Chosen Ciphertext Secure Keyed Two-Level Homomorphic Encryption[*]

Yusaku Maeda[1] Koji Nuida[23]

[1] The University of Tokyo, Japan[†]
yusaku.maeda1996@gmail.com
[2] Institute of Mathematics for Industry (IMI), Kyushu University, Japan
nuida@imi.kyushu-u.ac.jp
[3] National Institute of Advanced Industrial Science and Technology (AIST), Japan

Abstract

Homomorphic encryption (HE) is a useful variant of public key encryption (PKE), but it has a drawback that HE cannot fully achieve IND-CCA2 security, which is a standard security notion for PKE. Beyond existing HE schemes achieving weaker IND-CCA1 security, Emura et al. (PKC 2013) proposed the notion of "keyed" version of HE, called KH-PKE, which introduces an evaluation key controlling the functionality of homomorphic operations and achieves security stronger than IND-CCA1 and as close to IND-CCA2 as possible. After Emura et al.'s scheme which can evaluate linear polynomials only, Lai et al. (PKC 2016) proposed a fully homomorphic KH-PKE, but it requires indistinguishability obfuscation and hence has a drawback in practical feasibility. In this paper, we propose a "two-level" KH-PKE scheme for evaluating degree-two polynomials, by cleverly combining Emura et al.'s generic framework with a recent efficient two-level HE by Attrapadung et al. (ASIACCS 2018). Our scheme is the first KH-PKE that can handle non-linear polynomials while keeping practical efficiency.

1 Introduction

1.1 Background

Homomorphic Encryption (HE) is a cryptographic primitive first introduced by Rivest et al. [19], which allows one to compute on encrypted data without a secret key. It has been extensively studied since its introduction, and many applications have also been proposed.

There are several different types of HE depending on the operations it can handle. The most basic ones are Additively Homomorphic Encryption (AHE), which only allows addition between ciphertexts, and Multiplicative Homomorphic Encryption (MHE), which only allows multiplication. Although the applications of these schemes are limited due to their simple functionality, they generally are known to have a comparatively low computation cost. On the other hand, Fully Homomorphic Encryption (FHE) can carry out arbitrary computations on encrypted data. Especially when the message space is a finite field, FHE can be constructed by enabling (unlimited number of) both addition and multiplication between ciphertexts. Despite its strong functionality, efficiency of FHE is still not sufficiently practical, and improvement on its efficiency is one of the most intensely studied topics in cryptography. To take the advantages of both types of HE, another type of HE called Somewhat Homomorphic Encryption (SHE) is also studied. SHE allows one to compute unlimited number of addition and a limited number of multiplication. Among SHE schemes,

[*]This work was done when the first author was a graduate student at Graduate School of Information Science and Technology, The University of Tokyo and the second author was also with the same department.

[†]Currently, the first author has left the university and is with a private company.

Two-Level Homomorphic Encryption (2LHE) schemes [3, 5, 14], which enables a single multiplication (as well as unlimited number of addition) over ciphertexts, have significantly better efficiency than the other SHE schemes. In those schemes, the ciphertexts are classified into "level-1" and "level-2" ciphertexts, and ciphertext addition is allowed within the same level, while ciphertext multiplication with two level-1 ciphertexts results in a level-2 ciphertext. The state-of-the-art 2LHE scheme at the present is the scheme proposed by Attrapadung et al. in 2018 [3], which is based on prime-order pairing group and practical computational assumption (SXDH assumption) and has simpler structure than the previous schemes.

As mentioned above, HE is a highly useful primitive and is expected to be applied to various fields. However, it has a drawback that its security is weaker than non-homomorphic encryption schemes. Let us explain this in details. The most common security notions for public key encryption are IND-CPA, IND-CCA1, and IND-CCA2 security. Among them, IND-CCA2 security is the strongest, and IND-CPA security is the weakest. Today, IND-CCA2 security is thought to be the preferable security for public key encryption. However, it is well-known that a homomorphic encryption scheme cannot satisfy IND-CCA2 security in principle. Roughly speaking, this is due to the fact that enabling computation over ciphertexts is equivalent to allowing alteration of ciphertexts, the latter being contradictory to IND-CCA2 security.

To overcome this issue, Emura et al. pointed out that the property that anyone could perform homomorphic operation was the main obstacle for achieving IND-CCA2 security, and proposed Keyed-Homomorphic Public Key Encryption (KH-PKE) in 2013 [12, 13][1]. In KH-PKE, the functionality to perform homomorphic operations is administrated by an evaluation key ek. They also proposed a new security notion called KH-CCA security for KH-PKE schemes. KH-CCA security achieves IND-CCA1 security against adversaries possessing ek from the beginning, and IND-CCA2 security against adversaries not possessing ek at all. Moreover, as an intermediate situation, it also deals with the leakage of the ek during the security game. Based on the security notion, they proposed a concrete construction of KH-PKE schemes using a cryptographic primitive called Hash Proof System (HPS). Here, their KH-PKE scheme is practically efficient, but on the other hand, it realizes the functionality of AHE only. There are no existing studies in the literature to realize stronger types of HE functionality while keeping the practical efficiency (see Section 1.3 below for details).

1.2 Our Contributions

In this paper, we introduce new type of KH-PKE named *Keyed Two-Level Homomorphic Encryption* (*Keyed-2LHE*), which can handle unlimited number of addition and a single multiplication over ciphertexts administrated by an evaluation key. We also give a concrete construction of Keyed-2LHE schemes. Although our Keyed-2LHE scheme has some overhead compared to the KH-PKE scheme by Emura et al. [12, 13] and to the 2LHE scheme by Attrapadung et al. [3], the underlying setting (prime-order pairing groups and SXDH assumption) is the same as [3] and the overhead is within a feasible range (e.g., our public key and ciphertext sizes are only up to four times larger than those of [3]). This is the first KH-PKE that allows both addition and multiplication with practical efficiency.

The very first idea for constructing our Keyed-2LHE scheme is simple; it is a kind of abstraction of Attrapadung et al.'s 2LHE scheme that can then be interpreted in the context of Emura et al.'s generic framework for realizing KH-PKE. We emphasize, however, that it was never a straightforward task to successfully combine Emura et al.'s and Attrapadung et al.'s schemes. In detail, the generic construction by Emura et al. uses three kinds of HPSs $\boldsymbol{P}$, $\widehat{\boldsymbol{P}}$, and $\widetilde{\boldsymbol{P}}$. Here $\boldsymbol{P}$ is for masking the plaintext, and $\widetilde{\boldsymbol{P}}$ is for achieving IND-CCA2 security when the evaluation key is not used. The role of $\widehat{\boldsymbol{P}}$ is most complicated; it should simultaneously take care of having (additive) homomorphic property and achieving IND-CCA1 security when the evaluation key is available. In our proposed scheme, there are two levels of ciphertexts, and hence $3 \times 2 = 6$ HPSs are used in total. Among them, the constructions of five HPSs except $\widehat{\boldsymbol{P}}$ for level-2 ciphertexts are relatively simple; these are, in some sense, direct products of HPSs where the underlying HPSs follow known constructions already used by Emura et al. [12]. On the other hand, for the $\widehat{\boldsymbol{P}}$ in level-2 part, it is not sufficient to take care of additive homomorphic property and IND-CCA1 security; it

[1]The notion of KH-PKE was proposed in 2013 [13]. A construction for concrete KH-PKE schemes was also given in that paper, but its security proof was not correct and their scheme was actually not secure. The issue was then fixed in 2018 [12] by modifying the construction as well as the security proof.

should also be related in an appropriate manner to the HPSs in level-1 part in order to realize multiplicative homomorphic property. For simultaneously achieving these three requirements, the known construction of HPSs used in [12] was not enough, and we had to develop a new tailor-made HPS to fill in the last piece of our construction. We also note that our resulting scheme fortunately shares a key property (called source ciphertext hiding property) with Emura et al.'s construction, which makes the security proof of our proposed scheme just analogous to the proof in [12].

It is also worth noting that in our scheme, it is possible to permit only a certain type of homomorphic operation by distributing ek partially. This is a new feature of our scheme that the previous schemes did not have. More precisely, in our scheme, only a part of ek is sufficient to perform level-1 homomorphic addition, while the remaining part of ek (as well as the former part) is used by the other operations, i.e., level-2 homomorphic addition and homomorphic multiplication. It is expected that the administration of homomorphic operations can be made hierarchical by such a way, though we do not study the corresponding security formulation nor security proof in this paper (for the sake of simplicity) and leave them as future research topics.

1.3 Related Work

1.3.1 Researches Related to KH-PKE

After the introduction of KH-PKE by Emura et al. [12, 13], several schemes based on different security assumptions were proposed. Libert et al. [17] proposed a KH-PKE scheme based on DLIN/SXDH assumptions, and Jutla et al. [15] proposed a scheme based on SXDH assumption. Each of their schemes has an additional property such as threshold decryption or public verifiability, but their efficiency is worse compared to [12, 13]. Also, in comparison to our scheme, their schemes only allow a single type of homomorphic operation. On the other hand, Lai et al. [16] proposed a Keyed-FHE scheme, that is, a KH-PKE scheme that can carry out any operation over ciphertexts. However, their construction has a drawback that it uses a cryptographic primitive called indistinguishability obfuscation ($i\mathcal{O}$). Since a practical construction of $i\mathcal{O}$ is still not established, their scheme is more of a theoretical result rather than a practical one. Also, unlike our scheme, their scheme does not allow partial permission of homomorphic operations.

There are also several researches on achieving security close to IND-CCA2 while allowing computation over ciphertexts [2, 7, 18]. However, they differ from KH-PKE in that they are not equipped with an evaluation key and they only achieve security strictly weaker than IND-CCA2 (in contrast to our scheme achieving IND-CCA2 security against adversaries not possessing the evaluation key).

1.3.2 Researches Related to SHE

Our Keyed-2LHE scheme is constructed based on a 2LHE scheme of Attrapadung et al. [3] which uses pairings. There are other pairing-based 2LHE schemes such as ones by Boneh et al. [5] and Freeman [14]. Also, Catalano and Fiore proposed a method for converting an AHE scheme to a 2LHE scheme [8]. However, all these schemes have lower efficiency compared to the scheme in [3]. In more details, the scheme in [5] uses expensive composite-order pairing groups instead of prime-order ones as in [3], and the scheme in [8] has a drawback that homomorphic addition over level-2 ciphertexts increases the ciphertext size. The scheme in [14] uses prime-order pairing groups and is fairly efficient, but the scheme in [3] is even more efficient than that in [14]. We also note that (lattice-based) SHE/FHE schemes (e.g., [6]) are superior to 2LHE schemes in terms of functionality, but their efficiency is significantly worse than the scheme in [3].

