diseases, it can leak familial relationships between records, too. A receiver sends a query to the server (e.g., average of body weights of cancer patients), in which the operation contains conditioning by a keyword. Then, the server performs the homomorphic operation only with the ciphertexts of the medical records containing the keyword specified by the query, and returns the ciphertext of the average of body weights of them. In this scenario, the server should NOT be allowed to perform the homomorphic operation against ciphertexts related to different keywords. If such a mis-operation happens, then medical records of different diseases are unexpectedly mixed. Of course, if all the programs that will process the ciphertexts are determined in advance, we do not have a strong motivation to consider such a mis-operation of ciphertexts. However, from the standpoint of data engineering, if ciphertexts are stored in a database and repeatedly used for various purposes for a long time, management of the provenance of the ciphertexts is not an easy task because of the security of the ciphertexts, particularly in the outsourcing setting. Once data is encrypted, no one except the secret key holder can confirm the provenance of the data anymore. However, in the conventional HE (and KH-PKE also), there is no way to prevent such an unexpected homomorphic operation, and this fact may become visible after decrypting a ciphertext, or as the most serious case it might be never detected.

One may think that considering some tags of ciphertexts is a reasonable solution since it could distinguish whether homomorphic operations are allowed or not. However, if keywords are directly regarded as tags, disease names become known to the server. Therefore, providing a secure keyword search functionality could be a solution.

1.3 Naive Approach and Its Limitations

The most naive approach is to simply combine a ciphertext of a searchable encryption scheme (public key encryption with keyword search (PEKS) [6]) and a ciphertext of a public key HE scheme. However, as pointed out in [3, 5, 34, 11], this simple setting does not achieve an appropriate security condition. For example, even if the public key encryption scheme is CCA secure, the combined ciphertexts are not secure against the CCA attack. In other words, since ciphertexts of the HE scheme are still malleable or ciphertexts of searchable encryption are replaced to ciphertexts associated with a different keyword by an adversary, the server may perform homomorphic operations against them even the server follows the protocol. Moreover, even if a (lower-level trusted) server forcibly performs homomorphic operations against ciphertexts with different keywords, the decryption algorithm should reject the result, and the receiver should be able to recognize such a mis-operation happens in the evaluation phase. So, regardless of trust level of the server, such a mis-operation should be protected in the scheme level, and this functionality is not supported in the simple construction. For considering a CCA security in this setting, PEKS/PKE has been proposed [3, 5, 34, 11]. However, since PEKS/PKE does not preserve the homomorphic property of the underlying PKE scheme, PEKS/PKE is not applicable for our usage.

For achieving the mis-operation resistance with preserving keyword privacy, one may think that it is enough to employ a double encryption methodology, where prepare an encryption and decryption key pair for each disease name, and encrypt a ciphertext of a HE scheme by using an encryption key associated with a disease when it is related to the disease. Then, a server who has a decryption key of a disease can perform homomorphic operations to ciphertexts associated with the disease. However, many key pairs need to be managed. Moreover, if a server has more than one key pair, then still the server can perform homomorphic operations against ciphertexts associated with different diseases, and this fact is not detected.

Verifiable computation, e.g., [17, 4], might be a solution. That is, if the receiver (client in the verifiable computation context) can specify a function F that can exclude ciphertexts associated with different keywords, then the receiver can recognize whether a mis-operation happens in the evaluation phase or not, by verifying whether the output is a correct evaluation of F. If the keyword is publicly available and the server can know it, it seems we can directly employ verifiable computations. However, as mentioned before we here suppose not only the data but also the keyword should be kept private. Thus, since secure searchability is not supported, current verifiable computations are not directly applicable in our usage to the best of our knowledge.

Indistinguishability obfuscation ($i\mathcal{O}$) [18] (or functional encryption [8] also) might be applicable where, for example, a plaintext space is partitioned into multiple spaces and homomorphic operations are allowed if the corresponding plaintexts belong to the same space. Though this could be a solution, this is not an efficient solution since the current efficiency of $i\mathcal{O}$ is far from a practical use. Targeted malleability [8], where homomorphic operations are allowed with respect to a function $F \in \mathcal{F}$, also seems to be a candidate to achieve the mis-operation resistance. However, the evaluation algorithm adds a succinct non-interactive argument that proves a function $F \in \mathcal{F}$ is performed, and it

[1]We do not use the word "evaluation key" in order to distinguish evaluation keys of FHE schemes which are contained in public keys for homomorphic operations. As a remark, though some FHE schemes, e.g., [9], require the evaluation key, these schemes do not consider the CCA security against one who does not have the evaluation key.

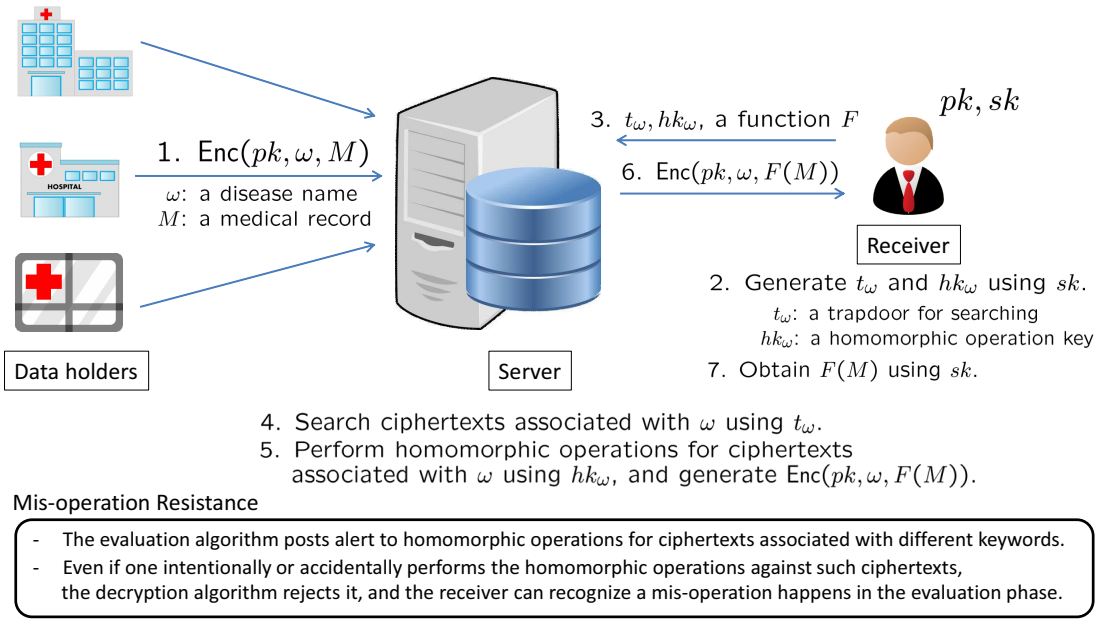

Figure 1: Framework of MR-SHE

seems non-trivial to instantiate an argument system for a language that supports the mis-operation resistance.

Another candidate is KH-IBE [14] where a homomorphic operation key is defined for each identity, and the homomorphic operation is allowed against ciphertexts encrypted by the same identity. Moreover, Abdalla et al. [2] show that PEKS can be generically constructed by anonymous identity-based encryption (anonymous IBE). Since the KH-IBE scheme [14] is constructed by the Gentry IBE scheme [19] and the Gentry IBE scheme is anonymous, the KH-IBE scheme is also anonymous with a reasonable setting (See Def 7.6 as a remark). So, one may expect that PEKS constructed from the Gentry-based KH-IBE scheme can work well to achieve mis-operation resistance. However, the Abdalla et al. construction cannot be used for our purpose. Briefly, in the Abdalla et al. construction, a random plaintext (say R) is encrypted by a keyword as the identity, the corresponding trapdoor is a decryption key of the keyword, and a ciphertext of searchable encryption is this IBE ciphertext and R. The test algorithm returns 1 if the decryption result of the IBE ciphertext by using the trapdoor is R. That is, the corresponding plaintext R is required for the test capability, and is directly contained in the ciphertext. This is meaningless in our usage, i.e., for searching disease names in a secure way, medical records need to be revealed to the server. Moreover, even if we can circumvent this problem, the evaluation algorithm requires the corresponding identity as input. That is, the corresponding keyword is known to the server. So, we need to invent other way to construct mis-operation resistant homomorphic encryption with secure searchability.

1.4 Our Contribution

In this paper, we propose homomorphic encryption with both mis-operation resistant and secure searchability, which we call mis-operation resistant searchable homomorphic encryption (MR-SHE). MR-SHE supports the following properties.

Confidentiality: A keyword and data are encrypted simultaneously, and no information of the keyword and data is revealed from the ciphertext, as in PEKS/PKE.

Secure Searchability: A secure keyword search is allowed, as in PEKS.

Keyed Homomorphic Property: No one, except a user who has a homomorphic operation key, can perform homomorphic operations, and CCA security is guaranteed, as in KH-PKE.

Mis-operation Resistance: The evaluation algorithm posts alert to homomorphic operations for ciphertexts associated with different keywords. Even if one intentionally or accidentally performs the homomorphic operations against such ciphertexts, the decryption algorithm rejects it, and the receiver can recognize a mis-operation happens in the evaluation phase.

See Figure 1 for a brief description of MR-SHE. A receiver (a researcher) setups a key pair (pk, sk), and a data holder (a hospital) encrypts a medical record M and its disease name ω by using the public key of the receiver pk, and sends its ciphertext to the server. No information of ω and M is leaked from the ciphertext. For a disease name ω, the receiver computes a trapdoor t_ω which can be used for searching ciphertexts associated with ω, and also computes a homomorphic operation key hk_ω which can be used for performing homomorphic operations over M to ciphertexts associated with ω. The receiver secretly sends t_ω and hk_ω to a server. The server searches ciphertexts associated with ω by using t_ω, and performs homomorphic operations F to ciphertexts associated with ω by using hk_ω.[2] Finally, the server sends a ciphertext of $F(M)$ to the receiver, and the receiver ob-

[2]Remark that our scheme inherits linear search complexity of PEKS.

tains $F(M)$ by decrypting the ciphertext using sk.[3] It is particularly worth noting that even if the server performs homomorphic operations against ciphertexts related to ω' by using hk_ω, where $\omega \neq \omega'$, the evaluation algorithm detects this fact. Moreover, even if one forcibly performs the homomorphic operations against such ciphertexts, a ciphertext associated with a random keyword is generated, the decryption algorithm rejects it, and the receiver can recognize such a mis-operation happens in the evaluation phase.

Technically, we point out that homomorphic operation keys can be trapdoors for searching whereas decryption keys are trapdoors in the Abdalla et al. construction [2]. That is, in our construction, $t_\omega = hk_\omega$. Then no plaintext is contained in the ciphertext in contrast to the Abdalla et al. construction. Moreover, we prove that our scheme has consistency where the test algorithm does not output 1 with overwhelming probability if a ciphertext is tested by a trapdoor of a different keyword. Since the KH-IBE scheme is anonymous, no information of ω is revealed from the ciphertext. Moreover, the CCA security against adversaries who do not have the homomorphic operation key is taken over from the security of KH-IBE. So, we can prevent unexpected modifications of ciphertexts by the adversaries, and can protect information of plaintexts them in the sense of the CCA security. Note that we need to add a value to homomorphic operation keys in order for the server to be allowed to perform homomorphic operations without knowing the corresponding keyword itself, and show that this modification does not affect the security.

Though the proposed MR-SHE scheme supports multiplicative homomorphic operations on the target group of pairing, additive homomorphic operations over suitably small integers can also be supported by the lifted ElGamal encryption approach [13]. Moreover, by employing the Catalano-Fiore transformation [10] which transforms an additive homomorphic encryption scheme into a scheme which additionally supports one multiplicative homomorphic operation, we realize inner products of encrypted vectors. We implement it by using the Pairing-Based Cryptography (PBC) library [1], and show the running time is still reasonable. We also propose an algorithm that evaluates multiple ciphertexts by single execution. This modification allows us to efficiently implement inner products of vectors.

1.5 Our Scenario: χ^2 Test of Independence for Hidden Keywords

We introduce a scenario that inner products of encrypted vectors need to be securely computed as follows. The χ^2 test for independence is designed to test the null hypothesis that there is no association between a pair of factors. Suppose a researcher, who has (pk, sk), and hospitals (hospital 1 and 2) collaboratively assess dependence between a SNP (Single Nucleotide Polymorphism) variant and a target disease. To this end, the researcher and hospitals compute a frequency table for a case-control study (Table 1). A case group includes subjects with a target disease (e.g., lung cancer), and a control group includes non-diseased subjects.

[3]Note that the receiver can decrypt a ciphertext $\mathsf{Enc}(pk, \omega, M)$ sent from a data holder to the server, and can directly obtain M in the current setting. If this case should be avoided, then we can simply assume that the server also has a public key and a secret key, and data holders encrypt the ciphertext by using the server public key.

Table 1: Frequency table

	A	a	
Case	n_{1A}	n_{1a}	$n_1 := n_{1A} + n_{1a}$
Control	n_{2A}	n_{2a}	$n_2 := n_{2A} + n_{2a}$
	$n_A := n_{1A} + n_{2A}$	$n_a := n_{1a} + n_{2a}$	n

Let us consider a biallelic locus with allele A and a. Here, n is the total number of subjects, and other value, e.g., n_{1A} is the number of subjects affecting the target disease and having allele A at the locus. Assume that the hospital 1 has a vector $\boldsymbol{x}_0 := (x_{0,1}, \ldots, x_{0,\ell}) \in \mathbb{Z}_2^\ell$ where $x_{0,i} = 1$ if the subject i affects the target disease, and $x_{0,i} = 0$ otherwise. Assume that the hospital 2 has a vector $\boldsymbol{x}_1 := (x_{1,1}, \ldots, x_{1,\ell}) \in \mathbb{Z}_2^\ell$ where $x_{1,i} = 1$ if the subject i has allele A at the locus, and $x_{1,i} = 0$ otherwise. Then, n_{1A} can be computed as $n_{1A} = \sum_{i=1}^{\ell} x_{0,i}x_{1,i}$. Since n, n_1, n_2, n_A, and n_a are contained in the table, assume that these values are publicly available among the researcher and hospitals. That is, if n_{1A} is computed, then all other values can be computed. From the table, χ^2 test statistic is computed as $T = \frac{n(n_{1A}n_{2a} - n_{1a}n_{2A})^2}{n_A n_a n_1 n_2}$. Greater T indicates stronger dependence between the SNP and the disease. Let us consider the case that the computation of n_{1A} is delegated to a server (which is not the fully trusted third party). Since $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ are highly sensitive information, these vectors need to be hidden to the server. Moreover, since disease name is also sensitive information, such a keyword is also desired to be hidden to the server. To compute $n_{1A} = \sum_{i=1}^{\ell} x_{0,i}x_{1,i}$, without revealing $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, and the target disease to the server, we employ MR-SHE as follows. The hospital 1 and the hospital 2 encrypt $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, respectively, with the keyword "lung cancer", and send the ciphertexts to the server. Here, we assume that the hospital 1 and the hospital 2 also compute ciphertexts for other target diseases (e.g., glycosuria, nosocomephrenia, leukemia, and so on), and the server preserves them. When the researcher needs to compute the χ^2 test statistic for "lung cancer", the researcher computes the corresponding trapdoor and the corresponding homomorphic operation key, and sends the keys with the query to the server. The server searches the corresponding ciphertexts by using the trapdoor, and computes a ciphertext of n_{1A} by using the homomorphic operation key. Then, we can guarantee that the ciphertext of n_{1A} is computed by two encrypted vectors associated with the same disease. Moreover, no information of diseases is revealed from the ciphertexts.

1.6 Related Work

Shimizu et al. [31] pointed out that tversky index, which evaluates a distance among chemical compounds, can be captured by using addition operations, and the Paillier encryption scheme [28] is employed for searching. We remark that the homomorphic property is used for searching, and is not used for evaluating encrypted chemical compounds data. As a similar attempt, secure matching or private database queries using somewhat HE schemes have been proposed in [7, 32, 33, 25].

Kiltz [24] proposed a tag-based encryption (TBE) where a ciphertext is associated with a tag. As an interesting property, we pointed out that the Kiltz TBE scheme has a homomorphic property for ciphertexts if they are associated

with the same tag (even a kind of chosen ciphertext security is guaranteed). Moreover, a public verification algorithm is defined where a ciphertext is valid under a tag or not. Similarly, Gentry, Sahai and Waters [21] propose identity-based FHE where homomorphic operations are allowed to ciphertexts encrypted by the same identity. This scheme does not require a user-specific evaluation key. On the other hand, our scheme requires a keyword-specific evaluation key for keyed homomorphic property, and this setting allows us to control who can perform the homomorphic operations and to protect information of plaintext in the sense of the CCA security. So, these schemes [24, 21] might be applicable for our usage if neither secure search functionality nor keyed homomorphic property are required.

Since the receiver can recognize that all ciphertexts of homomorphic operations are associated with the same keyword, MR-SHE can be regarded as a kind of verifiable computation [17, 4] with searchability. However, MR-SHE does not support the verifiability of the computation result unlike the conventional verifiable computations. For ciphertexts with the same keyword, the server still may compute the function that the receiver requires is also regarded as a mis-operation. Thus, supporting such the verifiability in the MR-SHE context is an interesting future work of this paper.

2. DEFINITIONS OF MIS-OPERATION RESISTANT SEARCHABLE HOMOMORPHIC ENCRYPTION

In this section, we give definitions of mis-operation resistant searchable homomorphic encryption (MR-SHE). A MR-SHE scheme $\mathcal{MR}\text{-}\mathcal{SHE}$ consists of eight algorithms: (MR-SHE.KeyGen, MR-SHE.HomKeyGen, MR-SHE.Trapdoor, MR-SHE.Enc,
MR-SHE.Test, MR-SHE.Dec, MR-SHE.Eval). Let $\mathcal{W}$ be the keyword space, $\mathcal{M}$ be the plaintext space, and $\kappa \in \mathbb{N}$ be the security parameter. Note that in our construction given in Section 3, we set $hk_\omega = t_\omega$. So, we assume that key generation algorithms share these keys, and thus our scheme is stateful. See Section 3 for details. For the sake of simplicity, we define algorithms when these are run for the first time.

MR-SHE.KeyGen : A key generation algorithm takes as input 1^κ, and returns a receiver public key pk and a receiver secret key sk. We assume that $\mathcal{W}$ and $\mathcal{M}$ are contained in pk.

MR-SHE.KeyGen : A key generation algorithm takes as input 1^κ, and returns a receiver public key pk and a receiver secret key sk. We assume that $\mathcal{W}$ and $\mathcal{M}$ are contained in pk.

MR-SHE.HomKeyGen : A homomorphic operation key generation algorithm takes as input pk, sk and a keyword $\omega \in \mathcal{W}$, and returns a homomorphic operation key hk_ω.

MR-SHE.Trapdoor : A trapdoor generation algorithm takes as input pk, sk, and a keyword $\omega \in \mathcal{W}$, and returns a trapdoor t_ω corresponding to keyword ω.

MR-SHE.Enc : An encryption algorithm takes as input pk, ω, and M, and returns a ciphertext C.

MR-SHE.Test : A test algorithm takes as input pk, t_ω, and C, and returns 1 or 0.

MR-SHE.Dec : A decryption algorithm takes as input pk, sk, ω, and C, and returns M or $\perp$.

MR-SHE.Eval : An evaluation algorithm takes as input pk, hk_ω, and two ciphertexts C_1 and C_2, and returns a ciphertext C or $\perp$. This algorithm has two functionalities: 1) checks the integrity and 2) evaluates the function homomorphically.

We require the correctness property as follows: For all $(pk, sk) \leftarrow \mathsf{MR\text{-}SHE.KeyGen}(1^\kappa)$, and all $\omega \in \mathcal{K}$ and $M \in \mathcal{M}$, for $C \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega, M)$ and $t_\omega \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$,

- $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C) = 1$ holds, and
- $\mathsf{MR\text{-}SHE.Dec}(pk, sk, \omega, C) = M$ holds.

Moreover, we require the homomorphic property as follows: For all $(pk, sk) \leftarrow \mathsf{MR\text{-}SHE.KeyGen}(1^\kappa)$, and all $\omega \in \mathcal{K}$ and $M_1, M_2 \in \mathcal{M}$, and all $hk_\omega \leftarrow \mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$,

- For $C \leftarrow \mathsf{MR\text{-}SHE.Eval}(pk, hk_\omega, C_1, C_2)$ where $C_1 \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega, M_1)$ and $C_2 \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega, M_2)$, $\mathsf{MR\text{-}SHE.Dec}(pk, sk, \omega, C) = M_1 \odot M_2$ holds, where $\odot$ is a binary operation over $\mathcal{M}$.

Next, we define consistency as follows. This guarantees that the MR-SHE.Test algorithm does not output 1 with overwhelming probability if a ciphertext is tested by a trapdoor of a different keyword. Remark that in our scheme a fresh ciphertext generated by the encryption algorithm and a ciphertext output by the evaluation algorithm have the same form and are identical. So, we simply consider the case that a ciphertext is computed by the encryption algorithm here.

Definition 2.1 (CONSISTENCY). *For any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathbb{N}$, we define the experiment* $\mathrm{Exp}^{consist}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$ *as follows.*

$\mathrm{Exp}^{consist}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$:
$(pk, sk) \leftarrow \mathsf{MR\text{-}SHE.KeyGen}(1^\kappa)$
$(\omega, \omega', M) \leftarrow \mathcal{A}(pk)$
$\omega, \omega' \in \mathcal{W};\ \omega \neq \omega';\ M \in \mathcal{M}$
$C \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega, M)$
$t_{\omega'} \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega')$
if $\mathsf{MR\text{-}SHE.Test}(pk, t_{\omega'}, C) = 1$ *then return* 1
else return 0

We say that $\mathcal{MR}\text{-}\mathcal{SHE}$ is consistent if the advantage $\mathrm{Adv}^{consist}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa) := \Pr[\mathrm{Exp}^{consist}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa) = 1]$ *is negligible for any PPT adversary $\mathcal{A}$.*

Next, we define data privacy which guarantees that no information of plaintext is revealed from ciphertexts. As in KH-IBE, an adversary is allowed to obtain the evaluation results even for the challenge ciphertext, and such challenge-related ciphertexts are not allowed to be inputs of the decryption oracle $\mathcal{O}_{dec}$. The evaluation oracle $\mathcal{O}_{eval}$ returns the result of the MR-SHE.Eval algorithm. That is, the oracle returns $\perp$ unless two ciphertexts input are associated with the same keyword. This captures the property that evaluation should fail when the ciphertexts have non-matching tags.