1.4 Organization of the Paper

In Section 2, we will review some basic notions on mathematics and cryptography required. In Section 3, we will describe Hash Proof System (HPS), a main tool for constructing our scheme. Then we will explain definitions and construction of KH-PKE proposed by Emura et al. [12, 13] in Section 4. We will give a concrete description of our proposed scheme in Section 5; its detailed security proof is not included in the main text and is described in Appendix as it is basically analogous to the original security proof in [12]. Finally, we will evaluate the performance of our scheme in Section 6.

2 Preliminaries

In this section, we review basic notions on mathematics and cryptography. Throughout the paper, we refer to a probabilistic polynomial-time algorithm as a PPT algorithm. For a probabilistic algorithm $\mathcal{A}$, we write $a \leftarrow \mathcal{A}$ to represent that a is obtained as an output of $\mathcal{A}$. Similarly, for a set A, we write $a \leftarrow A$ to represent that a is obtained by uniformly and randomly choosing an element of A.

Definition 1 (negligible function). *A function $f\colon \mathbb{N} \to \mathbb{R}$ is said to be* negligible*, when for any polynomial* $\mathsf{poly}(n)$*, there exists $N > 0$ such that for any $n > N$, we have $f(n) < 1/\mathsf{poly}(n)$.*

Definition 2 (statistical indistinguishability). *Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be families of random variables defined on a finite set Ω. $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are said to be ε-*close*, when the statistical distance*

$$\Delta(X_n, Y_n) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[X_n = \omega] - \Pr[Y_n = \omega]|$$

is less than or equal to ε. Furthermore, when $\varepsilon = \varepsilon(n)$ is negligible in n, we say $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are statistically indistinguishable*, denoted by $\{X_n\}_{n\in\mathbb{N}} \overset{s}{\approx} \{Y_n\}_{n\in\mathbb{N}}$.*

Definition 3 (computational indistinguishability). *Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be families of random variables. $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are said to be* computationally indistinguishable*, denoted by $\{X_n\}_{n\in\mathbb{N}} \overset{c}{\approx} \{Y_n\}_{n\in\mathbb{N}}$, when for any PPT algorithm $\mathcal{A}$, we have*

$$|\Pr[\mathcal{A}(X_n) = 1] - \Pr[\mathcal{A}(Y_n) = 1]| = \mathrm{negl}(n)$$

where $\mathrm{negl}(n)$ *denotes a negligible function in n.*

Definition 4 (approximate samplability). *For a finite set B_n and its subset B'_n indexed by $n \in \mathbb{N}$, we say B'_n is* approximately samplable relative to *B_n when there exists a sequence of random variables on B_n that is statistically indistinguishable from the uniform distribution on B'_n and polynomial-time samplable.*

We also recall basic properties of hash functions.

Definition 5 (collision resistant hash function). *Let $\{f_i\}_{i\in I}$ be a family of hash functions indexed by $i \in I$, which is specified by the security parameter 1^ℓ. $\{f_i\}_{i\in I}$ is said to be* collision resistant (CR)*, when for any PPT algorithm $\mathcal{A}$, the probability*

$$\Pr[x \neq x^* \wedge f_i(x) = f_i(x^*) \mid i \leftarrow I; (x, x^*) \leftarrow \mathcal{A}(1^\ell, i)]$$

is negligible in ℓ.

The following well-known lemma is used in the security proof of our proposed scheme.

Lemma 1 (difference lemma). *Assuming that three events A, B, C satisfy $\Pr[A \wedge \neg C] = \Pr[B \wedge \neg C]$, we have $|\Pr[A] - \Pr[B]| \leq \Pr[C]$.*

2.1 Pairings

Definition 6 (bilinear group generation algorithm). *A* bilinear group generation algorithm GenBG *takes a security parameter 1^ℓ as an input, and outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$. Here, $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of prime order $p = \Theta(2^\ell)$, while g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively, and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map called pairing.*

Pairings are classified into three types depending on the relationship between $\mathbb{G}_1$ an $\mathbb{G}_2$.

- Type-1: the case when $\mathbb{G}_1 = \mathbb{G}_2$.
- Type-2: the case when $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable homomorphism $\mathbb{G}_1 \to \mathbb{G}_2$.
- Type-3: the case when $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists no efficiently computable homomorphism $\mathbb{G}_1 \to \mathbb{G}_2$.

In this paper, we only deal with Type-3 pairings.

Definition 7 (SXDH assumption)**.** *We say that* Symmetric External Diffie–Hellman (SXDH) assumption *holds in* GenBG*, when for* $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathsf{GenBG}(1^\ell)$ *and* $i = 1, 2$*, two distributions*

$$\{(\mathsf{pp}, g_i^\alpha, g_i^\beta, g_i^{\alpha\beta}) \mid \alpha, \beta \leftarrow \mathbb{Z}_p\}, \quad \{(\mathsf{pp}, g_i^\alpha, g_i^\beta, g_i^\gamma) \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p\}$$

are computationally indistinguishable.

Roughly speaking, SXDH assumption states that DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.2 Notation

From now on, all operations on cyclic groups will be written additively, unless otherwise noted. Also for simplicity, we will use a symbol for tensor product to represent pairings: i.e., for $x \in \mathbb{G}_1$ and $y \in \mathbb{G}_2$, we write $x \otimes y$ to represent $e(x, y)$. We will extend this notation to matrices in the following manner. We define the tensor product between $x \in \mathbb{G}_1$ and a matrix $Y = (y_{ij}) \in \mathbb{G}_2^{k \times \ell}$ as

$$x \otimes Y := \begin{bmatrix} x \otimes y_{11} & \dots & x \otimes y_{1\ell} \\ \vdots & \ddots & \vdots \\ x \otimes y_{k1} & \dots & x \otimes y_{k\ell} \end{bmatrix} \in \mathbb{G}_T^{k \times \ell}$$

and for a matrix $X = (x_{ij}) \in \mathbb{G}_1^{m \times n}$, we define the tensor product between X and Y as

$$X \otimes Y := \begin{bmatrix} x_{11} \otimes Y & \dots & x_{1n} \otimes Y \\ \vdots & \ddots & \vdots \\ x_{m1} \otimes Y & \dots & x_{mn} \otimes Y \end{bmatrix} \in \mathbb{G}_T^{mk \times n\ell}.$$

Note that this definition satisfies $(X \otimes Y)^\top = X^\top \otimes Y^\top$, where $\top$ represents the trasposition of a matrix.

Furthermore, we define the tensor product between two elements $a, b \in \mathbb{Z}_p$ as $a \otimes b := ab$, and extend it to matrices over $\mathbb{Z}_p$ in a similar manner as above. Also, multiplication of a matrix over $\mathbb{Z}_p$ to a matrix with components in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$ is defined in the same way as the usual matrix multiplication where the scalar multiplication to group elements plays the role of multiplication between matrix components. Note that these definitions satisfy $(AX) \otimes (BY) = (A \otimes B)(X \otimes Y)$ (assuming that the matrix sizes are consistent to multiplication), where A and B are matrices over $\mathbb{Z}_p$, X is a matrix over $\mathbb{G}_1$, and Y is a matrix over $\mathbb{G}_2$.

When we apply this notation, SXDH assumption can be rewritten in the following way.

Proposition 1. *SXDH assumption is equivalent to the following statement: for* $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathsf{GenBG}(1^\ell)$*,* $g_1' \leftarrow \mathbb{G}_1$*,* $g_2' \leftarrow \mathbb{G}_2$*,* $\boldsymbol{g}_1 = (g_1, g_1') \in \mathbb{G}_1^2$*, and* $\boldsymbol{g}_2 = (g_2, g_2') \in \mathbb{G}_2^2$*, two distributions*

$$\{(\mathsf{pp}, \boldsymbol{g}_i, \boldsymbol{x}) \mid \boldsymbol{x} \leftarrow \langle \boldsymbol{g}_i \rangle\}, \quad \{(\mathsf{pp}, \boldsymbol{g}_i, \boldsymbol{x}) \mid \boldsymbol{x} \leftarrow \mathbb{G}_i^2\}$$

are computationally indistiguishable for both $i = 1, 2$*.*

Proof. Two distributions in the statement can be rewritten as

$$\{(\mathsf{pp}, \boldsymbol{g}_i, \boldsymbol{x}) \mid \boldsymbol{x} \leftarrow \langle \boldsymbol{g}_i \rangle\} = \{(\mathsf{pp}, (g_i, \alpha g_i), (\beta g_i, \alpha\beta g_i)) \mid \alpha, \beta \leftarrow \mathbb{Z}_p\} \ ,$$

$$\{(\mathsf{pp}, \boldsymbol{g}_i, \boldsymbol{x}) \mid \boldsymbol{x} \leftarrow \mathbb{G}_i^2\} = \{(\mathsf{pp}, (g_i, \alpha g_i), (\beta g_i, \gamma g_i)) \mid \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p\} \ ,$$

which are obviously equivalent to the distributions in Definition 7. Hence the proposition holds. $\square$

3 Hash Proof System (HPS)

Hash Proof System (HPS), or Smooth Projective Hash Function (SPHF), is a cryptographic primitive, originally introduced by Cramer and Shoup [10, 11] for constructing IND-CCA2 secure cryptosystem. Since then, it has been extensively studied and wide applications are known such as password-authenticated key exchange (PAKE) [1, 4], key-dependent message (KDM)-secure encryption [20], and oblivious transfer (OT) [9]. Keyed-2LHE proposed in this paper is also constructed using HPS. In this section, we introduce definitions, properties, and constructions of HPS.

3.1 Definitions for HPS

Definition 8 (hash proof system)**.** *Let X, Π be finite sets and $L \subset X$ be a non-empty subset. We assume that any element $x \in L$ has a witness w to ensure that x belongs to L, and that a random element of L can be efficiently sampled together with its witness. Then* Hash Proof System (HPS) $\boldsymbol{P} = (X, L, \Pi)$ *is defined by specifying the following five algorithms.*

- $\mathsf{SetUp}(1^\ell)$ *takes security parameter 1^ℓ as an input and outputs public parameter* pp*, which includes descriptions of sets X and L.*
- $\mathsf{HashKG}(\mathsf{pp})$ *takes* pp *as an input and outputs a secret key* hk*.*
- $\mathsf{ProjKG}(\mathsf{hk})$ *takes* hk *as an input and outputs the corresponding public key* hp*.*
- $\mathsf{Hash}(\mathsf{hk}, x)$ *takes* hk *and $x \in X$ as inputs and outputs the corresponding hash value $\pi \in \Pi$.*
- $\mathsf{ProjHash}(\mathsf{hp}, x, w)$ *takes* hp*, $x \in L$, and its witness w as inputs and outputs the corresponding hash value $\pi \in \Pi$.*

We define W, K, S to be the sets consisting of all the possible values of witness w, secret keys hk*, and public keys* hp*, respectively.*

We may omit input hk in the algorithm Hash or inputs hp, w in $\mathsf{ProjHash}$ when they are obvious from the context. Also, when we give concrete constructions of the above algorithms, we may omit the description on SetUp if it is obvious.