Definition 2.2 (Data Privacy). *For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathbb{N}$, we define the experiment* $\mathrm{Exp}^{data\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$ *as follows.*

$\mathrm{Exp}^{data\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$:
$(pk, sk) \leftarrow \mathsf{MR\text{-}SHE.KeyGen}(1^\kappa);\ \mathcal{D} \leftarrow \emptyset$
$(\omega^*, M_0^*, M_1^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
$b \xleftarrow{\$} \{0,1\};\ C^* \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega^*, M_b^*)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{C^*\};\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(C^*, st)$
if $b = b'$ *then return* 1
else return 0

st is state information that an adversary $\mathcal{A}$ can preserve any information, and st is used for transferring state information to the other stage. $\mathcal{O}$ is a set of oracles defined as $\mathcal{O} := \{\mathcal{O}^{MR\text{-}SHE}_{revhk}(pk, sk, \cdot), \mathcal{O}^{MR\text{-}SHE}_{dec}(pk, sk, \cdot\cdot), \mathcal{O}^{MR\text{-}SHE}_{trapdoor}(pk, sk, \cdot), \mathcal{O}^{MR\text{-}SHE}_{test}(pk, \cdot, \cdot), \mathcal{O}^{MR\text{-}SHE}_{eval}(pk, \cdot, \cdot, \cdot)\}$. *Each oracle is defined as follows.*

- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{revhk}}$ *: This homomorphic operation key reveal oracle takes as input a keyword $\omega \in \mathcal{W}$, and returns* $hk_\omega \leftarrow \mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$. *We remark that ω^* is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{dec}}$ *: This decryption oracle takes as input a keyword $\omega \in \mathcal{W}$ and a ciphertext C where $C \notin \mathcal{D}$, and outputs the result of* $\mathsf{MR\text{-}SHE.Dec}(pk, sk, \omega, C)$. *If ω^* has been input to the $\mathcal{O}^{MR\text{-}SHE}_{revhk}$ oracle, then this oracle returns $\perp$.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{trapdoor}}$ *: This trapdoor oracle takes as input a keyword $\omega \in \mathcal{W}$, and returns* $t_\omega \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$. *We remark that ω^* is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{test}}$ *: This test oracle takes as input a keyword $\omega \in \mathcal{W}$ and a ciphertext C, and returns the result of* $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$ *where* $t_\omega \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$. *We remark that (ω^*, C^*) is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{eval}}$ *: This evaluation oracle takes as input a keyword $\omega \in \mathcal{W}$ and two ciphertext C_1 and C_2, and returns the result of* $\mathsf{MR\text{-}SHE.Eval}(pk, hk_\omega, C_1, C_2)$ *where* $hk_\omega \leftarrow \mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$. *Moreover, if either $C_1 \in \mathcal{D}$ or $C_2 \in \mathcal{D}$, and the return of this oracle is not $\perp$ (say C), then this oracle updates $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$.*

Next, we define keyword privacy which guarantees that no information of keyword is revealed from ciphertexts. As in the case of data privacy, we need to guarantee that the homomorphic operations do not affect keyword privacy. We remark that an adversary chooses a plaintext only when the homomorphic operation is performed to the challenge ciphertext (or its related ciphertexts) due to consistency. Otherwise, the adversary always wins the game as follows: the adversary computes of a ciphertext using ω_0^* and sends it and the challenge ciphertext (encrypted by ω_b^* where $b \in \{0,1\}$) to the evaluation oracle. If the return is not $\perp$, then $b = 0$ and $b = 1$ otherwise. Moreover, due to the same reason, the adversary is not allowed to obtain homomorphic operation keys of ω_0^* and ω_1^*. That is, if the adversary is allowed to obtain the homomorphic operation keys of ω_0^* and ω_1^*, then the adversary always win the game since the adversary can evaluate the challenge ciphertext by using the homomorphic operation keys. For example, if the challenge ciphertext is computed by ω_0^*, then the evaluation algorithm with $hk_{\omega_0^*}$ outputs a ciphertext whereas the challenge ciphertext is computed by ω_1^*, then the output is $\perp$. So, these restrictions are necessary when we consider keyword privacy for an evaluated ciphertext and consistency without contradiction.

Definition 2.3 (Keyword Privacy). *For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathbb{N}$, we define the experiment* $\mathrm{Exp}^{keyword\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$ *as follows.*

$\mathrm{Exp}^{keyword\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa)$:
$(pk, sk) \leftarrow \mathsf{MR\text{-}SHE.KeyGen}(1^\kappa);\ \mathcal{D} \leftarrow \emptyset$
$(\omega_0^*, \omega_1^*, M^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
$b \xleftarrow{\$} \{0,1\};\ C^* \leftarrow \mathsf{MR\text{-}SHE.Enc}(pk, \omega_b^*, M^*)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{C^*\}\ b' \leftarrow \mathcal{A}^{\mathcal{O}}(C^*, st)$
if $b = b'$ *then return* 1
else return 0

Here, $\mathcal{O}$ is a set of oracles defined as $\mathcal{O} := \{\mathcal{O}^{MR\text{-}SHE}_{revhk}(pk, sk, \cdot), \mathcal{O}^{MR\text{-}SHE}_{dec}(pk, sk, \cdot, \cdot), \mathcal{O}^{MR\text{-}SHE}_{trapdoor}(pk, sk, \cdot), \mathcal{O}^{MR\text{-}SHE}_{test}(pk, \cdot, \cdot), \mathcal{O}^{MR\text{-}SHE}_{eval}(pk, \cdot, \cdot, \cdot), \mathcal{O}'^{MR\text{-}SHE}_{eval}(pk, \cdot, \cdot)\}$. *Each oracle is defined as follows.*

- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{revhk}}$ *: This homomorphic operation key reveal oracle takes as input a keyword $\omega \in \mathcal{W}$, and returns* $hk_\omega \leftarrow \mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$. *We remark that ω^* is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{dec}}$ *: This decryption oracle takes as input a keyword $\omega \in \mathcal{W}$ and a ciphertext C where $C \notin \mathcal{D}$, and outputs the result of* $\mathsf{MR\text{-}SHE.Dec}(pk, sk, \omega, C)$. *If ω^* has been input to the $\mathcal{O}^{MR\text{-}SHE}_{revhk}$ oracle, then this oracle returns $\perp$.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{trapdoor}}$ *: This trapdoor oracle takes as input a keyword $\omega \in \mathcal{W}$, and returns* $t_\omega \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$. *We remark that ω^* is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{test}}$ *: This test oracle takes as input a keyword $\omega \in \mathcal{W}$ and a ciphertext C, and returns the result of* $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$ *where* $t_\omega \leftarrow \mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$. *We remark that (ω^*, C^*) is allowed to be an input of this oracle.*
- $\mathcal{O}^{\mathbf{MR\text{-}SHE}}_{\mathbf{eval}}$ *: This evaluation oracle takes as input a keyword $\omega \in \mathcal{W}$ and two ciphertext C_1 and C_2, and returns the result of* $\mathsf{MR\text{-}SHE.Eval}(pk, hk_\omega, C_1, C_2)$ *where* $hk_\omega \leftarrow \mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$. *Moreover, if either $C_1 \in \mathcal{D}$ or $C_2 \in \mathcal{D}$, and the return of this oracle is not $\perp$ (say C), then this oracle updates $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$.*

We say that $\mathcal{MR}\text{-}\mathcal{SHE}$ is keyword private if the advantage $\mathrm{Adv}^{keyword\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa) := |\Pr[\mathrm{Exp}^{keyword\text{-}privacy}_{\mathcal{MR}\text{-}\mathcal{SHE},\mathcal{A}}(\kappa) = 1] - 1/2|$ *is negligible for any PPT adversary $\mathcal{A}$.*

3. PROPOSED MR-SHE SCHEME

3.1 High-level Description

First, we briefly introduce the Gentry IBE scheme [19] and the KH-IBE scheme [14] which is based on the Gentry IBE scheme as follows. Let a plaintext M be encrypted by an identity ID and a randomness s. Then, $(c_1, c_2, c_3, c_4) = (g_1^s g^{-s\mathsf{ID}}, e(g,g)^s, M \cdot e(g,h_1)^{-s}, e(g,h_2)^s e(g,h_3)^{s\beta})$ is a ciphertext of the Gentry IBE scheme where $\beta = \Gamma_{hk}(c_1, c_2, c_3)$ and Γ_{hk} is a target collision resistance (TCR) hash function. The 4th component, $e(g,h_2)^s e(g,h_3)^{s\beta}$, is used for integrity check for achieving the CCA security. In the KH-IBE scheme, c_4 is modified to be $c_4 = e(g,h_2)^s$ (which is explained later) and a new component $\tau = f(c_5)$ is added to a ciphertext where $c_5 = e(g,h_3)^s e(g,h_4)^{s\beta}$, $\beta = \Gamma_{hk}(c_1, c_2, c_3, c_4)$, and f is a smooth function. The smoothness is introduced for compressing the size of a ciphertext. As in the Gently IBE, c_5 is required for integrity check and for achieving the CCA security. A decryption key of ID is $(r_{\mathsf{ID},i}, h_{\mathsf{ID},i})$ where $h_{\mathsf{ID},i} = (h_i g^{-r_{\mathsf{ID},i}})^{1/(\alpha-\mathsf{ID})}$ $(i = 1,2,3,4)$ and α is the master secret key managed by the key generation center. Then, a part of decryption key $((r_{\mathsf{ID},3}, h_{\mathsf{ID},3}), (r_{\mathsf{ID},4}, h_{\mathsf{ID},4}))$ is regarded as a homomorphic operation key since one who has the key can run the integrity check of a ciphertext, where $\tau = f(e(c_1, h_{\mathsf{ID},3} h_{\mathsf{ID},4}^{\delta}) c_2^{r_{\mathsf{ID},3} + r_{\mathsf{ID},4}\delta})$ or not, and can reconstruct a ciphertext after homomorphic operation. In the meantime, one cannot decrypt the ciphertext since it requires $(r_{\mathsf{ID},1}, h_{\mathsf{ID},1})$. Due to $(r_{\mathsf{ID},2}, h_{\mathsf{ID},2})$ and c_4, a CCA1 (i.e., lunch-time) security is guaranteed against ones who have the homomorphic operation key only.

Next, we explain our strategy to construct MR-SHE as follows. As in the Abdalla et al. construction [2], in our scheme a keyword is regarded as an identity of the KH-IBE scheme. In the Abdalla et al. construction a random plaintext (say R) is encrypted, a decryption key is regarded as a trapdoor, and the test algorithm returns 1 if the decryption result of the IBE ciphertext by using the trapdoor is R. So, R is required to be a part of ciphertext. In the MR-SHE usage, the corresponding plaintext needs to be hidden. To overcome this problem, we pointed out the integrity check can be used for secure searching since this procedure essentially checks whether a ciphertext is valid under a specific ID or not. This allows us to run the test algorithm without knowing the corresponding plaintext. So, a homomorphic operation key of the KH-IBE scheme is regarded as a trapdoor, and the integrity check equation of the evaluation algorithm of KH-IBE is employed to be the test algorithm. Due to this setting, we can encrypt a plaintext M, and M is not required to be a part of a ciphertext.

Other problem to be solved is that the corresponding identity ID is required for the evaluation algorithm of the original KH-IBE scheme. In our usage, if a keyword ω is required for evaluation, then, no keyword privacy is guaranteed. For achieving keyword privacy, we add g^{ω} to a homomorphic operation key for evaluating ciphertexts without knowing the corresponding keyword ω. Then the server that runs the evaluation algorithm can compute $g_1^s g^{-s\omega} = (g_1^{-1} g^{\omega})^{-s}$ by choosing $s \xleftarrow{\$} \mathbb{Z}_p$. We emphasize that this modification does not affect anonymity (and other security requirements). We will explain the reason later.

3.2 Our Construction

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, $\mathcal{W} := \mathbb{Z}_p$ be the keyword space, $\mathcal{M} := \mathbb{G}_T$ be the message space, $\{\Gamma = \Gamma_{hk} : \mathbb{G}^4 \to \{0, 1, \ldots, p-1\} \mid hk \in \mathcal{HK}\}$ be a TCR hash family, and $f : \mathbb{G}_T \to \mathcal{Y}$ is a smooth function.

In the KH-IBE scheme and the original Gentry IBE scheme, assume that there is only one key for each identity due to the security proofs. So, we also assume that keys $\{(r_{\omega,i}, h_{\omega,i})\}_{i=1,2,3,4}$ are generated only once, and each algorithm shares these keys. For example, $\{(r_{\omega,i}, h_{\omega,i})\}_{i=1,2,3,4}$ is the decryption key and a part of this key, $\{(r_{\omega,i}, h_{\omega,i})\}_{i=3,4}$, is the homomorphic operation key or the trapdoor. So, our scheme is stateful. For the sake of simplicity, we give algorithms when these are run for the first time. Our proposed scheme is given as follows.

$\mathsf{MR\text{-}SHE.KeyGen}(1^{\kappa})$: Choose $hk \xleftarrow{\$} \mathcal{HK}$, $g \xleftarrow{\$} \mathbb{G}$, $h_1, h_2, h_3, h_4 \xleftarrow{\$} \mathbb{G}$, and $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and compute $g_1 \leftarrow g^{\alpha}$. Return $pk = (g, g_1, h_1, h_2, h_3, h_4, hk, f)$ and $sk = \alpha$.

$\mathsf{MR\text{-}SHE.HomKeyGen}(pk, sk, \omega)$: For $i = 3, 4$, choose $r_{\omega,i} \xleftarrow{\$} \mathbb{Z}_p$ and compute $h_{\omega,i} \leftarrow (h_i g^{-r_{\omega,i}})^{1/(\alpha-\omega)}$, and return $hk_{\omega} = (g^{\omega}, (r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$.

$\mathsf{MR\text{-}SHE.Trapdoor}(pk, sk, \omega)$: For $i = 3, 4$, choose $r_{\omega,i} \xleftarrow{\$} \mathbb{Z}_p$ and compute $h_{\omega,i} \leftarrow (h_i g^{-r_{\omega,i}})^{1/(\alpha-\omega)}$, and return $t_{\omega} = (g^{\omega}, (r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$.

$\mathsf{MR\text{-}SHE.Enc}(pk, \omega, M)$: Choose $s \xleftarrow{\$} \mathbb{Z}_p$ and compute $c_1 \leftarrow g_1^s g^{-s\omega}$, $c_2 \leftarrow e(g,g)^s$, $c_3 \leftarrow M \cdot e(g,h_1)^{-s}$, $c_4 \leftarrow e(g,h_2)^s$, $\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4)$, and $c_5 \leftarrow e(g,h_3)^s e(g,h_4)^{s\delta}$. Compute $\tau \leftarrow f(c_5)$ and return $C = (c_1, c_2, c_3, c_4, \tau)$.

$\mathsf{MR\text{-}SHE.Test}(pk, t_{\omega}, C)$: Parse t_{ω} as $(g^{\omega}, (r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$ and C as $(c_1, c_2, c_3, c_4, \tau)$. Compute $\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4)$. If $\tau = f(e(c_1, h_{\omega,3} h_{\omega,4}^{\delta}) c_2^{r_{\omega,3} + r_{\omega,4}\delta})$ holds, then return 1 and 0 otherwise.

$\mathsf{MR\text{-}SHE.Dec}(pk, sk, \omega, C)$: Parse $sk = \alpha$ and $C = (c_1, c_2, c_3, c_4, \tau)$. For $i = 1, 2, 3, 4$, choose $r_{\omega,i} \xleftarrow{\$} \mathbb{Z}_p$ and compute $h_{\omega,i} \leftarrow (h_i g^{-r_{\omega,i}})^{1/(\alpha-\omega)}$. Compute $\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4)$, $c_4' \leftarrow e(c_1, h_{\omega,2}) c_2^{r_{\omega,2}}$, and $c_5 \leftarrow e(c_1, h_{\omega,3} h_{\omega,4}^{\delta}) c_2^{r_{\omega,3} + r_{\omega,4}\delta}$. If $c_4' \neq c_4$ or $\tau \neq f(c_5)$ then return $\perp$. Otherwise, return $M \leftarrow c_3 \cdot e(c_1, h_{\omega,1}) c_2^{r_{\omega,1}}$.

$\mathsf{MR\text{-}SHE.Eval}(pk, hk_{\omega}, C_1, C_2)$: Parse hk_{ω} as $(g^{\omega}, (r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$, C_1 as $(c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, \tau_1)$, and C_2 as $(c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}, \tau_2)$.

Integrity Check: Compute $\delta_1 \leftarrow \Gamma_{hk}(c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4})$, $c_{1,5} = e(c_{1,1}, h_{\omega,3} h_{\omega,4}^{\delta_1}) c_{1,2}^{r_{\omega,3} + r_{\omega,4}\delta_1}$, $\delta_2 \leftarrow \Gamma_{hk}(c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4})$, and $c_{2,5} = e(c_{2,1}, h_{\omega,3} h_{\omega,4}^{\delta_2}) c_{2,2}^{r_{\omega,3} + r_{\omega,4}\delta_2}$. If $\tau_1 \neq f(c_{1,5})$ or $\tau_2 \neq f(c_{2,5})$ then return $\perp$.

Homomorphic Operation: Choose $s \xleftarrow{\$} \mathbb{Z}_p$, and compute $c_1 \leftarrow c_{1,1} c_{2,1} \cdot g_1^s g^{-s\omega}$, $c_2 \leftarrow c_{1,2} c_{2,2} \cdot e(g,g)^s$, $c_3 \leftarrow c_{1,3} c_{2,3} \cdot e(g,h_1)^{-s}$, $c_4 \leftarrow c_{1,4} c_{2,4} \cdot e(g,h_2)^s$, $\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4)$, $c_5 \leftarrow e(c_1, h_{\omega,3} h_{\omega,4}^{\delta}) c_2^{r_{\omega,3} + r_{\omega,4}\delta}$, and $\tau \leftarrow f(c_5)$, and return $C = (c_1, c_2, c_3, c_4, \tau)$.

Here, we explain the reason why our modification, adding g^{ω} to a trapdoor and a homomorphic operation key, does not affect anonymity as follows. In the definition of anonymity, an adversary chooses challenge keywords ω_0^* and ω_1^*. Even though, the adversary cannot distinguish whether the challenge ciphertext is computed by ω_0^* or ω_1^*. That is, the adversary can compute $g^{\omega_0^*}$ and $g^{\omega_1^*}$ for the first place, and therefore our modification does not affect anonymity. Nevertheless, one may think that the adversary (or the server) can check whether a trapdoor (or a homomorphic operation key) is computed by a keyword ω' or not by checking $g^{\omega'}$ is equal to g^{ω} or not. This observation is true. However, the server can do the same thing even if the server is not allowed to obtain g^{ω} since the server is allowed to search ciphertexts. First, the server chooses a keyword ω' and computes a ciphertext associated with ω'. Second the server runs the test algorithm by using a trapdoor sent from the receiver. Finally, the server knows whether the trapdoor is computed by ω' or not from the evaluation result. Since we delegate the search capability to the server, this information leakage is acceptable, and theoretically impossible to be handled, even if searchable encryption secure keyword guessing attacks [29, 16] is employed.

One may have a concern if the evaluation algorithm forcibly performs the homomorphic operation even though $C_1 := (g_1^s g^{-s\omega}, \cdots)$ and $C_2 := (g_1^{s'} g^{-s'\omega'}, \cdots)$ are encrypted by using different keywords ω and ω', respectively. In this case, the first component of the ciphertext c_1 is represented as $c_1 := g_1^{s+s'+s''} g^{-(s+s''\omega-s'\omega')}$ for some randomness s'' chosen in the evaluation algorithm. Then c_1 can be represented as $c_1 = g_1^{s+s'+s''} g^{-(s+s'+s'')\omega''}$ where $\omega'' := -((s+s')\omega + s'\omega')/(s+s'+s'')$ is an unknown keyword. Let a receiver send a query for ciphertexts associated with ω. Then, it is natural that the receiver decrypts returned ciphertexts by using $\{(r_{\omega,i}, h_{\omega,i})\}_{i=1,2,3,4}$.[4] Then, since an artificially-mixed ciphertext is associated with a random keyword ω'', the decryption algorithm rejects it with overwhelming probability, and the receiver can recognize whether a mis-operation happens in the evaluation phase.

3.3 Security Analysis

Security proofs of following theores are given in the Appendix.

Theorem 3.1 (Consistency). *The proposed MR-SHE scheme is consistent if f is a smooth function and the discrete logarithm assumption holds.*

[4]Since the receiver has the master key α, the receiver may obtain the corresponding keyword of artificially-mixed ciphertexts, and may be able to decrypt them. For preventing such a decryption, IBE with anonymity against key generation center (KGC) that has the master key could be employed [22]. Moreover, IBE with anonymous ciphertext indistinguishability [12], which guarantees that no information of plaintext is revealed from a ciphertext encrypted by an unknown identity even against KGC, also could be employed. However, since the receiver requests the homomorphic operation to the server for ciphertexts associated with a specific keyword ω, it seems natural to assume that the receiver decrypts returned ciphertexts by using $\{(r_{\omega,i}, h_{\omega,i})\}_{i=1,2,3,4}$. Thus, we assume that no receiver tries to decrypt such artificially-mixed ciphertexts, and employing such IBEs for enhancing security is left as a future work of this paper.

Theorem 3.2 (Data Privacy). *The MR-SHE scheme is data private if the truncated decision q-ABDHE assumption holds.*

Theorem 3.3 (Keyword Privacy). *The MR-SHE scheme is keyword private if the truncated decision q-ABDHE assumption holds.*

One may expect that MR-SHE can be generically constructed from anonymous KH-IBE, as in the Abdalla et al. construction [2]. However, such a construction is not consistent in the sense of provable security. More precisely, in the proof of the Abdalla et al. construction we can obtain a decryption key from an adversary of consistency that can decrypt the challenge ciphertext since a trapdoor is a decryption key. On the other hand, since a trapdoor is just a part of a decryption key, we do not decrypt the challenge ciphertext even if the adversary breaks consistency of our MR-SHE scheme. It seems some other conditions are required for a generic construction, and it is an interesting future work of this paper.

4. EVALUATING MULTIPLE CIPHERTEXTS BY SINGLE EXECUTION

In the MR-SHE.Eval algorithm, two ciphertexts, C_1 and C_2, are input. In this section, we discuss how to evaluate multiple (i.e., more than two) ciphertexts $(C_1, C_2, \ldots, C_L)$ where L is a polynomial of the security parameter, by single execution as follows. We call this algorithm MR-SHE.mEval.

MR-SHE.mEval$(pk, hk_{\omega}, \{C_i\}_{i=1}^{L})$: Parse hk_{ω} as $(g^{\omega}, (r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$ and C_i as $(c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}, \tau_i)$ for $i = 1, \ldots, L$.

Integrity Check: If there exixt $i \in [1, L]$ such that $\tau_i \neq f(c_{i,5})$, where $\delta_i \leftarrow \Gamma_{hk}(c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4})$ and $c_{i,5} = e(c_{i,1}, h_{\omega,3} h_{\omega,4}^{\delta_1}) c_{i,2}^{r_{\omega,3}+r_{\omega,4}\delta_i}$, then return $\perp$.

Homomorphic Operation: Choose $s \xleftarrow{\$} \mathbb{Z}_p$, and compute

$$c_1 \leftarrow g_1^s g^{-s\omega} \prod_{i=1}^{L} c_{i,1},\ c_2 \leftarrow e(g,g)^s \prod_{i=1}^{L} c_{i,2}$$

$$c_3 \leftarrow e(g,h_1)^{-s} \prod_{i=1}^{L} c_{i,3},\ c_4 \leftarrow e(g,h_2)^s \prod_{i=1}^{L} c_{i,4}$$

$$\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4),\ c_5 \leftarrow e(c_1, h_{\omega,3} h_{\omega,4}^{\delta}) c_2^{r_{\omega,3}+r_{\omega,4}\delta}$$

and return $C = (c_1, c_2, c_3, c_4, \tau)$ where $\tau \leftarrow f(c_5)$.

Before analyzing the MR-SHE.mEval agorithm, we investigate the original MR-SHE.Eval algorithm as follows. In the MR-SHE.Eval algorithm, a new randomness $s \in \mathbb{Z}_p$ is chosen. Essentially, this randomness is used for computing a ciphertext (excluding the fifth component) of $M = 1$, $(g_1^s g^{-s\omega}, e(g,g)^s, e(g,h_1)^{-s}, e(g,h_2)^s)$, which does not affect the result of homomorphic operation. This randomness is indispensable for achieving the source ciphertext hiding property of KH-IBE [14]: for a ciphertext C_1, C_1', C_2 where all ciphertexts are computed by the same identity and C_1 and C_1' are ciphertexts of the same plaintext, the distributions of the evaluation result of (C_1, C_2) and (C_1', C_2) are identical. This

property is employed to replace challenge-related ciphertexts into harmless ciphertexts in the security proof. Let $\mathcal{D} = (C_0, C_1, \ldots, C_k)$ be the dictionary of challenge-related ciphertexts. Moreover, other dictionary $\mathcal{D}'$ is managed in the proof where $\mathcal{D}' := ((D'_1, D''_1), (D'_2, D''_2), \ldots, (D'_k, D''_k))$. Each of D'_i and D''_i ($i \in [1, \kappa]$) is either a ciphertext with fifth component being consistent or an index in $\{0, 1, \ldots, i-1\}$. $(D'_i, D''_i) \in \mathcal{D}'$ means that $C_i \in \mathcal{D}$ was the reply to the evaluation query (D'_i, D''_i). Here, if D'_i or D''_i is an index j, then it is interpreted as C_j. By using $\mathcal{D}'$ and the source ciphertext hiding property, each challenge-related ciphertexts are newly computed.