HPS is required to satisfy the correctness as follows.

Definition 9 (correctness of HPS)**.** *We say that HPS is* correct*, when for* $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp})$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$*, and for any $x \in L$ and its witness w, we have* $\mathsf{Hash}(\mathsf{hk}, x) = \mathsf{ProjHash}(\mathsf{hp}, x, w)$*.*

Definition 10 (homomorphic HPS)**.** *We say that HPS is* homomorphic*, when X, Π are abelian groups and* $\mathsf{Hash}(\mathsf{hk}, x) + \mathsf{Hash}(\mathsf{hk}, x') = \mathsf{Hash}(\mathsf{hk}, x + x')$ *holds for any* $\mathsf{hk} \in K$ *and $x, x' \in X$.*

3.2 Properties of HPS

Here, we introduce several properties of HPS, required to construct our Keyed-2LHE.

3.2.1 Smoothness

Definition 11 (smoothness)**.** *HPS is said to be ε-*smooth relative to $X' \subset X$*, when for* $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp})$*, and* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$*, the following two distributions*

$$\{(\mathsf{hp}, x, \mathsf{Hash}(x)) \mid x \leftarrow X' \setminus L\}, \quad \{(\mathsf{hp}, x, \pi) \mid x \leftarrow X' \setminus L, \pi \leftarrow \Pi\}$$

are ε-close. In particular, when ε is negligible in ℓ, we simply say that the HPS is smooth*.*

The smoothness of HPS intuitively means that a hash value of an element in $X' \setminus L$ is indistinguishable from a uniformly random value, even if the public key is given.

3.2.2 Universality

Definition 12 (universal$_1$)**.** *HPS is said to be* ε-universal$_1$*, when for any* $\mathsf{hp} \in S$*,* $x \in X \setminus L$*, and* $\pi \in \Pi$*,*

$$\Pr_{\mathsf{hk} \leftarrow K}[\mathsf{Hash}(\mathsf{hk}, x) = \pi \wedge \mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}] \leq \varepsilon \cdot \Pr_{\mathsf{hk} \leftarrow K}[\mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}]$$

holds. In particular, when ε *is negligible in* ℓ*, we simply say that the HPS is* universal$_1$*.*

The universal$_1$ property intuitively means that the probability for correctly guessing a hash value corresponding to an element in $X \setminus L$ is negligible, even if the public key is given.

Definition 13 (universal$_2$)**.** *HPS is said to be* ε-universal$_2$*, when for any* $\mathsf{hp} \in S$*,* $x, x^* \in X \setminus L$ *(*$x \neq x^*$*), and* $\pi, \pi^* \in \Pi$*,*

$$\begin{aligned}&\Pr_{\mathsf{hk} \leftarrow K}[\mathsf{Hash}(\mathsf{hk}, x) = \pi \wedge \mathsf{Hash}(\mathsf{hk}, x^*) = \pi^* \wedge \mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}] \\ &\leq \varepsilon \cdot \Pr_{\mathsf{hk} \leftarrow K}[\mathsf{Hash}(\mathsf{hk}, x^*) = \pi^* \wedge \mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}]\end{aligned}$$

holds. In particular, when ε *is negligible in* ℓ*, we simply say that the HPS is* universal$_2$*.*

The universal$_2$ property intuitively means that the probability for correctly guessing a hash value corresponding to an element in $X \setminus L$ is negligible even if the public key and an extra "hint" (a pair of another element of $X \setminus L$ and the corresponding hash value) were given.

Next, we introduce a computational variant of the universal$_2$ property proposed in [12].

Definition 14 (first-adaptive computationally universal$_2$)**.** *Let* $\boldsymbol{P}$ *be a HPS. We define the following game between a challenger and an adversary* $\mathcal{A}$*:*

1. *The challenger randomly picks* $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$ *and* $\mathsf{hk} \leftarrow K$*, computes* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$*, and sends* pp, hp *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *queries to* Hash *oracle.* Hash *oracle takes* $x \in X$ *as an input and returns* $\mathsf{Hash}(\mathsf{hk}, x)$ *if* $x \in L$*, and returns* $\perp$ *otherwise.*
3. *At an arbitrary point of the game,* $\mathcal{A}$ *submits* $x^* \in X$ *to the challenger. The challenger responds by sending* $\pi^* = \mathsf{Hash}(\mathsf{hk}, x^*)$ *back to* $\mathcal{A}$*.* $\mathcal{A}$ *is allowed to continue* Hash *queries even after the submission.*
4. $\mathcal{A}$ *outputs* $x \in X \setminus L$ *and* $\pi \in \Pi$ *.*

$\mathcal{A}$ *wins the game if the output of the game satisfies* $x \neq x^*$ *and* $\mathsf{Hash}(\mathsf{hk}, x) = \pi$*. We say that* $\boldsymbol{P}$ *is* first-adaptive computationally universal$_2$*, if the probability for winning the game is negligible in* ℓ *for any PPT algorithm* $\mathcal{A}$*.*

There is a weaker variant of computational universal$_2$ property called first-uniform computationally universal$_2$ property [12]. To emphasize the difference from a computational definition, we may refer to the original universal$_2$ property as information-thoretically universal$_2$.

3.2.3 Subset Membership Problem

Definition 15 (hardness of a subset membership problem)**.** *We say that* subset membership problem *of the HPS is* hard relative to $X' \subset X$*, when for* $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$*, two distributions*

$$\{(\mathsf{pp}, x) \mid x \leftarrow L\}, \quad \{(\mathsf{pp}, x) \mid x \leftarrow X' \setminus L\}$$

are computationally indistinguishable.

The hardness of a subset membership problem intuitively means that an element of L and an element of $X' \setminus L$ cannot be distinguished in polynomial time. We note that there are cases where distinction is possible given an additional information. This situation is formalized in the definition below.

Definition 16 (trapdoor subset membership problem)**.** *When HPS has the following two PPT algorithms in addition to the five algorithms in Definition 8, we say that the subset membership problem associated to the HPS has a* trapdoor*:*

- $\mathsf{TrapdoorSetUp}(1^\ell)$ *takes security parameter* 1^ℓ *as an input, and outputs public parameter* pp *together with a trapdoor* τ*.*
- $\mathsf{Distinguish}(x, \tau)$ *takes* $x \in X$ *and a trapdoor* τ *as inputs and decides whether* $x \in L$ *or not.*

3.2.4 Relationships between the Properties of HPS

The smoothness and the universal$_1$ peoperty satisfy the following relationship.

Proposition 2. *If an HPS is* ε*-universal*$_1$*, then it is* $((\varepsilon|\Pi| - 1)(|\Pi| - 1)/2)$*-smooth. Moreover, if the HPS is 0-smooth, then it is* $(1/|\Pi|)$*-universal*$_1$*.*

Proof. The statement follows immediately from Lemmas 1 and 2 of [12]. □

Also, there are the following relationships between different versions of universality.

Proposition 3.
1. *If an HPS is universal*$_2$*, then it is universal*$_1$*.*
2. *If an HPS is universal*$_2$*, then it is first-adaptive computationally universal*$_2$*.*
3. *If an HPS is first-adaptive computationally universal*$_2$ *and* $X' \setminus L$ *is approximately samplable relative to* X*, then it is also first-uniform computationally universal*$_2$ *relative to* X'*.*

Proof. The Part 1 follows immediately from the definitions of universal$_1$ and universal$_2$ properties. The Parts 2 and 3 follow from Lemmas 4 and 5 in [12]. □

3.3 Construction of HPS based on Diverse Vector Space

As a generic construction of HPS, a construction based on a diverse group system is proposed in [11]. However, since we only deal with the construction based on cyclic groups, it is more convenient to think about the special case of diverse group system called diverse vector space [1, 4]. All the HPSs used for our proposed scheme are based on this framework.

The definition of a diverse vector space is given below. Note that this definition is slightly modified from the definition in [1] in order to make our argument simpler.

Definition 17 (diverse vector space)**.** *Let* $\mathbb{G}$ *be a cyclic group of prime order* p*,* $X = \mathbb{G}^n$*, and* $L = \langle \boldsymbol{g}_1, \ldots, \boldsymbol{g}_d \rangle \subset X$*. In this situation, any homomorphism* ϕ *from* X *to* $\Pi = \mathbb{G}$ *can be represented as* $\phi(\boldsymbol{x}) = \boldsymbol{k}^\top \boldsymbol{x}$ *for some* $\boldsymbol{k} \in \mathbb{Z}_p^n$*. Therefore,* $\mathrm{Hom}(X, \Pi)$ *can be identified with* $K = \mathbb{Z}_p^n$*. We call the tuple* (K, X, L, Π) *a* diverse vector space*.*

We will mostly handle the case with $d = 1$, i.e., $X = \mathbb{G}^n$ and $L = \langle \boldsymbol{g} \rangle$. In the remaining part of this section, we assume that $d = 1$ unless otherwise noted.

Given a diverse vector space (K, X, L, Π), we can construct an HPS by defining the algorithms as follows. Note that the witness for $\boldsymbol{x} \in L$ can be set as $w \in \mathbb{Z}_p$ that satisfies $\boldsymbol{x} = w\boldsymbol{g}$.

- $\mathsf{HashKG}(\mathsf{pp})$ takes the public parameter pp as an input and outputs $\mathsf{hk} = \boldsymbol{k} \leftarrow K$.
- $\mathsf{ProjKG}(\mathsf{hk})$ takes the secret key $\mathsf{hk} = \boldsymbol{k}$ as an input and outputs a public key $\mathsf{hp} = s := \boldsymbol{k}^\top \boldsymbol{g}$.
- $\mathsf{Hash}(\mathsf{hk}, \boldsymbol{x})$ takes $\mathsf{hk} = \boldsymbol{k}$ and $\boldsymbol{x} \in X$ as inputs and outputs a hash value $\pi = \boldsymbol{k}^\top \boldsymbol{x}$.
- $\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{x}, w)$ takes a public key $\mathsf{hp} = s$ and $\boldsymbol{x} \in L$ as inputs together with its witness w, and outputs a hash value $\pi = ws$.

The correctness of the above construction can be shown by checking

$$\mathsf{Hash}(\boldsymbol{x}) = \boldsymbol{k}^\top \boldsymbol{x} = \boldsymbol{k}^\top (w\boldsymbol{g}) = w(\boldsymbol{k}^\top \boldsymbol{g}) = ws = \mathsf{ProjHash}(\boldsymbol{x}, w) \ .$$

Moreover, the above HPS satisfies the following properties.

Proposition 4. *The above HPS is* 0*-smooth and* $(1/p)$*-universal*$_1$. *Also, the subset membership problem associated to the HPS has a trapdoor.*

Proof. The smoothness and the universal$_1$ property follow from Example 1 in Section 7.4.1 of [11].