Due to the proof methodology above, what we need to consider is (1) the source ciphertext hiding property, and (2) the size of $\mathcal{D}'$. Fortunately, the source ciphertext hiding property trivially holds. Moreover, $\mathcal{D}'$ is represented as $\mathcal{D}' := ((D_1^{(1)}, D_1^{(2)}, \ldots, D_1^{(L)}), \ldots, (D_k^{(1)}, D_k^{(2)}, \ldots, D_k^{(L)}))$ and its size is still polynomial of the security parameter. That is, our modification does not affect the security proof. We use the MR-SHE.mEval algorithm in our implementations.

5. IMPLEMENTATION RESULTS

5.1 Basic Operations

We employ the PBC library [1] and $y^2 = x^3 + x$ (Type A, defined on a 512-bit prime field, to provide 80-bit security level) as the underlying elliptic curve. Our implementation environment is: CPU: Xeon E5-2660 v3 @ 2.60GHz, gcc 4.9.2, openssl 1.0.2d, and pbc-0.5.14. We give benchmarks of basic operations in Table 2.

Table 2: Basic Operations

Operations	Time (msec)
Scalar mul. in $\mathbb{G}$ (unknonw point)	1.351
Scalar mul. in $\mathbb{G}$ (fixed point)	0.191
Exp. in $\mathbb{G}_T$ (unknonw element)	0.114
Exp. in $\mathbb{G}_T$ (fixed element)	0.023
Pairing computation	0.840

5.2 Storage Size and Communication Overhead

The sizes of a scalar value in $\mathbb{Z}_p$, an element in $\mathbb{G}$, an element in $\mathbb{G}_T$, and a hash value are 20 bytes, 65 bytes (using `element_to_bytes_compressed()`), 128 bytes, and 64 bytes, respectively. Here, we use SHA512 as Γ and f. With these values, sizes of datas in our scheme can be estimated (Table 3).

Table 3: Sizes of Data in Our Scheme

	Components	Size (bytes)
M	$\lvert\mathbb{G}_T\rvert$	128
M (Lifted ElGamal)	1 small integer	4–8
C	$\lvert\mathbb{G}\rvert + 3\lvert\mathbb{G}_T\rvert + 1$ hash value	513
sk	$\lvert\mathbb{Z}_p\rvert$	20
pk	$6\lvert\mathbb{G}\rvert$	390
hk_ω $(=t_\omega)$	$2\lvert\mathbb{Z}_p\rvert + 3\lvert\mathbb{G}\rvert$	235

Ciphertext is commonly communicated among participants and stored in a server, as shown in Figure 1. Therefore its size affects both communication overhead and storage size. The size is 513 bytes, relatively large for the 80-bit security setting[5], and might be a bottleneck for a large scale setting. However, this is a common problem of public key homomorphic encryption schemes, and the solution of this problem is beyond the scope of this paper. We should remark about the size of homomorphic keys, because they are stored in a server once they are generated for keywords. The size of keys is $235N_\omega$ bytes, where N_ω is the number of keywords.

5.3 Algorithms

We give benchmarks of algorithms in Table 4 where all algorithms work in the milliseconds order. In our implementations, all precomputable pairings, $e(g, h_i)$ for $i = 1, 2, 3, 4$, are computed in advance.

Table 4: Benchmarks of Algorithms

Algorithm	Time (msec)	Entity
MR-SHE.KeyGen	9.5	Receiver
MR-SHE.HomKeyGen	1.0	Receiver
(MR-SHE.Trapdoor)	(1.0)	(Receiver)
MR-SHE.Enc	0.5	Data Holder
MR-SHE.Dec	4.7	Receiver
MR-SHE.Test	2.2	Server
MR-SHE.Eval	8.1	Server

If the input ciphertexts of the MR-SHE.Eval algorithm have been tested by the MR-SHE.Test algorithm before, then the ciphertexts are guaranteed that these are associated with the same keyword. So, in this case the integrity check procedure of the MR-SHE.Eval algorithm can be skipped, and the MR-SHE.Eval algorithm can start from the homomorphic operation procedure. Since the integrity check procedure is run for two ciphertexts in the MR-SHE.Eval algorithm, the MR-SHE.Eval algorithm can be run approximately 3.7 msec in this situation.

Next, we give the benchmarks of MR-SHE.mEval algorithm in Fig 2. Here, $L = 4, 8, 16, \ldots, 8192$ $(= 2^{13})$ is the number of input ciphertexts in the algorithm.

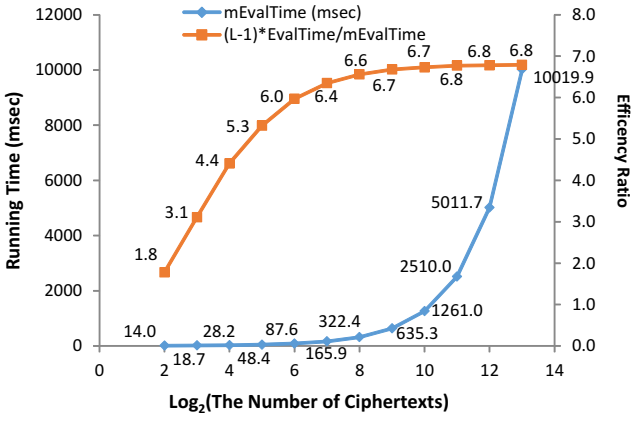

Figure 2: Benchmarks of MR-SHE.mEval

Due to our implementation result, the running time of the MR-SHE.mEval algorithm is asymptotically seven-times faster

[5]This will be larger when Catalano-Fiore transformation is employed, because, after computing inner product for ℓ-dimensional vectors, the resultant ciphertext has $2\ell + 1$ ciphertext components. See Section 5.4.2 for details.

than that of $(L-1)$-times executions of the MR-SHE.Eval algorithm.

5.4 Evaluating Inner Products of Encrypted Vectors

Next, we give our implementation result of inner products of encrypted vectors. We employ the lifted ElGamal-like additive homomorphic operations as in [13]. Moreover, for the case that both vectors are encrypted, we employ the Catalano-Fiore transformation [10] for supporting one multiplicative homomorphic operation.

5.4.1 Catalano-Fiore Transformation

Briefly, the Catalano-Fiore transformation is explained as follows. Let $\mathcal{HE} := (\mathsf{Enc}, \mathsf{Dec})$ is an additive HE (here we omit the public/secret keys for the sake of simplicity). For encrypting a plaintext $M \in \mathcal{M}$, where $\mathcal{M}$ is a plaintext space and is required to be a ring (i.e., supporting additions and multiplications), randomly choose $b \xleftarrow{\$} \mathcal{M}$, compute $a = M - b$ (here "$-$" is subtraction over the ring $\mathcal{M}$) and $\mathsf{Enc}(b)$, and the ciphertext is $(a, \mathsf{Enc}(b))$. Let $(a_1 := M_1 - b_1, \mathsf{Enc}(b_1))$ and $(a_2 := M_2 - b_2, \mathsf{Enc}(b_2))$ be two ciphertexts. Then, compute a_1a_2 and $\mathsf{Enc}(a_1a_2)+a_1\mathsf{Enc}(b_2)+a_2\mathsf{Enc}(b_1) = \mathsf{Enc}(M_1M_2 - b_1b_2)$. Here "$+$" is the additive homomorphic operation of $\mathcal{HE}$. The resultant ciphertext is $(\mathsf{Enc}(M_1M_2-b_1b_2), \mathsf{Enc}(b_1), \mathsf{Enc}(b_2))$. For computing M_1M_2, the Dec algorithm computes b_1 and b_2 by decrypting $(\mathsf{Enc}(b_1), \mathsf{Enc}(b_2))$, computes b_1b_2, computes $M_1M_2-b_1b_2$ by decrypting $\mathsf{Enc}(M_1M_2-b_1b_2)$, and finally computes M_1M_2.

5.4.2 Employing the Catalano-Fiore Transformation

Let $G := e(g,g)$. For encrypting a plaintext $M \in \mathbb{Z}_t$, randomly choose $b \xleftarrow{\$} \mathbb{Z}_T$ for $T \geq t$, compute $a = M - b \bmod T$ and $C = \mathsf{MR\text{-}SHE.Enc}(pk, \omega, G^b)$, and the ciphertext is (a, C). Let $(a_1 := M_1-b_1, C_1)$ and $(a_2 := M_2-b_2, C_2)$ be two ciphertexts. The multiplication of these ciphertexts is $(\mathsf{MR\text{-}SHE.Enc}(pk, \omega, G^{M_1M_2-b_1b_2 \bmod T}), C_1, C_2)$, whose first component is computable by $\mathsf{MR\text{-}SHE.Enc}(pk, \omega, a_1a_2)+a_1C_2+a_2C_1$. Here "$+$" means an evaluation by MR-SHE.Eval, and the multiplication of a scalar $a \in \mathbb{Z}_T$ and a ciphertext C means a-times evaluations of C. Then we can obtain M_1M_2 by decrypting each component and computing these discrete logarithms.

Using the above multiplicative homomorphic operation, a vector inner product can be computed as follows. Let $\boldsymbol{x}_0 := (x_{0,1}, \ldots, x_{0,\ell}) \in \mathbb{Z}_t^\ell$ and $\boldsymbol{x}_1 := (x_{1,1}, \ldots, x_{1,\ell}) \in \mathbb{Z}_t^\ell$ be two ℓ-dimensional vectors, and each encrypted vectors are $((a_{i,1}, C_{i,1}), \ldots, (a_{i,\ell}, C_{i,\ell}))$ for $i = 0, 1$. Then the inner product of the encrypted vectors is $(\sum_{j=1}^{\ell} \mathsf{MR\text{-}SHE.Enc}(pk, \omega, G^{x_{0,j}x_{1,j}-b_{0,j}b_{1,j} \bmod T}), \{C_{0,j}, C_{1,j}\}_{j=1}^{\ell})$, which contains $2\ell+1$ components. Decrypting $2\ell+1$ components and computing these discrete logarithms, the desired inner product $\sum_{j=1}^{\ell} x_{0,j}x_{1,j}$ can be computed.

As a remark, since the above inner product operation is over $\mathbb{Z}_T$, but we want to compute it over $\mathbb{Z}$, the modulus T must be larger than the result of inner product. For ℓ-dimensional vector of plaintext space $\mathbb{Z}_t$, the inner product is at most $\ell(t-1)^2$, so we set $T = \ell(t-1)^2 + 1$. Then the maximum value of the discrete logarithm in the inner product computation is $\mathsf{maxdl} = 3\ell(T-1)^2 = 3\ell^3(t-1)^4$. For example, the inner product of 1024-dimensional binary vectors requires $T = 1025$ and $\mathsf{maxdl} \approx 2^{31.6}$. We employ Baby-step Giant-step approach for computing the discrete logarithm, and limit maxdl to 2^{41} for our experiments.

For preventing that one encrypts non-$\mathbb{Z}_t$ elements, we can employ the technique of restrictive PKE [30] which guarantees that a plaintext of a ciphertext is in $\mathbb{Z}_t$ by adding non-interactive zero-knowledge proofs of membership, as in the attempt of [31]. In our implementation, simply we assume that $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathbb{Z}_t^\ell$. Moreover, due to the Catalano-Fiore transformation, a ciphertext additionally consists of $a_{0,j}$ and $a_{1,j}$ respectively. So, we need to slightly modify the security definition in order to rule out a trivial CCA attacks, where an adversary is not allowed to issue a decryption query (a, C) if $C \in \mathcal{D}$. To avoid such an undesirable modification, fully or somewhat homomorphic property is required. We leave these topics as future works of this paper.

5.4.3 Implementation Result: Both Vectors are Encrypted

We show the implementation result of our scenario given in Section 1.5 in Fig 3 where two vectors associated with a disease name are encrypted by a public key of the researcher, and the server computes its inner product. In this setting, the hospitals can fully delegate the computation of ciphertexts of inner products to the server. Here, we treat the case of binary vectors ($t = 2$). Let ℓ be the dimension of vectors (which is the number of total subjects n in the scenario), and the x-axis indicates $\log_2(\ell)$. We show the cases of $\ell = 4, 8, 16, \ldots, 8192 = 2^{13}$, respectively. IP Time means that the running time of computing a ciphertext of the inner product $(\sum_{j=1}^{\ell} \mathsf{MR\text{-}SHE.Enc}(pk, \omega, G^{x_{0,j}x_{1,j}-b_{0,j}b_{1,j} \bmod T}), \{C_{0,j}, C_{1,j}\}_{j=1}^{\ell})$. We show the actual IP Time in Fig 3. Dec after IP means that the running time to decrypt the ciphertext and obtain the inner product $\sum_{i=1}^{\ell} x_{0,i}x_{1,i} \bmod T$. We omit the actual running times of Dec after IP. Del ratio means how much a receiver (the researcher) can delegate the computation of inner products to the server, i.e., IP Time/(IP Time+Dec after IP).

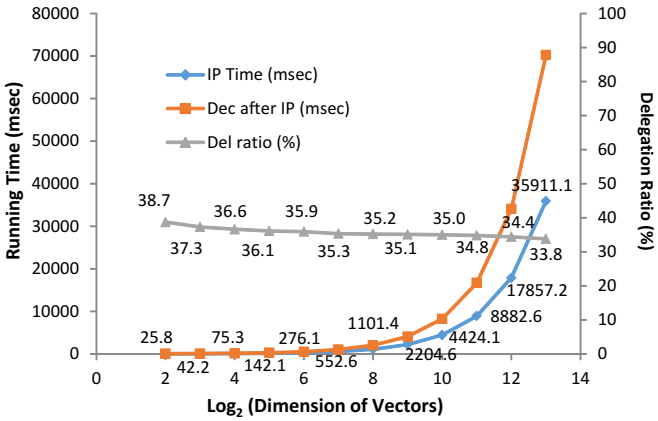

Figure 3: Benchmarks (Both Vectors are Encrypted)

A ciphertext of inner products can be efficiently computed by the server even for high-dimensional vectors. For example, for 8192-dementional vectors, the server can compute the ciphertext of the inner product approximately 35 seconds. One drawback of this setting is the decryption costs since the number of decryptions depends on the vector dimension ℓ due to the Catalano-Fiore transformation, and the delegation ratio is approximately 35%. Nevertheless,

our scheme seems efficient in practice. For example, for 8192-dementional vectors, the researcher can decrypt the ciphertext approximately 70 seconds. For relatively small-dimensional vectors, e.g., $\ell < 2^6$, both computation of an inner product of ciphertexts and that of a decryption are within millisecond order.

5.4.4 *Implementation Result: One Vector is Encrypted*

Next, we give benchmarks of inner products when a vector $\boldsymbol{x}_0$ is encrypted but the other vector $\boldsymbol{x}_1$ is not encrypted in Fig 4. In this setting[6], the hospital 1 has a role of the server. The hospital 2 encrypts a vector with a keyword, and sends the encrypted vectors to the hospital 1. The hospital 1 checks whether all components of the encrypted vector are associated with the same keyword, runs the MR-SHE.mEval algorithm, and sends the resultant ciphertext to the researcher. Mis-operation resistant property works for preventing that the hospital 1 performs homomorphic operations against ciphertexts associated with different keywords. Though the hospital 1 cannot delegate the computation of ciphertexts of inner products to the server, one advantage of this setting is that additive homomorphic property is enough to compute inner products, and the researcher needs to run the decryption algorithm only once (which requires approximately 5 msec) regardless of the dimension of vectors ℓ. So, the delegation ratio becomes higher according to increase of ℓ. Moreover, we do not have to modify the security definition in contrast to the case of employing the Catalano-Fiore transformation. So, we can say that the proposed scheme is practically efficient in this setting.

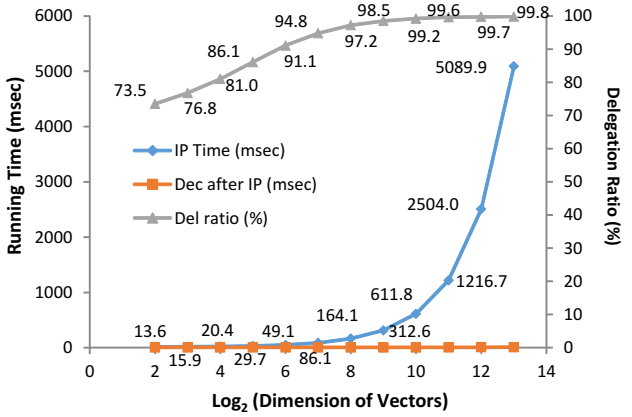

Figure 4: Benchmarks (One Vector is Encrypted)

5.5 Evaluation for Tested Ciphertexts

The MR-SHE.mEval algorithm also can skip the integrity check procedure if the input ciphertexts have been tested since then the ciphertexts are guaranteed that these are associated with the same keyword. In this section, we give the ratio of the integrity check procedure in the MR-SHE.mEval algorithm and in the computation of the ciphertext of inner products in Fig 5, and show that how much computations can be skipped for tested ciphertexts. For the MR-SHE.mEval algorithm, the x-axis indicates $\log_2(L)$ where L is the number of input ciphertexts, and for other cases, the x-axis indicates $\log_2(\ell)$ where ℓ is the dimension of vectors. P and C means the computation of inner products when a vector $\boldsymbol{x}_0$ is encrypted but the other vector $\boldsymbol{x}_1$ is not encrypted. C and C means that the computation of inner products when both vectors are encrypted. Even for the case that both vectors are encrypted, the occupancy of the integrity check procedure is more than 50 %. In other cases, almost computations are subject to occupancy by integrity check procedure. So, in the actual situation, where firstly the server searches ciphertext and secondly the server performs homomorphic operations against ciphertexts which have been guaranteed that they are associated with the same keyword, we can skip the majority of the computations.

[6]This setting is employed by Shimizu et al. [31] for evaluating a distance among chemical compounds.

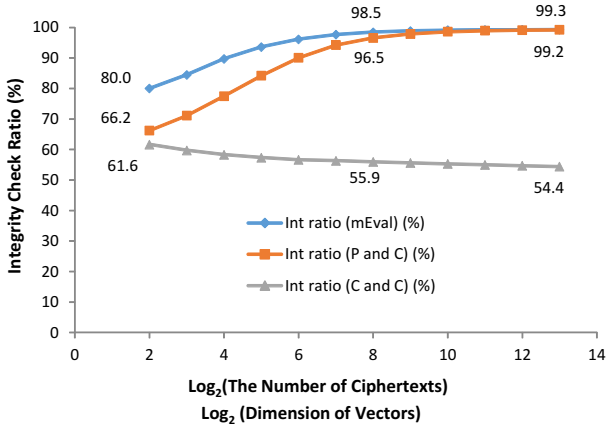

Figure 5: Integrity Check Ratio

6. CONCLUSION

In this paper, we attach great importance to a controllability of homomorphic operations, and propose MR-SHE. A keyword is regarded as a tag, and homomorphic operations are allowed for ciphertexts when these ciphertexts are associated with the same keyword. In addition to this, the evaluation algorithm posts alert to homomorphic operations for ciphertexts associated with different keywords. Even if one intentionally or accidentally performs the homomorphic operations against such ciphertexts, the decryption algorithm rejects it, and the receiver can recognize a mis-operation happens in the evaluation phase. Moreover, our scheme supports secure keyword search. We also construct a MR-SHE scheme by modifying the Gentry-based KH-IBE scheme. We also give the implementation results of computing inner products. Naively supporting fully or somewhat homomorphic property and supporting the verifiability of the computation result as in verifiable computations in the MR-SHE context are future works of this paper.

Acknowledgment: This work was partially supported by JSPS KAKENHI Grant Numbers JP26540003, JP16H02864, JP15K00028.

7. REFERENCES

[1] The PBC (pairing-based cryptography) library. Available at http://crypto.stanford.edu/pbc/.

[2] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven,

P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptology*, 21(3):350–391, 2008.

[3] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In *TCC*, pages 480–497, 2010.

[4] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *ACM Conference on Computer and Communications Security*, pages 863–874, 2013.

[5] J. Baek, R. Safavi-Naini, and W. Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *ISC*, pages 217–232, 2006.

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[7] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. Private database queries using somewhat homomorphic encryption. In *ACNS*, pages 102–118, 2013.

[8] D. Boneh, G. Segev, and B. Waters. Targeted malleability: homomorphic encryption for restricted computations. In *Innovations in Theoretical Computer Science*, pages 350–366, 2012.

[9] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.

[10] D. Catalano and D. Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In *ACM Conference on Computer and Communications Security*, pages 1518–1529, 2015.

[11] Y. Chen, J. Zhang, D. Lin, and Z. Zhang. Generic constructions of integrated PKE and PEKS. *Des. Codes Cryptography*, 78(2):493–526, 2016.

[12] S. S. M. Chow. Removing escrow from identity-based encryption. In *Public Key Cryptography*, pages 256–276, 2009.

[13] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, pages 103–118, 1997.

[14] K. Emura, G. Hanaoka, K. Nuida, G. Ohtake, T. Matsuda, and S. Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption. Cryptology ePrint Archive, Report 2013/390, 2013. http://eprint.iacr.org/2013/390.

[15] K. Emura, G. Hanaoka, G. Ohtake, T. Matsuda, and S. Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption. In *Public-Key Cryptography*, pages 32–50, 2013.

[16] L. Fang, W. Susilo, C. Ge, and J. Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.*, 238:221–241, 2013.

[17] D. Fiore, R. Gennaro, and V. Pastro. Efficiently verifiable computation on encrypted data. In *ACM Conference on Computer and Communications Security*, pages 844–855, 2014.

[18] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[19] C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[20] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[21] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

[22] M. Izabachène and D. Pointcheval. New anonymity notions for identity-based encryption. In *SCN*, pages 375–391, 2008.

[23] C. S. Jutla and A. Roy. Dual-system simulation-soundness with applications to UC-PAKE and more. In *ASIACRYPT*, pages 630–655, 2015.

[24] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC*, pages 581–600, 2006.

[25] M. Kim, H. T. Lee, S. Ling, S. Q. Ren, B. H. M. Tan, and H. Wang. Better security for queries on encrypted databases. Cryptology ePrint Archive, Report 2016/470, 2016. http://eprint.iacr.org/2016/470.

[26] J. Lai, R. H. Deng, C. Ma, K. Sakurai, and J. Weng. CCA-secure keyed-fully homomorphic encryption. In *Public-Key Cryptography*, pages 70–98, 2016.

[27] B. Libert, T. Peters, M. Joye, and M. Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive nizk proofs and CCA2-secure encryption from homomorphic signatures. In *EUROCRYPT*, 2014.

[28] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[29] H. S. Rhee, W. Susilo, and H. Kim. Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electronic Express*, 6(5):237–243, 2009.

[30] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, and K. Omote. Methods for restricting message space in public-key encryption. *IEICE Transactions*, 96-A(6):1156–1168, 2013.

[31] K. Shimizu, K. Nuida, H. Arai, S. Mitsunari, N. Attrapadung, M. Hamada, K. Tsuda, T. Hirokawa, J. Sakuma, G. Hanaoka, and K. Asai. Privacy-preserving search for chemical compound databases. *Bioinformatics*, 16(18), 2015.

[32] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Secure pattern matching using somewhat homomorphic encryption. In *CCSW*, pages 65–76, 2013.

[33] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption. In *ACISP*, pages 338–353, 2014.

[34] R. Zhang and H. Imai. Combining public key encryption with keyword search and public key encryption. *IEICE Transactions*, 92-D(5):888–896, 2009.

Appendix

A.1 Omitted Definitions

Here, we introduce definitions of smooth function, the truncated decisional augmented bilinear Diffie-Hellman exponent (truncated decisional ABDHE) assumption, and KH-IBE. Moreover, we give the definition of anonymity of KH-IBE.

Definition 7.1 (SMOOTH FUNCTION [14]). *Let $f : \mathcal{X} \to \mathcal{Y}$ be a hash function. We say that f is* ϵ-smooth*, if the quantity $\mathsf{Smth}_f := \max_{y \in \mathcal{Y}} \Pr_{x \xleftarrow{\$} \mathcal{X}}[f(x) = y]$ is not larger than ϵ. We say that f is* smooth*, if it is ϵ-smooth for a negligible ϵ.*

Smoothness is introduced for compressing the size of the ciphertext, and a one-way function (OWF) has the property. In our implementation, we use SHA512 as the smooth function.

The truncated decisional ABDHE assumption is defined as follows.