We can check that the subset membership problem has a trapdoor, by giving two algorithms in Definition 16 as follows:

- $\mathsf{TrapdoorSetUp}(1^\ell)$ generates a generator $\boldsymbol{g}$ of L by calculating

$$\boldsymbol{g} = (g, \tau_1 \cdot g, \tau_2 \cdot g, \ldots, \tau_{n-1} \cdot g) \text{ where } g \leftarrow \mathbb{G} \setminus \{0\}, \tau_1, \ldots, \tau_{n-1} \leftarrow \mathbb{Z}_p$$

  and outputs $\tau := (\tau_1, \ldots, \tau_{n-1})$ as a trapdoor.

- $\mathsf{Distinguish}(\boldsymbol{x}, \tau)$ takes $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in X$ and $\tau = (\tau_1, \ldots, \tau_{n-1})$ as inputs, and check if $x_i = \tau_i \cdot x_0$ holds for $i = 1, \ldots, n-1$. If all the conditions are satisfied, the algorithm decides $\boldsymbol{x} \in L$, and otherwise decides $\boldsymbol{x} \notin L$.

Hence the claim holds. □

Also, we can construct universal$_2$ HPS by combining the above construction with a hash function. Let E be a finite set, and modify the definitions of X and L by $X = \mathbb{G}^n \times E$, $L = \langle \boldsymbol{g} \rangle \times E$. In addition, let $\Gamma \colon X \to \mathbb{Z}_p^m$ be a hash function.

- $\mathsf{HashKG}(\mathsf{pp})$ outputs $\mathsf{hk} = (\boldsymbol{k}_0, \ldots, \boldsymbol{k}_m) \leftarrow (\mathbb{Z}_p^n)^{m+1}$.

- $\mathsf{ProjKG}(\mathsf{hk})$ takes hk as an input and outputs $\mathsf{hp} = (s_0, \ldots, s_m) := (\boldsymbol{k}_0^\top \boldsymbol{g}, \ldots, \boldsymbol{k}_m^\top \boldsymbol{g})$.

- $\mathsf{Hash}(\mathsf{hk}, \boldsymbol{x})$ takes hk and $(\boldsymbol{x}, e) \in X$ as inputs, calculates $\Gamma(\boldsymbol{x}, e) = (\gamma_1, \ldots, \gamma_m)$, then outputs

$$\pi = \boldsymbol{k}_0^\top \boldsymbol{x} + \sum_{i=1}^{m} \gamma_i \boldsymbol{k}_i^\top \boldsymbol{x}$$

  as a hash value.

- $\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{x}, w)$ takes hp, $(\boldsymbol{x}, e) \in L$, and the corresponding witness w as inputs, calculates $\Gamma(\boldsymbol{x}, e) = (\gamma_1, \ldots, \gamma_m)$, and outputs

$$\pi = ws_0 + \sum_{i=1}^{m} \gamma_i w s_i$$

  as a hash value.

Proposition 5. *The above construction of HPS satisfies the following:*

1. *If* Γ *is injective, then the HPS is information-theoretically universal*$_2$.

2. *If* Γ *is sampled from a family of collision resistant hash functions, then the HPS is first-adaptive computationally universal*$_2$.

Proof. The statement follows immediately from Proposition 1 of [12]. □

Also, we note that when Γ is sampled from a family of target collision resistant hash functions, the above HPS satisfies first-uniform computationally universal$_2$ property (see [12] for the detail).

3.4 Direct Product of HPS

In this section, we define direct product of HPS, and describe some properties. We note that a notion of direct product of HPS is also defined in [1], but their definition slightly differs from ours.

Suppose that two HPSs $\boldsymbol{P}_1 = (X_1, L_1, \Pi_1)$, $\boldsymbol{P}_2 = (X_2, L_2, \Pi_2)$ are given. We denote algorithms and sets related to each HPS by putting the corresponding subscript. In this situation, we can construct a new HPS $\boldsymbol{P} = (X_1 \times X_2, L_1 \times L_2, \Pi_1 \times \Pi_2)$ by defining the algorithms in the following manner.

- $\mathsf{HashKG}(\mathsf{pp})$ calculates $\mathsf{hk}_i \leftarrow \mathsf{HashKG}_i(\mathsf{pp})$ $(i = 1, 2)$, and outputs $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$.
- $\mathsf{ProjKG}(\mathsf{hk})$ takes $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$ as an input, calculates $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}_i(\mathsf{hk}_i)$ $(i = 1, 2)$, and outputs $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$.
- $\mathsf{Hash}(\mathsf{hk}, x)$ takes $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$ and $x = (x_1, x_2) \in X_1 \times X_2$ as inputs, calculates $\pi_i \leftarrow \mathsf{Hash}_i(\mathsf{hk}_i, x_i)$ $(i = 1, 2)$, and outputs $\pi = (\pi_1, \pi_2)$.
- $\mathsf{ProjHash}(\mathsf{hp}, x, w)$ takes $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$, $x = (x_1, x_2) \in L_1 \times L_2$ and the pair of the corresponding witnesses $w = (w_1, w_2)$ as inputs, calculates $\pi_i \leftarrow \mathsf{ProjHash}_i(\mathsf{hp}_i, x_i, w_i)$ $(i = 1, 2)$, and outputs $\pi = (\pi_1, \pi_2)$.

Proposition 6. *The HPS $\boldsymbol{P}$ constructed as above satisfies the following:*

1. *If $\boldsymbol{P}_1$ is ε_1-smooth and $\boldsymbol{P}_2$ is ε_2-smooth, then $\boldsymbol{P}$ is $(\varepsilon_1+\varepsilon_2)$-smooth relative to $X' = (X_1 \setminus L_1) \times (X_2 \setminus L_2)$.*
2. *If $\boldsymbol{P}_1$ is ε_1-universal$_1$ and $\boldsymbol{P}_2$ is ε_2-universal$_1$, then $\boldsymbol{P}$ is $(\max\{\varepsilon_1, \varepsilon_2\})$-universal$_1$.*
3. *If $\boldsymbol{P}_1$ is ε_1-universal$_2$ and $\boldsymbol{P}_2$ is ε_2-universal$_2$, then $\boldsymbol{P}$ is $(\max\{\varepsilon_1, \varepsilon_2\})$-universal$_2$.*

Proof. For Part 1, first, it can be shown that for $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp})$, and $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$,

$$\begin{aligned}
&\{(\mathsf{hp}, x, \mathsf{Hash}(x)) \mid x \leftarrow X' \setminus L\} \\
&= \{((\mathsf{hp}_1, \mathsf{hp}_2), (x_1, x_2), (\mathsf{Hash}_1(x_1), \mathsf{Hash}_2(x_2))) \mid x_i \leftarrow X_i \setminus L_i\} \\
&\overset{s}{\approx} \{((\mathsf{hp}_1, \mathsf{hp}_2), (x_1, x_2), (\pi_1, \mathsf{Hash}_2(x_2))) \mid x_i \leftarrow X_i \setminus L_i, \pi_1 \leftarrow \Pi_1\} \\
&\overset{s}{\approx} \{((\mathsf{hp}_1, \mathsf{hp}_2), (x_1, x_2), (\pi_1, \pi_2)) \mid x_i \leftarrow X_i \setminus L_i, \pi_i \leftarrow \Pi_i\} \\
&= \{(\mathsf{hp}, x, \pi) \mid x \leftarrow X' \setminus L, \pi \leftarrow \Pi\}
\end{aligned}$$

holds, by using smoothness of $\boldsymbol{P}_1$ between the second and the third lines and the smoothness of $\boldsymbol{P}_2$ between the third and the fourth lines, respectively. Since the statistical distance between the second and the third lines is at most ε_1 and the statistical distance between the third and the fourth lines is at most ε_2, the statement holds.

For Part 2, let $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \in S$, $x = (x_1, x_2) \in X \setminus L$, and $\pi = (\pi_1, \pi_2) \in \Pi$. Since $(x_1, x_2) \in X \setminus L$, at least one of $x_1 \in X_1 \setminus L_1$ and $x_2 \in X_2 \setminus L_2$ holds. When the former condition is satisfied,

$$\Pr_{\mathsf{hk}_1 \leftarrow K_1}[\mathsf{Hash}_1(\mathsf{hk}_1, x_1) = \pi_1 \wedge \mathsf{ProjKG}_1(\mathsf{hk}_1) = \mathsf{hp}_1] \leq \varepsilon_1 \cdot \Pr_{\mathsf{hk}_1 \leftarrow K_1}[\mathsf{ProjKG}_1(\mathsf{hk}_1) = \mathsf{hp}_1]$$

holds from the universal$_1$ property of $\boldsymbol{P}_1$. Therefore, we obtain

$$\begin{aligned}
&\Pr_{\mathsf{hk} \leftarrow K}[\mathsf{Hash}(\mathsf{hk}, x) = \pi \wedge \mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}] \\
&= \Pr_{\mathsf{hk}_1 \leftarrow K_1}[\mathsf{Hash}_1(\mathsf{hk}_1, x_1) = \pi_1 \wedge \mathsf{ProjKG}_1(\mathsf{hk}_1) = \mathsf{hp}_1] \\
&\quad \cdot \Pr_{\mathsf{hk}_2 \leftarrow K_2}[\mathsf{Hash}_2(\mathsf{hk}_2, x_2) = \pi_2 \wedge \mathsf{ProjKG}_2(\mathsf{hk}_2) = \mathsf{hp}_2] \\
&\leq \varepsilon_1 \cdot \Pr_{\mathsf{hk}_1 \leftarrow K_1}[\mathsf{ProjKG}_1(\mathsf{hk}_1) = \mathsf{hp}_1] \cdot \Pr_{\mathsf{hk}_2 \leftarrow K_2}[\mathsf{ProjKG}_2(\mathsf{hk}_2) = \mathsf{hp}_2] \\
&= \varepsilon_1 \cdot \Pr_{\mathsf{hk} \leftarrow K}[\mathsf{ProjKG}(\mathsf{hk}) = \mathsf{hp}] \ ,
\end{aligned}$$

and the ε_1-universal$_1$ property of $\boldsymbol{P}$ follows. Similarly, when the latter condition holds, we can show ε_2-universal$_1$ property of $\boldsymbol{P}$ from the universal$_1$ property of $\boldsymbol{P}_2$. Combining the discussions above, the statement follows. Part 3 can be shown in the same manner as Part 2. $\square$

Proposition 7. *If both $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are first-adaptive computationally universal$_2$ and the subset membership problem for each of $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ has a trapdoor, then $\boldsymbol{P}$ is first-adaptive computationally universal$_2$ and its subset membership problem also has a trapdoor.*

Proof. The statement for the existence of a trapdoor for $\boldsymbol{P}$ holds obviously. Let $\mathcal{A}$ be an adversary in the first-adaptive computationally universal$_2$ game for $\boldsymbol{P}$ (denoted here by $\mathcal{G}$). Then for each $i = 1, 2$, we construct an adversary $\mathcal{B}_i$ in the first-adaptive computationally universal$_2$ game for $\boldsymbol{P}_i$ (denoted here by $\mathcal{G}_i$) as follows:

1. $\mathcal{B}_i$ receives pp_i and hp_i for $\boldsymbol{P}_i$ from the challenger in the game $\mathcal{G}_i$, while $\mathcal{B}_i$ generates pp_{3-i}, hk_{3-i}, and a trapdoor τ_{3-i} for $\boldsymbol{P}_{3-i}$. Then $\mathcal{B}_i$ sends $\mathsf{pp} = (\mathsf{pp}_1, \mathsf{pp}_2)$ and $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$ to $\mathcal{A}$.