Definition 7.2 (TRUNCATED DECISION q-ABDHE [19]). *Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups with prime order p, where $\langle g \rangle = \mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Let $g' \xleftarrow{\$} \mathbb{G}$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and $Z \xleftarrow{\$} \mathbb{G}_T$, and set $g'_i := g'^{(\alpha^i)}$ and $g_i := g^{(\alpha^i)}$. We say that truncated decision q-ABDHE assumption holds, if for any PPT adversary $\mathcal{A}$, its advantage $Adv^{\mathsf{ABDHE}}_{\mathcal{A}}(\kappa)$ defined by $Adv^{\mathsf{ABDHE}}_{\mathcal{A}}(\kappa) := |\Pr[\mathcal{A}(g', g'_{q+2}, g, g_1, \ldots, g_q, e(g_{q+1}, g')) = 0] - \Pr[\mathcal{A}(g', g'_{q+2}, g, g_1, \ldots, g_q, Z) = 0]|$ is negligible*

Definition 7.3 (SYNTAX OF KH-IBE [14]). *Let $\mathcal{M}$ be a message space, $\mathcal{ID}$ be an identity space, and $\odot$ be a binary operation over $\mathcal{M}$. A KH-IBE scheme $\mathcal{KH}$-$\mathcal{IBE}$, which consists of five algorithms* ($\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec}, \mathsf{IBE.Eval}$)*, is defined as follows:*

$\mathsf{IBE.Setup}$*: A setup algorithm takes a security parameter 1^κ ($\kappa \in \mathbb{N}$) as input, and returns a public parameter params and a master secret key* msk*.*

$\mathsf{IBE.KeyGen}$*: A key generation algorithm takas params,* msk*, and an identity $\mathsf{ID} \in \mathcal{ID}$ as input, and returns a decryption key $sk_{d,\mathsf{ID}}$ and a homomorphic operation key $sk_{h,\mathsf{ID}}$.*

$\mathsf{IBE.Enc}$*: An encryption algorithm takes params,* ID*, and a message $M \in \mathcal{M}$ as input, and returns a ciphertext C.*

$\mathsf{IBE.Dec}$*: An decryption algorithm takes params, $sk_{d,\mathsf{ID}}$ and C as input, and returns M or $\perp$.*

$\mathsf{IBE.Eval}$*: An evaluation algorithm takes params, $sk_{h,\mathsf{ID}}$ and two ciphertexts C_1 and C_2 as input, and returns a ciphertext C or $\perp$.*

Let $\mathsf{ID} \in \mathcal{ID}$ be an identity, *params* be a public parameter generated by the $\mathsf{IBE.Setup}$, and $\mathcal{C}_{\mathsf{ID},M}$ be the set of all ciphertexts of $M \in \mathcal{M}$ under the public key ID, i.e., $\mathcal{C}_{\mathsf{ID},M} = \{C | \exists r \in \{0,1\}^* \text{ s.t. } C = \mathsf{IBE.Enc}(params, \mathsf{ID}, M; r)\}$.

Definition 7.4 (CORRECTNESS [14]). *We say that a KH-IBE scheme for homomorphic operation $\odot$ is* correct *if for all $(params, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^k)$, (1) for all $\mathsf{ID} \in \mathcal{ID}$ and $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$, all $M \in \mathcal{M}$, and all $C \in \mathcal{C}_{\mathsf{ID},M}$, it holds that $\mathsf{IBE.Dec}(params, sk_{d,\mathsf{ID}}, C) = M$. (2) For all $\mathsf{ID} \in \mathcal{ID}$ and all $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$, all $M_1, M_2 \in \mathcal{M}$, all $C_1 \in \mathcal{C}_{\mathsf{ID},M_1}$ and $C_2 \in \mathcal{C}_{\mathsf{ID},M_2}$, it holds that $\mathsf{IBE.Eval}(params, sk_{h,\mathsf{ID}}, C_1, C_2) \in \mathcal{C}_{\mathsf{ID},M_1 \odot M_2}$.*

Next, we introduce the security notion for KH-IBE, which we call *indistinguishability of message under adaptive chosen ciphertext and identity attacks* (KH-ID-CCA).

Definition 7.5 (KH-ID-CCA [14]). *We say that a KH-IBE scheme is KH-ID-CCA secure if for any PPT adversary $\mathcal{A}$, the advantage*

$$\begin{aligned}\mathrm{Adv}^{KH\text{-}ID\text{-}CCA}_{\mathsf{KH\text{-}IBE},\mathcal{A}}(\kappa) = \big|\Pr[&(params, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^k);\\ &(\mathsf{ID}^*, M_0^*, M_1^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, params);\\ &b \xleftarrow{\$} \{0,1\};\ C^* \leftarrow \mathsf{IBE.Enc}(params, \mathsf{ID}^*, M_b^*);\\ &b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, C^*, st) :\ b = b'] - \frac{1}{2}\big|\end{aligned}$$

is negligible in κ. $\mathcal{O}$ consists of $\mathcal{O}^{KH\text{-}IBE}_{revhk}(params, \mathsf{msk}, \cdot)$, $\mathcal{O}^{KH\text{-}IBE}_{dec}(\cdot,\cdot)$, $\mathcal{O}^{KH\text{-}IBE}_{revdk}(params, \mathsf{msk}, \cdot)$, and $\mathcal{O}^{KH\text{-}IBE}_{eval}(params, \cdot, \cdot, \cdot)$ which are defined as follows. Let $\mathcal{D}$ be a list which is set as $\mathcal{D} = \{C^*\}$ right after the challenge stage ($\mathcal{D}$ is set as $\emptyset$ in the find *stage).*

The homomorphic operation key reveal oracle $\mathcal{O}^{KH\text{-}IBE}_{revhk}$ responds to a query $\mathsf{ID} \in \mathcal{ID}$ with $sk_{h,\mathsf{ID}}$ where $sk_{h,\mathsf{ID}}$ is a part of $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$.

The decryption oracle $\mathcal{O}^{KH\text{-}IBE}_{dec}$: For a query (ID, C) and $\mathsf{ID} = \mathsf{ID}^*$, this oracle is not available if $\mathcal{A}$ has sent ID^* to $\mathcal{O}^{KH\text{-}IBE}_{revhk}$ (i.e, $\mathcal{A}$ has obtained sk_{h,ID^*}) and $\mathcal{A}$ has obtained the challenge ciphertext C^*. Otherwise, this oracle responds to a query C with the result of $\mathsf{IBE.Dec}(sk_{d,\mathsf{ID}}, C)$ if $C \notin \mathcal{D}$ or $\mathsf{ID} \neq \mathsf{ID}^*$, or returns $\perp$ if $C \in \mathcal{D}$ and $\mathsf{ID} = \mathsf{ID}^*$.

The key generation oracle $\mathcal{O}^{KH\text{-}IBE}_{revdk}$ responds to a query $\mathsf{ID} \in \mathcal{ID}$ with $sk_{d,\mathsf{ID}}$ where $sk_{d,\mathsf{ID}}$ is the result of $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$. $\mathcal{A}$ is not allowed to query ID^* to the oracle.

The evaluation oracle $\mathcal{O}^{KH\text{-}IBE}_{eval}$ responds to a query (ID, C_1, C_2) with the result of $C \leftarrow \mathsf{IBE.Eval}(sk_{h,\mathsf{ID}}, C_1, C_2)$. In addition, in the case $\mathsf{ID} = \mathsf{ID}^*$, if $C \neq \perp$ and either $C_1 \in \mathcal{D}$ or $C_2 \in \mathcal{D}$, then the oracle updates the list by $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}$.

Next, we define the anonymity of KH-IBE (KH-ANON-CCA). The security proof of KH-IBE and that of the Gentry IBE scheme are essentially the same, except the evaluation oracle. In the security model of KH-IBE (KH-IBE-CCA), an adversary is allowed to issue evaluation queries even for the challenge ciphertext, and such challenge-related ciphertexts are not allowed to be inputs of the decryption oracle. Though these evaluation queries do not contradict the KH-IBE-CCA security of KH-IBE, they contradict anonymity. That is, an adversary can always win the game if the adversary is allowed to issue a ciphertext of the challenge identity (either ID_0^* or ID_1^*) since the evaluation oracle cannot help returning $\perp$ if the ciphertext is not computed by ID_b^* where $b \in \{0,1\}$ is the challenge bit. So, we need to restrict such a query as follows: the adversary sends a plaintext M and the challenge-related ciphertext when the adversary would like to obtain the evaluation result of the challenge-related ciphertext, and the oracle encrypts M by using ID_b^*, evaluates these ciphertexts, and returns the result to the adversary. Moreover, due to the same reason, the adversary is not allowed to obtain homomorphic operation keys of ID_0^* and ID_1^*. Under these restrictions, the Gentry-based KH-IBE scheme is anonymous.

Definition 7.6 (KH-ANON-CCA). *We say that a KH-IBE scheme is KH-ANON-CCA secure if for any PPT adversary $\mathcal{A}$, the advantage*

$$\mathrm{Adv}^{KH\text{-}ANON\text{-}CCA}_{\mathsf{KH\text{-}IBE},\mathcal{A}}(\kappa) = \Big|\Pr[(params, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\kappa); (M^*, \mathsf{ID}_0^*, \mathsf{ID}_1^*, st) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, params); b \xleftarrow{\$} \{0,1\};\ C^* \leftarrow \mathsf{IBE.Enc}(params, \mathsf{ID}_b^*, M^*); b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, C^*, st) :\ b = b'] - \frac{1}{2}\Big|$$

is negligible in κ, where $\mathcal{O}$ consists of $\mathcal{O}^{KH\text{-}IBE}_{revhk}(params, \mathsf{msk}, \cdot)$, $\mathcal{O}^{KH\text{-}IBE}_{dec}(\cdot,\cdot)$, $\mathcal{O}^{KH\text{-}IBE}_{revdk}(params, \mathsf{msk}, \cdot)$, $\mathcal{O}^{KH\text{-}IBE}_{eval}(params, \cdot,\cdot,\cdot)$, and $\mathcal{O}'^{KH\text{-}IBE}_{eval}(params, \cdot,\cdot,\cdot)$ which are defined as follows. Let $\mathcal{D}$ be a list which is set as $\mathcal{D} = \{C^*\}$ right after the challenge stage ($\mathcal{D}$ is set as $\emptyset$ in the find *stage).*

The homomorphic operation key reveal oracle $\mathcal{O}^{KH\text{-}IBE}_{revhk}$ responds to a query $\mathsf{ID} \in \mathcal{ID} \setminus \{\mathsf{ID}_0^*, \mathsf{ID}_1^*\}$ with $sk_{h,\mathsf{ID}}$ where $sk_{h,\mathsf{ID}}$ is a part of $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$.

The decryption oracle $\mathcal{O}^{KH\text{-}IBE}_{dec}$: For a query (ID, C) this oracle responds to a query C with the result of $\mathsf{IBE.Dec}(sk_{d,\mathsf{ID}}, C)$. We remark that $C \in \mathcal{D}$ is allowed to be an input of this oracle. In this case, the input identity is not required.

The key generation oracle $\mathcal{O}^{KH\text{-}IBE}_{revdk}$ responds to a query $\mathsf{ID} \in \mathcal{ID}\backslash\{\mathsf{ID}_0^*, \mathsf{ID}_1^*\}$ with $sk_{d,\mathsf{ID}}$ where $sk_{d,\mathsf{ID}}$ is a part of $(sk_{d,\mathsf{ID}}, sk_{h,\mathsf{ID}}) \leftarrow \mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID})$.

The evaluation oracle $\mathcal{O}^{KH\text{-}IBE}_{eval}$ responds to a query (ID, C_1, C_2) where $C_1, C_2 \notin \mathcal{D}$, with the result of $C \leftarrow \mathsf{IBE.Eval}(sk_{h,\mathsf{ID}}, C_1, C_2)$.

The evaluation oracle for the challenge-related ciphertext $\mathcal{O}'^{KH\text{-}IBE}_{eval}$ responds to a query (M, C), where $M \in \mathcal{M}$ and $C \in \mathcal{D}$, with the result of $C'' \leftarrow \mathsf{IBE.Eval}(sk_{h,\mathsf{ID}_b^*}, C, C')$ where $C' = \mathsf{IBE.Enc}(params, \mathsf{ID}_b^*, M)$ and $sk_{h,\mathsf{ID}_b^*} \leftarrow$ is generated by executing $\mathsf{IBE.KeyGen}(params, \mathsf{msk}, \mathsf{ID}_b^*)$. This oracle updates the list by $\mathcal{D} \leftarrow \mathcal{D} \cup \{C''\}$. We remark ID_b^* has been determined in the experiment if $\mathcal{D} \neq \perp$.

A.2 Proofs of Theorems

First, we give the security proof of Theorem 3.1 (Consistency). For the proof, we assume that the discrete logarithm problem over $\mathbb{G}$ is hard. That is, given $(g, h, \mathbb{G}, \mathbb{G}_T, e, p)$ where $g, h \xleftarrow{\$} \mathbb{G}$, it is computationally infeasible to compute $\log_g h \in \mathbb{Z}_p$.

Proof: If either $\omega = \alpha$ or $\omega' = \alpha$, then we can immediately construct an algorithm that breaks the discrete logarithm problem $(g, g_1 := g^\alpha)$. From now on, we assume that $\omega, \omega' \neq \alpha$. Let $C = (c_1, c_2, c_3, c_4, \tau)$ be a ciphertext generated in the experiment, and for $\delta \leftarrow \Gamma_{hk}(c_1, c_2, c_3, c_4)$, $c_5 \leftarrow e(g,h_3)^s e(g,h_4)^{s\delta}$ be a part of ciphertext such that $\tau = f(c_5)$ holds, where s is a random number which is used for computing (c_1, c_2, c_3, c_4). Let $c_5' := e(c_1, h_{\omega',3} h_{\omega',4}^{\delta}) c_2^{r_{\omega',3} + r_{\omega',4}\delta}$. In the experiment, $\mathsf{MR\text{-}SHE.Test}(pk, t_{\omega'}, C) = 1$ means either (1) $c_5 = c_5'$ or (2) $c_5 \neq c_5'$ and $\tau = f(c_5')$.

Case (1):

$$\begin{aligned} & e(c_1, h_{\omega',3} h_{\omega',4}^{\delta}) c_2^{r_{\omega',3} + r_{\omega',4}\delta} \\ =& e(g,h_3)^{\frac{\alpha-\omega}{\alpha-\omega'}s} e(g,h_4)^{\frac{\alpha-\omega}{\alpha-\omega'}s\delta} e(g,g)^{s(r_{\omega',3} + \delta r_{\omega',4})(1-\frac{\alpha-\omega}{\alpha-\omega'})} \end{aligned}$$

holds. Here,

$$\begin{aligned} c_5 =& e(g,h_3)^s e(g,h_4)^{s\delta} \\ =& e(g,h_3)^{\frac{\alpha-\omega}{\alpha-\omega'}s} e(g,h_4)^{\frac{\alpha-\omega}{\alpha-\omega'}s\delta} e(g,g)^{s(r_{\omega',3} + \delta r_{\omega',4})(1-\frac{\alpha-\omega}{\alpha-\omega'})} \end{aligned}$$

holds. Set $A := \frac{\alpha-\omega}{\alpha-\omega'}$. Then,

$$\begin{aligned} & s\log_g h_3 + s\delta \log_g h_4 \\ =& As\log_g h_3 + As\delta \log_g h_4 + s(r_{\omega',3} + \delta r_{\omega',4})(1-A) \end{aligned}$$

holds. From this equation, we obtain

$$s(1-A)(\log_g h_3 + \delta \log_g h_4 - r_{\omega',3} - \delta r_{\omega',4}) = 0$$

Since $\omega \neq \omega'$, $1 - A = 1 - \frac{\alpha-\omega}{\alpha-\omega'} \neq 0$. Here, we can assume that $s \neq 0$ since $s \xleftarrow{\$} \mathbb{Z}_p$. Assume $\log_g h_3 + \delta \log_g h_4 - r_{\omega',3} - \delta r_{\omega',4} = 0$ holds. Then, since $\log_g h_3 = -\delta \log_g h_4 + r_{\omega',3} + \delta r_{\omega',4}$ we can construct an algorithm that breaks the discrete logarithm problem as follows. Let (g, h_3) be an instance of the discrete logarithm problem. Then, the algorithm setups the scheme with one exception that choose $u \xleftarrow{\$} \mathbb{Z}_p$ and set $h_4 = g^u$. Then, the algorithm can compute $\log_g h_3 = -\delta u + r_{\omega',3} + \delta r_{\omega',4}$. That is, the probability that the case (1) happens is negligible.

Case (2): The probability that the case (2) happens is negligible since f is a smooth function. □

As a remark, $t_{\omega'}$ is honestly generated in the definition of consistency. Thus, the probability that $\mathsf{MR\text{-}SHE.Test}(pk, t_{\omega'}, C) = 1$ for a ciphertext C associated with ω where $\omega \neq \omega'$ is negligible. In other word, there exist a trapdoor that breaks consistency, where $\log_g h_3 + \delta \log_g h_4 - r_{\omega',3} - \delta r_{\omega',4} = 0$ holds. Since we consider computational security, this does not contradict consistency.

Next, we give the security proofs of Theorem 3.2 (Data privacy) and 3.3 (Keyword privacy). Intuitively, Data privacy directly holds since the underlying KH-IBE scheme is KH-ID-CCA secure, and the definition of data privacy and that of KH-ID-CCA is essentially the same. Similarly, keyword privacy also directly holds if the underlying KH-IBE scheme is anonymous. Since the KH-IBE scheme is KH-ID-CCA and KH-ANON-CCA secure under the truncated decision q-ABDHE (augmented bilinear Diffie-Hellman exponent) assumption [19], our scheme also relies on the same assumption. We prove two lemmas where, for data privacy and keyword privacy, MR-SHE can be constructed from anonymous KH-IBE in a (almost) generic way. That is, we need to confirm that $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$ can be simulated by $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$. In our scheme, for a $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$ query ω, $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$ returns $sk_{h,\omega} = ((r_{\omega,3}, h_{\omega,3}), (r_{\omega,4}, h_{\omega,4}))$. After obtaining $sk_{h,\omega}$, the simulator can compute t_ω such that $t_\omega = (g^\omega, sk_{h,\omega})$. Thus, in the following proofs, we assume that $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$ can be simulated by $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$.

Lemma 7.1. *If the underlying KH-IBE scheme is KH-ID-CCA secure, then the MR-SHE scheme is data private.*

Proof. Let $\mathcal{A}$ be an adversary of the data privacy of MR-SHE. We construct an algorithm $\mathcal{B}$ that breaks KH-ID-CCA security of KH-IBE as follows. Let C be the challenger of the KH-ID-CCA game of KH-IBE.

First, C runs $(params, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^k)$, and gives $params$ to $\mathcal{B}$, and $\mathcal{B}$ forwards $params$ to $\mathcal{A}$ as pk.

When $\mathcal{A}$ issues $\mathcal{O}^{\text{MR-SHE}}_{\text{revhk}}$, $\mathcal{O}^{\text{MR-SHE}}_{\text{dec}}$, $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdor}}$, and $\mathcal{O}^{\text{MR-SHE}}_{\text{eval}}$ queries, $\mathcal{B}$ simulates the oracles by using $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$, $\mathcal{O}^{\text{KH-IBE}}_{\text{dec}}$, $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$, and $\mathcal{O}^{\text{KH-IBE}}_{\text{eval}}$ oracles of KH-IBE, respectively.

For a $\mathcal{O}^{\text{MR-SHE}}_{\text{test}}$ query (ω, C), if ω has been input to $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$, then $\mathcal{B}$ directly returns the result of $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$. Otherwise, if ω is not input $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$ and if the challenge keyword ω^* is not given by $\mathcal{A}$, then $\mathcal{B}$ should not obtain the corresponding t_ω since ω might be ω^*. Thus, $\mathcal{B}$ computes a ciphertext $C' \leftarrow \mathsf{IBE.Enc}(params, \omega, M)$ for a random M, and sends (ω, C, C') to $\mathcal{C}$ as a $\mathcal{O}^{\text{KH-IBE}}_{\text{dec}}$ query. If the response is $\perp$, then C is not associated with ω. Thus, $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns 1 to $\mathcal{A}$. If $\mathcal{B}$ can recognize $\omega \neq \omega^*$, i.e., after the challenge kerword is given by $\mathcal{A}$, then $\mathcal{B}$ can simply send ω as a $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$ query, obtains $sk_{h,\omega}$, computes t_ω from $sk_{h,\omega}$, and returns the result of $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$.

$\mathcal{A}$ sends (ω^*, M_0^*, M_1^*) to $\mathcal{B}$ as the challenge of MR-SHE. $\mathcal{B}$ forwards (ω^*, M_0^*, M_1^*) to $\mathcal{C}$ as the challenge of KH-IBE, obtains the challenge ciphertext C^*, and returns C^* to $\mathcal{A}$.

$\mathcal{B}$ responds queries issued by $\mathcal{A}$ as in the previous phase. Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b', and can break the KH-ID-CCA security with the same advantage of breaking the data privacy. □

Lemma 7.2. *If the underlying KH-IBE scheme is KH-ANON-CCA secure, then the MR-SHE scheme is keyword private.*

Proof. Let $\mathcal{A}$ be an adversary of the keyword privacy of MR-SHE. We construct an algorithm $\mathcal{B}$ that breaks KH-ANON-CCA security of KH-IBE as follows. Let C be the challenger of the KH-ANON-CCA game of KH-IBE.

First, C runs $(params, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^k)$, and gives $params$ to $\mathcal{B}$, and $\mathcal{B}$ forwards $params$ to $\mathcal{A}$ as pk.

When $\mathcal{A}$ issues $\mathcal{O}^{\text{MR-SHE}}_{\text{revhk}}$, $\mathcal{O}^{\text{MR-SHE}}_{\text{dec}}$, $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdor}}$, $\mathcal{O}^{\text{MR-SHE}}_{\text{eval}}$, and $\mathcal{O}'^{\text{MR-SHE}}_{\text{eval}}$ queries, $\mathcal{B}$ simulates the oracles by using $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$, $\mathcal{O}^{\text{KH-IBE}}_{\text{dec}}$, $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$, $\mathcal{O}^{\text{KH-IBE}}_{\text{eval}}$, and $\mathcal{O}'^{\text{KH-IBE}}_{\text{eval}}$ oracles of KH-IBE, respectively.

For a $\mathcal{O}^{\text{MR-SHE}}_{\text{test}}$ query (ω, C), if ω has been input to $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$, then $\mathcal{B}$ directly returns the result of $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$. Otherwise, if ω is not input $\mathcal{O}^{\text{MR-SHE}}_{\text{trapdoor}}$ and if the challenge keyword ω^* is not given by $\mathcal{A}$, then $\mathcal{B}$ should not obtain the corresponding t_ω since ω might be ω^*. Thus, $\mathcal{B}$ computes a ciphertext $C' \leftarrow \mathsf{IBE.Enc}(params, \omega, M)$ for a random M, and sends (ω, C, C') to $\mathcal{C}$ as a $\mathcal{O}^{\text{KH-IBE}}_{\text{dec}}$ query. If the response is $\perp$, then C is not associated with ω. Thus, $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns 1 to $\mathcal{A}$. If $\mathcal{B}$ can recognize $\omega \neq \omega^*$, i.e., after the challenge kerword is given by $\mathcal{A}$, then $\mathcal{B}$ can simply send ω as a $\mathcal{O}^{\text{KH-IBE}}_{\text{revhk}}$ query, obtains $sk_{h,\omega}$, computes t_ω from $sk_{h,\omega}$, and returns the result of $\mathsf{MR\text{-}SHE.Test}(pk, t_\omega, C)$.

$\mathcal{A}$ sends $(\omega_0^*, \omega_1^*, M^*)$ to $\mathcal{B}$ as the challenge of MR-SHE. $\mathcal{B}$ forwards $(\omega_0^*, \omega_1^*, M^*)$ to $\mathcal{C}$ as the challenge of KH-IBE, obtains the challenge ciphertext C^*, and returns C^* to $\mathcal{A}$.

$\mathcal{B}$ responds queries issued by $\mathcal{A}$ as in the previous phase. Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}$ outputs b', and can break the KH-ANON-CCA security with the same advantage of breaking the keyword privacy. □