2. $\mathcal{B}_i$ simulates the game $\mathcal{G}$ for $\mathcal{A}$. During the game, when $\mathcal{A}$ makes a Hash query with input $x = (x_1, x_2) \in X$, $\mathcal{B}_i$ makes Hash query in the game $\mathcal{G}_i$ with input x_i and obtains the response ρ_i. On the other hand, $\mathcal{B}_i$ decides, by using the trapdoor τ_{3-i}, whether $x_{3-i} \in L_{3-i}$ or not. If $x_{3-i} \in L_{3-i}$, then $\mathcal{B}_i$ computes $\rho_{3-i} \leftarrow \mathsf{Hash}_{3-i}(\mathsf{hk}_{3-i}, x_{3-i})$; while if $x_{3-i} \notin L_{3-i}$, then $\mathcal{B}_i$ sets $\rho_{3-i} = \bot$. Finally, $\mathcal{B}_i$ sends (ρ_1, ρ_2) back to $\mathcal{A}$ if $\rho_1 \neq \bot$ and $\rho_2 \neq \bot$, while otherwise $\mathcal{B}_i$ sends $\bot$ back to $\mathcal{A}$.

3. When $\mathcal{A}$ submits $x^* = (x_1^*, x_2^*)$ to $\mathcal{B}_i$, $\mathcal{B}_i$ submits x_i^* to the challenger in $\mathcal{G}_i$ and obtains the response π_i^*, while $\mathcal{B}_i$ computes $\pi_{3-i}^* \leftarrow \mathsf{Hash}_{3-i}(\mathsf{hk}_{3-i}, x_{3-i}^*)$. Then $\mathcal{B}_i$ sends $\pi^* = (\pi_1^*, \pi_2^*)$ back to $\mathcal{A}$.

4. When $\mathcal{B}_i$ receives $x = (x_1, x_2)$ and $\pi = (\pi_1, \pi_2)$ as outputs of $\mathcal{A}$, $\mathcal{B}_i$ outputs (x_i, π_i) as an output of the game $\mathcal{G}_i$.

The distribution of $\mathcal{A}$'s output (x, π) above is identical to the real game $\mathcal{G}$ by the property of the trapdoor τ_{3-i} for $\boldsymbol{P}_{3-i}$.

Let C be the event that $\mathcal{A}$ wins the game $\mathcal{G}$, and for $i = 1, 2$, let C_i be the event that $\mathcal{B}_i$ wins the game $\mathcal{G}_i$. Now when the event C occurs, we have $x \neq x^*$, therefore we have $x_1 \neq x_1^*$ or $x_2 \neq x_2^*$. Hence $\Pr[C] \leq \Pr[C \wedge x_1 \neq x_1^*] + \Pr[C \wedge x_2 \neq x_2^*]$. Moreover, by the definition of $\mathcal{B}_i$, if C occurs and $x_i \neq x_i^*$, then $\mathcal{B}_i$ wins the game $\mathcal{G}_i$. Therefore we have $\Pr[C \wedge x_i \neq x_i^*] \leq \Pr[C_i]$. Summarizing, we have $\Pr[C] \leq \Pr[C_1] + \Pr[C_2]$. If $\mathcal{A}$ is PPT, then both $\mathcal{B}_1$ and $\mathcal{B}_2$ are also PPT. Hence $\Pr[C_1]$ and $\Pr[C_2]$ are negligible by the assumption, therefore $\Pr[C]$ is also negligible, as desired. Hence the statement holds. $\square$

4 Keyed-Homomorphic Public Key Encryption (KH-PKE)

4.1 Definition of KH-PKE

Definition 18 (syntax of KH-PKE)**.** *Let $\mathcal{M}$ be a message space. KH-PKE is defined by five algorithms below.*

- $\mathsf{ParamGen}(1^\ell)$ *takes security paramater 1^ℓ as an input and outputs public parameter* pp.
- $\mathsf{KeyGen}(\mathsf{pp})$ *takes* pp *as an input and outputs three keys* $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ *takes public key* pk *and a message $m \in \mathcal{M}$ as inputs and outputs ciphertext C.*
- $\mathsf{Dec}(\mathsf{sk}, C)$ *takes secret key* sk *and a ciphertext C as inputs and outputs plaintext $m \in \mathcal{M}$ or $\bot$, which represents a failure of decryption.*
- $\mathsf{Eval}(\mathsf{ek}, f, C, C')$ *takes evaluation key* ek*, operation $f \colon \mathcal{M}^2 \to \mathcal{M}$ and two ciphertexts C, C' as inputs and outputs a ciphertext C'' or $\bot$, which represents a failure of evaluation.*

KH-PKE is required to fulfill correctness.

Definition 19 (correctness of KH-PKE)**.** *Let* $\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\ell)$ *and* $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*. We say that KH-PKE is* correct*, when it satisfies the following:*

- *(Correctness of* Dec*) For any* $m \in \mathcal{M}$ *and* $C \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*,* $\mathsf{Dec}(\mathsf{sk}, C) = m$ *holds.*
- *(Correctness of* Eval*) For any* $m, m' \in \mathcal{M}$*, operation* f*,* $C \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*,* $C' \leftarrow \mathsf{Enc}(\mathsf{pk}, m')$*, and* $C'' \leftarrow \mathsf{Eval}(\mathsf{ek}, f, C, C')$*,* $\mathsf{Dec}(\mathsf{sk}, C'') = f(m, m')$ *holds.*

The notion of KH-CCA security is defined as follows.

Definition 20 (KH-CCA security)**.** *We say that KH-PKE scheme is* KH-CCA secure*, when for any PPT adversary* $\mathcal{A}$*, its advantage*

$$\Big| \Pr[\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\ell); (\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); (m_0^*, m_1^*, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, \mathsf{pk});$$

$$b \leftarrow \{0,1\}; C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*); b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, \mathsf{st}, C^*) \colon b = b'] - \frac{1}{2} \Big|$$

is negligible in ℓ*. Here* $\mathcal{O}$ *denotes oracles that the adversary can use, which consists of three oracles* RevEK*,* Dec*,* Eval*. The details of these oracles are given below. Also, we suppose that a list* List *of ciphertexts is used throughout the KH-CCA game, which is set to* $\emptyset$ *in the* find *stage and set to* $\mathsf{List} = (C^*)$ *at the beginning of the* guess *stage.*

- RevEK *returns evaluation key* ek*.*
- Dec *takes a ciphertext* C *as input and returns* $\perp$ *if* $C \in \mathsf{List}$*, and* $\mathsf{Dec}(\mathsf{sk}, C)$ *if* $C \notin \mathsf{List}$*.*
- Eval *takes two ciphertexts* C, C' *and an operation* f *as inputs and returns* $C'' = \mathsf{Eval}(\mathsf{ek}, f, C, C')$*. If* $C'' \neq \perp$ *and either* C *or* C' *is in* List*,* Eval *appends* C'' *to* List*.*

We also have the following constraints about oracle queries: RevEK *can be queried only once;* Eval *cannot be queried after* RevEK *has been queried; and* Dec *cannot be queried if* RevEK *is already queried and* $\mathcal{A}$ *has already received* C^* *from the challenger.*

Let us give some supplement about the List in the definition above. In the KH-CCA game, if the adversary queries the challenge ciphertext C^* to Eval oracle and queries its result to Dec oracle, it is obvious that the adversary can always win the game (this is essentially the same reason as the fact that HE cannot achieve IND-CCA2 security). To avoid this trivial attack, ciphertexts evaluated from C^* is recorded in the List, and ciphertext in the List are forbidden to be queried to the Dec oracle.

We note that KH-CCA security satisfies IND-CCA1 security against an adversary who has ek at the beginning of the game, and IND-CCA2 security against an adversary who never obtains ek.

4.2 Generic Construcion of KH-PKE

Emura et al. [12] gave a generic construction of KH-PKE equipped with KH-CCA security, using HPS as a building block. We review their construction briefly in this section.

To construct KH-PKE, three HPSs $\boldsymbol{P}, \widehat{\boldsymbol{P}}, \widetilde{\boldsymbol{P}}$ are required. They are supposed to satisfy the following properties:

- $\boldsymbol{P} = (X, L, \Pi)$ is homomorphic and smooth, and its subset membership problem is hard.
- $\widehat{\boldsymbol{P}} = (X, L, \widehat{\Pi})$ is homomorphic and universal$_1$.
- $\widetilde{\boldsymbol{P}} = (X \times \Pi \times \widehat{\Pi}, L \times \Pi \times \widehat{\Pi}, \widetilde{\Pi})$ is universal$_2$.

```
$\mathsf{KeyGen}(1^\ell)$:
    $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$
    $\widehat{\mathsf{hk}} \leftarrow \widehat{\mathsf{HashKG}}(\mathsf{pp}), \widehat{\mathsf{hp}} \leftarrow \widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}})$
    $\widetilde{\mathsf{hk}} \leftarrow \widetilde{\mathsf{HashKG}}(\mathsf{pp}), \widetilde{\mathsf{hp}} \leftarrow \widetilde{\mathsf{ProjKG}}(\widetilde{\mathsf{hk}})$
    Output $\mathsf{pk} = (\mathsf{hp}, \widehat{\mathsf{hp}}, \widetilde{\mathsf{hp}}), \mathsf{sk} = (\mathsf{hk}, \widehat{\mathsf{hk}}, \widetilde{\mathsf{hk}}), \mathsf{ek} = \widetilde{\mathsf{hk}}$

$\mathsf{Enc}(\mathsf{pk}, m)$:
    Sample $x \in L$ together with its witness $w$
    $e \leftarrow m + \mathsf{ProjHash}(x, w)$
    $\widehat{\pi} \leftarrow \widehat{\mathsf{ProjHash}}(x, w)$
    $\widetilde{\pi} \leftarrow \widetilde{\mathsf{ProjHash}}((x, e, \widehat{\pi}), w)$
    Output $C = (x, e, \widehat{\pi}, \widetilde{\pi})$
```

```
$\mathsf{Dec}(\mathsf{sk}, C) : C = (x, e, \widehat{\pi}, \widetilde{\pi})$
    If $\widehat{\pi} \neq \widehat{\mathsf{Hash}}(x)$ or $\widetilde{\pi} \neq \widetilde{\mathsf{Hash}}(x, e, \widehat{\pi})$, output $\perp$
    Output $m \leftarrow e - \mathsf{Hash}(\mathsf{hk}, x)$

$\mathsf{Eval}(\mathsf{ek}, C, C') : C = (x, e, \widehat{\pi}, \widetilde{\pi}), C' = (x', e', \widehat{\pi}', \widetilde{\pi}')$
    If $\widetilde{\pi} \neq \widetilde{\mathsf{Hash}}(x, e, \widehat{\pi})$, output $\perp$
    If $\widetilde{\pi}' \neq \widetilde{\mathsf{Hash}}(x', e', \widehat{\pi}')$, output $\perp$
    Sample $x_0 \in L$ together with its witness $w_0$
    $x'' \leftarrow x + x' + x_0$, $e'' \leftarrow e + e' + \mathsf{ProjHash}(x_0, w_0)$
    $\widehat{\pi}'' \leftarrow \widehat{\pi} + \widehat{\pi}' + \widehat{\mathsf{ProjHash}}(x_0, w_0)$
    $\widetilde{\pi}'' \leftarrow \widetilde{\mathsf{Hash}}(x'', e'', \widehat{\pi}'')$
    Output $C'' = (x'', e'', \widehat{\pi}'', \widetilde{\pi}'')$
```

Figure 1: Generic construction of KH-PKE based on HPS

For $\widehat{\boldsymbol{P}}, \widetilde{\boldsymbol{P}}$, we indicate their sets and algorithms by putting hat and tilde, respectively.

Using above three HPSs, we can construct KH-PKE with plaintext space $\mathcal{M} = \Pi$, as in Figure 1. Note that in Eval, $x_0 \leftarrow L$ and its hash values are added to calculate $x'', e'', \widehat{\pi}''$. This corresponds to the operation of "adding ciphertext of 0" and therefore it does not affect correctness; this is important for the security. We refer to this operation as *rerandomization*, from now on.

The correctness of the construction follows immediately from the correctness and the homomorphic property of HPS.

Theorem 1. *KH-PKE constructed above satisfies KH-CCA security.*

Proof. It follows immediately from Theorem 1 of [12]. □

Intuitively, $\boldsymbol{P}$ is used to hide information of the plaintext. The smoothness and the hardness of the subset membership problem of $\boldsymbol{P}$ guarantee that the hash value used to mask the plaintext is indistinguishable from a uniformly random value. Also, $\widehat{\boldsymbol{P}}$ guarantees security against an adversary who has ek, and $\widetilde{\boldsymbol{P}}$ guarantees security against an adversary who does not have ek. The universal property of $\widehat{\boldsymbol{P}}$ and $\widetilde{\boldsymbol{P}}$ means that the adversary cannot forge a hash value for a ciphertext calculated in a correct manner.

We required information-theoretic universal$_2$ property for $\widetilde{\boldsymbol{P}}$, but it can be weakened to computational one in the following manner while retaining KH-CCA security.

- $\widetilde{\boldsymbol{P}}$ is first-adaptive computationally universal$_2$ and $X' \setminus L$ is approximately samplable relative to X.

Also, by assuming some more conditions, we can further weaken universal$_2$ property of $\widetilde{\boldsymbol{P}}$ to first-uniform computationally universal$_2$ property. See [12] for the details.

5 Keyed Two-Level Homomorphic Encryption (Keyed-2LHE)

5.1 Syntax of Keyed-2LHE

The syntax of Keyed-2LHE is formalized as follows.

Definition 21 (syntax of Keyed-2LHE)**.** *When a KH-PKE scheme satisfies the following conditions, we call it* Keyed-2LHE.

- *The message space is $\mathcal{M} = \{(i, m) \mid i \in \{1, 2\}, m \in \mathcal{M}'\}$, where $\mathcal{M}'$ is a ring. We set $\mathcal{M}_i = \{i\} \times \mathcal{M}'$ for $i = 1, 2$.*

- *An operation f is one of* $\mathsf{Add}^{(1)} \colon \mathcal{M}_1^2 \to \mathcal{M}_1$, $\mathsf{Add}^{(2)} \colon \mathcal{M}_2^2 \to \mathcal{M}_2$, *and* $\mathsf{Mult} \colon \mathcal{M}_1^2 \to \mathcal{M}_2$, *which represent the following operations:*

$$\begin{aligned}\mathsf{Add}^{(1)} &\colon (1,m),(1,m') \mapsto (1, m+m') \ , \\ \mathsf{Add}^{(2)} &\colon (2,m),(2,m') \mapsto (2, m+m') \ , \\ \mathsf{Mult} \ &\colon (1,m),(1,m') \mapsto (2, mm') \ .\end{aligned}$$

In a Keyed-2LHE scheme, we refer to a ciphertext corresponding to a plaintext with $i = 1$ as level-1 ciphertext. Level-2 ciphertext is defined similarly.

KH-CCA security for a Keyed-2LHE scheme is defined in the following manner.

Definition 22 (KH-CCA secure Keyed-2LHE)**.** *We say that a Keyed-2LHE scheme is* KH-CCA secure*, when for any PPT adversary $\mathcal{A}$, its advantage given by*

$$\Bigg| \Pr[\mathsf{pp} \leftarrow \mathsf{ParamGen}(1^\ell); (\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); (i^*, m_0^*, m_1^*, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find}, \mathsf{pk});$$

$$b \leftarrow \{0,1\}; C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, (i^*, m_b^*)); b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, \mathsf{st}, C^*) \colon b = b'] - \frac{1}{2} \Bigg|$$

is negligible in ℓ. Here $\mathcal{O}$ represents the oracles available to the adversary, which is given in the same manner as the original KH-CCA security.

We note that the adversary specifies the level of the challenge ciphertext in the KH-CCA game for Keyed-2LHE.

5.2 Overview of Our Construction

When focusing on each level, our scheme mostly follows the generic construction in Section 4.2, i.e., there are three HPSs (corresponding to $\boldsymbol{P}, \widehat{\boldsymbol{P}}, \widetilde{\boldsymbol{P}}$ in Section 4.2) for each level. Therefore, we require six HPSs in total to construct our scheme.

One of the differences from the construction in [12] is that the message space in our scheme is changed to $\mathcal{M}' = \mathbb{Z}_p$, and the construction of the ciphertext is slightly modified. (This is similar to the modification from ElGamal cryptosystem to so-called lifted-ElGamal cryptosystem.) To apply this modification, a restriction on the message space is required. We will discuss the details in Section 6. Another difference is that for HPSs $\boldsymbol{P}$ and $\widehat{\boldsymbol{P}}$, key generation algorithms $\mathsf{HashKG}, \mathsf{ProjKG}$ are common in both levels, and a hash value for the level-2 HPS can be calculated from those for the level-1 HPS by applying pairings. These properties are necessary for computing multiplications.

From now on, when we want to specify levels of HPS or the corresponding sets and algorithms, we denote this by using superscripts "(1)" and "(2)".

5.3 Construction of Hash Proof System

In this section, we give concrete constructions of HPSs for the proposed scheme.

All the HPSs described here computes $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathsf{GenBG}(1^\ell)$, chooses $g_1' \leftarrow \mathbb{G}_1$, $g_2' \leftarrow \mathbb{G}_2$ and sets $\boldsymbol{g}_1 := (g_1, g_1') \in \mathbb{G}_1^2$, $\boldsymbol{g}_2 := (g_2, g_2') \in \mathbb{G}_2^2$. Then, as pp, $\mathsf{SetUp}(1^\ell)$ outputs these together with $g_T := g_1 \otimes g_2$, $\boldsymbol{h}_1 := \boldsymbol{g}_1 \otimes \boldsymbol{g}_2$, $\boldsymbol{h}_2 := \boldsymbol{g}_1 \otimes g_2$, and $\boldsymbol{h}_3 := g_1 \otimes \boldsymbol{g}_2$. For universal$_2$ HPSs, we assume that the setup algorithm also outputs hash functions necessary for the construction.

5.3.1 HPS for Level-1 Ciphertexts

Let $X = \mathbb{G}_1^2 \times \mathbb{G}_2^2$ and $L = \langle \boldsymbol{g}_1 \rangle \times \langle \boldsymbol{g}_2 \rangle$. The witness for $x = (\boldsymbol{x}_1, \boldsymbol{x}_2) \in L$ is $w = (w_1, w_2) \in \mathbb{Z}_p^2$ that satisfies $(\boldsymbol{x}_1, \boldsymbol{x}_2) = (w_1 \boldsymbol{g}_1, w_2 \boldsymbol{g}_2)$. Furthermore, let $X' = (\mathbb{G}_1^2 \setminus \langle \boldsymbol{g}_1 \rangle) \times (\mathbb{G}_2^2 \setminus \langle \boldsymbol{g}_2 \rangle)$, $\Pi = \widehat{\Pi} = \widetilde{\Pi} = \mathbb{G}_1 \times \mathbb{G}_2$.

$\boldsymbol{P}^{(1)}$	$\widehat{\boldsymbol{P}}^{(1)}$	$\widetilde{\boldsymbol{P}}^{(1)}$
$\mathsf{HashKG}(\mathsf{pp})$: $\boldsymbol{k}_i \leftarrow \mathbb{Z}_p^2 \ (i=1,2)$ Output $\mathsf{hk} = (\boldsymbol{k}_1, \boldsymbol{k}_2)$	$\widehat{\mathsf{HashKG}}(\mathsf{pp})$: $\widehat{\boldsymbol{k}}_i \leftarrow \mathbb{Z}_p^2 \ (i=1,2)$ Output $\widehat{\mathsf{hk}} = (\widehat{\boldsymbol{k}}_1, \widehat{\boldsymbol{k}}_2)$	$\widetilde{\mathsf{HashKG}}(\mathsf{pp})$: $(\widetilde{\boldsymbol{k}}_{ij})_{j=0}^n \leftarrow (\mathbb{Z}_p^2)^{n+1} \ (i=1,2)$ Output $\widetilde{\mathsf{hk}} = ((\widetilde{\boldsymbol{k}}_{1j})_{j=0}^n, (\widetilde{\boldsymbol{k}}_{2j})_{j=0}^n)$
$\mathsf{ProjKG}(\mathsf{hk})$: $s_i = \boldsymbol{k}_i^\top \boldsymbol{g}_i \ (i=1,2)$ Output $\mathsf{hp} = (s_1, s_2)$	$\widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}})$: $\widehat{s}_i = \widehat{\boldsymbol{k}}_i^\top \boldsymbol{g}_i \ (i=1,2)$ Output $\widehat{\mathsf{hp}} = (\widehat{s}_1, \widehat{s}_2)$	$\widetilde{\mathsf{ProjKG}}(\widetilde{\mathsf{hk}})$: $\widetilde{s}_{ij} = \widetilde{\boldsymbol{k}}_{ij}^\top \boldsymbol{g}_i \ (i=1,2,\ j=0,\dots,n)$ Output $\widetilde{\mathsf{hp}} = ((\widetilde{s}_{1j})_{j=0}^n, (\widetilde{s}_{2j})_{j=0}^n)$
$\mathsf{Hash}(\mathsf{hk}, x)$: $\pi_i = \boldsymbol{k}_i^\top \boldsymbol{x}_i \ (i=1,2)$ Output $\pi = (\pi_1, \pi_2)$	$\widehat{\mathsf{Hash}}(\widehat{\mathsf{hk}}, x)$: $\widehat{\pi}_i = \widehat{\boldsymbol{k}}_i^\top \boldsymbol{x}_i \ (i=1,2)$ Output $\widehat{\pi} = (\widehat{\pi}_1, \widehat{\pi}_2)$	$\widetilde{\mathsf{Hash}}(\widetilde{\mathsf{hk}}, (x, e, \widehat{\pi}))$: $(\gamma_{ij})_{j=1}^n = \Gamma_i(\boldsymbol{x}_i, e_i, \widehat{\pi}_i) \ (i=1,2)$ $\widetilde{\pi}_i = \widetilde{\boldsymbol{k}}_{i0}^\top \boldsymbol{x}_i + \sum_{j=1}^n \gamma_{ij} \widetilde{\boldsymbol{k}}_{ij}^\top \boldsymbol{x}_i \ (i=1,2)$ Output $\widetilde{\pi} = (\widetilde{\pi}_1, \widetilde{\pi}_2)$
$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: $\pi_i = w_i s_i \ (i=1,2)$ Output $\pi = (\pi_1, \pi_2)$	$\widehat{\mathsf{ProjHash}}(\widehat{\mathsf{hp}}, x, w)$: $\widehat{\pi}_i = w_i \widehat{s}_i \ (i=1,2)$ Output $\widehat{\pi} = (\widehat{\pi}_1, \widehat{\pi}_2)$	$\widetilde{\mathsf{ProjHash}}(\widetilde{\mathsf{hp}}, (x, e, \widehat{\pi}), w)$: $(\gamma_{ij})_{j=1}^n = \Gamma_i(\boldsymbol{x}_i, e_i, \widehat{\pi}_i) \ (i=1,2)$ $\widetilde{\pi}_i = w_i \widetilde{s}_{i0} + w_i \sum_{j=1}^n \gamma_{ij} \widetilde{s}_{ij} \ (i=1,2)$ Output $\widetilde{\pi} = (\widetilde{\pi}_1, \widetilde{\pi}_2)$

Figure 2: HPSs for level-1 ciphertexts, where $\Gamma_1 \colon \mathbb{G}_1^4 \to \mathbb{Z}_p^n$ and $\Gamma_2 \colon \mathbb{G}_2^4 \to \mathbb{Z}_p^n$ are hash functions

In this situation, we define three HPSs for level-1 ciphertexts $\boldsymbol{P}^{(1)} = (X, L, \Pi)$, $\widehat{\boldsymbol{P}}^{(1)} = (X, L, \widehat{\Pi})$, $\widetilde{\boldsymbol{P}}^{(1)} = (X \times \Pi \times \widehat{\Pi}, L \times \Pi \times \widehat{\Pi}, \widetilde{\Pi})$ as in Figure 2.

We prove some properties of those HPSs.

Proposition 8. $\boldsymbol{P}^{(1)}$ *satisfies the following:*

1. $\boldsymbol{P}^{(1)}$ *is smooth relative to* X'*.*
2. *Under SXDH assumption, the subset membership problem of* $\boldsymbol{P}^{(1)}$ *is hard relative to* X' *and has a trapdoor.*
3. $X' \setminus L$ *is approximately samplable relative to* X*.*

Proof. Since $\boldsymbol{P}^{(1)}$ can be interpreted as the direct product of two HPSs constructed by applying generic construction based on a diverse vector space, the smoothness follows from Proposition 6.

Furthermore, the hardness of the subset membership problem and the fact that it has a trapdoor follow from Propositions 1 and 4, respectively.

The approximate samplability can be deduced from the fact that the uniform distribution on $X' \setminus L$ is statistically indistinguishable from the uniform distribution on X, as $|X' \setminus L| = p^2(p-1)^2$ and $|X| = p^4$. □

Proposition 9. $\widehat{\boldsymbol{P}}^{(1)}$ *is universal*$_1$*.*

Proof. Since $\widehat{\boldsymbol{P}}^{(1)}$ can be interpreted as the direct product of two $(1/p)$-universal$_1$ HPSs constructed by applying generic construction based on a diverse vector space, the statement follows from Proposition 6. □

Proposition 10. $\widetilde{\boldsymbol{P}}^{(1)}$ *satisfies the following:*

1. *If* Γ_1 *and* Γ_2 *are injective, then* $\widetilde{\boldsymbol{P}}^{(1)}$ *is information-theoretically universal*$_2$*.*

$\boldsymbol{P}^{(2)}$	$\widehat{\boldsymbol{P}}^{(2)}$	$\widetilde{\boldsymbol{P}}^{(2)}$
$\mathsf{HashKG}(\mathsf{pp})$: $\boldsymbol{k}_i \leftarrow \mathbb{Z}_p^2$ $(i=1,2)$ Output $\mathsf{hk} = (\boldsymbol{k}_1, \boldsymbol{k}_2)$	$\widehat{\mathsf{HashKG}}(\mathsf{pp})$: $\widehat{\boldsymbol{k}}_i \leftarrow \mathbb{Z}_p^2$ $(i=1,2)$ Output $\widehat{\mathsf{hk}} = (\widehat{\boldsymbol{k}}_1, \widehat{\boldsymbol{k}}_2)$	$\widetilde{\mathsf{HashKG}}(\mathsf{pp})$: $(\widetilde{\boldsymbol{k}}_{1j})_{j=0}^n \leftarrow (\mathbb{Z}_p^4)^{n+1}$, $(\widetilde{\boldsymbol{k}}_{2j})_{j=0}^n, (\widetilde{\boldsymbol{k}}_{3j})_{j=0}^n \leftarrow (\mathbb{Z}_p^2)^{n+1}$ Output $\widetilde{\mathsf{hk}} = ((\widetilde{\boldsymbol{k}}_{1j})_{j=0}^n, (\widetilde{\boldsymbol{k}}_{2j})_{j=0}^n, (\widetilde{\boldsymbol{k}}_{3j})_{j=0}^n)$
$\mathsf{ProjKG}(\mathsf{hk})$: $s_i = \boldsymbol{k}_i^\top \boldsymbol{g}_i$ $(i=1,2)$ Output $\mathsf{hp} = (s_1, s_2)$	$\widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}})$: $\widehat{s}_i = \widehat{\boldsymbol{k}}_i^\top \boldsymbol{g}_i$ $(i=1,2)$ Output $\widehat{\mathsf{hp}} = (\widehat{s}_1, \widehat{s}_2)$	$\widetilde{\mathsf{ProjKG}}(\widetilde{\mathsf{hk}})$: $\widetilde{s}_{ij} = \widetilde{\boldsymbol{k}}_{ij}^\top \boldsymbol{h}_i$ $(i=1,2,3,\ j=0,\dots,n)$ Output $\widetilde{\mathsf{hp}} = ((\widetilde{s}_{1j})_{j=0}^n, (\widetilde{s}_{2j})_{j=0}^n, (\widetilde{s}_{3j})_{j=0}^n)$
$\mathsf{Hash}(\mathsf{hk}, x)$: $\pi_1 = (\boldsymbol{k}_1 \otimes \boldsymbol{k}_2)^\top \boldsymbol{x}_1$ $\pi_2 = \boldsymbol{k}_1^\top \boldsymbol{x}_2, \pi_3 = \boldsymbol{k}_2^\top \boldsymbol{x}_3$ Output $\pi = -\pi_1 + \pi_2 + \pi_3$	$\widehat{\mathsf{Hash}}(\widehat{\mathsf{hk}}, x)$: $\widehat{\pi}_1 = (\widehat{\boldsymbol{k}}_1 \otimes \widehat{\boldsymbol{k}}_2)^\top \boldsymbol{x}_1$ $\widehat{\pi}_2 = \widehat{\boldsymbol{k}}_1^\top \boldsymbol{x}_2, \widehat{\pi}_3 = \widehat{\boldsymbol{k}}_2^\top \boldsymbol{x}_3$ Output $\widehat{\pi} = (\widehat{\pi}_1, \widehat{\pi}_2, \widehat{\pi}_3)$	$\widetilde{\mathsf{Hash}}(\widetilde{\mathsf{hk}}, (x, e, \widehat{\pi}))$: $(\gamma_{ij})_{j=1}^n = \Gamma_i(\boldsymbol{x}_i, e, \widehat{\pi}_i)$ $(i=1,2,3)$ $\widetilde{\pi}_i = \widetilde{\boldsymbol{k}}_{i0}^\top \boldsymbol{x}_i + \sum_{j=1}^n \gamma_{ij} \widetilde{\boldsymbol{k}}_{ij}^\top \boldsymbol{x}_i$ $(i=1,2,3)$ Output $\widetilde{\pi} = (\widetilde{\pi}_1, \widetilde{\pi}_2, \widetilde{\pi}_3)$
$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: $\pi_1 = w_1(s_1 \otimes s_2)$ $\pi_2 = w_2(s_1 \otimes g_2)$ $\pi_3 = w_3(g_1 \otimes s_2)$ Output $\pi = -\pi_1 + \pi_2 + \pi_3$	$\widehat{\mathsf{ProjHash}}(\widehat{\mathsf{hp}}, x, w)$: $\widehat{\pi}_1 = w_1(\widehat{s}_1 \otimes \widehat{s}_2)$ $\widehat{\pi}_2 = w_2(\widehat{s}_1 \otimes g_2)$ $\widehat{\pi}_3 = w_3(g_1 \otimes \widehat{s}_2)$ Output $\widehat{\pi} = (\widehat{\pi}_1, \widehat{\pi}_2, \widehat{\pi}_3)$	$\widetilde{\mathsf{ProjHash}}(\widetilde{\mathsf{hp}}, (x, e, \widehat{\pi}), w)$: $(\gamma_{ij})_{j=1}^n = \Gamma_i(\boldsymbol{x}_i, e, \widehat{\pi}_i)$ $(i=1,2,3)$ $\widetilde{\pi}_i = w_i \widetilde{s}_{i0} + w_i \sum_{j=1}^n \gamma_{ij} \widetilde{s}_{ij}$ $(i=1,2,3)$ Output $\widetilde{\pi} = (\widetilde{\pi}_1, \widetilde{\pi}_2, \widetilde{\pi}_3)$

Figure 3: HPSs for level-2 ciphertexts, where $\Gamma_1 \colon \mathbb{G}_T^6 \to \mathbb{Z}_p^n$, $\Gamma_2 \colon \mathbb{G}_T^4 \to \mathbb{Z}_p^n$, and $\Gamma_3 \colon \mathbb{G}_T^4 \to \mathbb{Z}_p^n$ are hash functions

2. *If Γ_1 and Γ_2 are sampled from a family of collision resistant hash functions, then $\widetilde{\boldsymbol{P}}^{(1)}$ is first-adaptive computationally universal$_2$.*

Proof. Since $\widetilde{\boldsymbol{P}}^{(1)}$ can be interpreted as the direct product of two HPSs constructed by applying generic construction based on a diverse vector space, the statement follows from Propositions 6 and 7. □

5.3.2 HPS for Level-2 Ciphertexts

Let $X = \mathbb{G}_T^4 \times \mathbb{G}_T^2 \times \mathbb{G}_T^2$ and $L = \langle \boldsymbol{h}_1 \rangle \times \langle \boldsymbol{h}_2 \rangle \times \langle \boldsymbol{h}_3 \rangle$. A witness for $x = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \in L$ is $w = (w_1, w_2, w_3) \in \mathbb{Z}_p^3$ satisfying $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = (w_1\boldsymbol{h}_1, w_2\boldsymbol{h}_2, w_3\boldsymbol{h}_3)$. Moreover, let $X' = \langle \boldsymbol{h}_1 \rangle \times \langle \boldsymbol{h}_2 \rangle \times \mathbb{G}_T^2 \subset X$, $\Pi = \mathbb{G}_T$, $\widehat{\Pi} = \widetilde{\Pi} = \mathbb{G}_T^3$. In this situation, we define three HPSs for level-2 ciphertexts $\boldsymbol{P}^{(2)} = (X, L, \Pi)$, $\widehat{\boldsymbol{P}}^{(2)} = (X, L, \widehat{\Pi})$, $\widetilde{\boldsymbol{P}}^{(2)} = (X \times \Pi \times \widehat{\Pi}, L \times \Pi \times \widehat{\Pi}, \widetilde{\Pi})$ as in Figure 3.

Proposition 11. *$\boldsymbol{P}^{(2)}$ satisfies the following:*

1. *$\boldsymbol{P}^{(2)}$ is smooth relative to X'.*

2. *Under SXDH assumption, the subset membership problem of $\boldsymbol{P}^{(2)}$ is hard relative to X' and has a trapdoor.*

3. *$X' \setminus L$ is approximately samplable relative to X.*

Proof. **Smoothness**: For $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp})$, and $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$, we have

$$\begin{aligned}
&\{(\mathsf{hp}, x, \mathsf{Hash}(\mathsf{hk}, x)) \mid x \leftarrow X' \setminus L\} \\
&= \{((s_1, s_2), (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), -(\boldsymbol{k}_1 \otimes \boldsymbol{k}_2)^\top \boldsymbol{x}_1 + \boldsymbol{k}_1^\top \boldsymbol{x}_2 + \boldsymbol{k}_2^\top \boldsymbol{x}_3) \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_3 \leftarrow \mathbb{G}_T^2 \setminus \langle \boldsymbol{h}_3 \rangle\} \\
&= \{((s_1, s_2), (w_1 \boldsymbol{h}_1, w_2 \boldsymbol{h}_2, g_1 \otimes \boldsymbol{x}_2'), -w_1(s_1 \otimes s_2) + w_2(s_1 \otimes g_2) + (g_1 \otimes \boldsymbol{k}_2^\top \boldsymbol{x}_2')) \\
&\qquad\qquad\qquad\qquad\qquad \mid w_1, w_2 \leftarrow \mathbb{Z}_p, \boldsymbol{x}_2' \leftarrow \mathbb{G}_2^2 \setminus \langle \boldsymbol{g}_2 \rangle\} \\
&= \{((s_1, s_2), (w_1 \boldsymbol{h}_1, w_2 \boldsymbol{h}_2, g_1 \otimes \boldsymbol{x}_2'), -w_1(s_1 \otimes s_2) + w_2(s_1 \otimes g_2) + (g_1 \otimes \pi_2')) \\
&\qquad\qquad\qquad\qquad\qquad \mid w_1, w_2 \leftarrow \mathbb{Z}_p, \boldsymbol{x}_2' \leftarrow \mathbb{G}_2^2 \setminus \langle \boldsymbol{g}_2 \rangle, \pi_2' \leftarrow \mathbb{G}_2\} \\
&= \{((s_1, s_2), (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3), \pi) \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_3 \leftarrow \mathbb{G}_T^2 \setminus \langle \boldsymbol{h}_3 \rangle, \pi \leftarrow \mathbb{G}_T\} \\
&= \{(\mathsf{hp}, x, \pi) \mid x \leftarrow X' \setminus L, \pi \leftarrow \Pi\}
\end{aligned}$$

and hence the smoothness holds. Here, at the third equality, we utilized the fact that the conditional distribution of $\boldsymbol{k}_2^\top \boldsymbol{x}_2'$ conditioned on a given $s_2 = \boldsymbol{k}_2^\top \boldsymbol{g}_2$ is uniformly random over $\mathbb{G}_2$, which follows from the linear independence of $\boldsymbol{g}_2$ and $\boldsymbol{x}_2'$.

Hardness of the Subset Membership Problem: For $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, we have

$$\begin{aligned}
&\{(\mathsf{pp}, x) \mid x \leftarrow L\} \\
&= \{\mathsf{pp}, (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_3 \leftarrow \langle \boldsymbol{h}_3 \rangle\} \\
&= \{\mathsf{pp}, (\boldsymbol{x}_1, \boldsymbol{x}_2, g_1 \otimes \boldsymbol{x}_2') \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_2' \leftarrow \langle \boldsymbol{g}_2 \rangle\} \\
&\overset{c}{\approx} \{\mathsf{pp}, (\boldsymbol{x}_1, \boldsymbol{x}_2, g_1 \otimes \boldsymbol{x}_2') \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_2' \leftarrow \mathbb{G}_2^2 \setminus \langle \boldsymbol{g}_2 \rangle\} \\
&= \{\mathsf{pp}, (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \mid \boldsymbol{x}_1 \leftarrow \langle \boldsymbol{h}_1 \rangle, \boldsymbol{x}_2 \leftarrow \langle \boldsymbol{h}_2 \rangle, \boldsymbol{x}_3 \leftarrow \mathbb{G}_T^2 \setminus \langle \boldsymbol{h}_3 \rangle\} \\
&= \{(\mathsf{pp}, x) \mid x \leftarrow X' \setminus L\}
\end{aligned}$$

and hence the hardness of the subset membership problem holds. Here, for the relation $\overset{c}{\approx}$ above, we used the SXDH assumption (or more specifically, DDH assumption on $\mathbb{G}_2$).

Trapdoor of the Subset Membership Problem: We can show that the subset membership problem has a trapdoor, by defining two algorithms in Definition 16 in the following manner:

- $\mathsf{TrapdoorSetUp}(1^\ell)$ chooses $\boldsymbol{g}_1, \boldsymbol{g}_2$ in the SetUp by computing

$$\boldsymbol{g}_1 = (g_1, \tau_1 \cdot g_1),\ \boldsymbol{g}_2 = (g_2, \tau_2 \cdot g_2) \text{ where } \tau_1, \tau_2 \leftarrow \mathbb{Z}_p$$

  and outputs $\tau = (\tau_1, \tau_2)$ as a trapdoor.

- $\mathsf{Distinguish}(x, \tau)$ takes $x = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \in X$ and $\tau = (\tau_1, \tau_2)$ as inputs. For $\boldsymbol{x}_1 = (x_{11}, x_{12}, x_{13}, x_{14})$, $\boldsymbol{x}_2 = (x_{21}, x_{22})$, and $\boldsymbol{x}_3 = (x_{31}, x_{32})$, if all of

$$x_{12} = \tau_2 \cdot x_{11},\ x_{13} = \tau_1 \cdot x_{11},\ x_{14} = \tau_1 \tau_2 \cdot x_{11},\ x_{22} = \tau_1 \cdot x_{21},\ x_{32} = \tau_2 \cdot x_{31}$$

  hold, then the algorithm decides that $\boldsymbol{x} \in L$; and otherwise decides that $\boldsymbol{x} \notin L$.

Approximate Samplability: The approximate samplability immediately follows from the fact that the uniform distribution over $X' \setminus L$ and the uniform distribution over X' are statistically indistinguishable, and the fact that the elements of X' are efficiently samplable. □

Proposition 12. *$\widetilde{\boldsymbol{P}}^{(2)}$ satisfies the following:*

1. *If Γ_1, Γ_2, and Γ_3 are injective, then $\widetilde{\boldsymbol{P}}^{(2)}$ is information-theoretically universal$_2$.*
2. *If Γ_1, Γ_2, and Γ_3 are sampled from a family of collision resistant hash functions, then $\widetilde{\boldsymbol{P}}^{(2)}$ is first-adaptive computationally universal$_2$.*

Proof. $\widetilde{\boldsymbol{P}}^{(2)}$ can be represented as the direct product of three HPSs based on diverse vector spaces (except that the finite set E are common in all three HPSs). Hence the universal$_2$ property can be shown in the same manner as Propositions 6 and 7. □

Proposition 13. $\widehat{\boldsymbol{P}}^{(2)}$ *is* $((2p-1)/p^2)$*-universal*$_1$.

Proof. Let $\widehat{\mathsf{hp}} = (\widehat{s}_1, \widehat{s}_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, $x = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \in X \setminus L$, and $\widehat{\pi} = (\widehat{\pi}_1, \widehat{\pi}_2, \widehat{\pi}_3) \in \mathbb{G}_T^3$ be chosen randomly. The goal of the proof is to show that

$$\Pr_{\widehat{\mathsf{hk}} \leftarrow \widehat{K}}[\widehat{\mathsf{Hash}}(x) = \widehat{\pi} \mid \widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}}) = \widehat{\mathsf{hp}}]$$

is at most $(2p-1)/p^2$. Here, x can be represented as

$$\begin{aligned}
\boldsymbol{x}_1 &= w_1(\boldsymbol{g}_1 \otimes \boldsymbol{g}_2) + w_1'(\boldsymbol{g}_1 \otimes \boldsymbol{g}_2') + w_1''(\boldsymbol{g}_1' \otimes \boldsymbol{g}_2) + w_1'''(\boldsymbol{g}_1' \otimes \boldsymbol{g}_2') \ , \\
\boldsymbol{x}_2 &= w_2(\boldsymbol{g}_1 \otimes g_2) + w_2'(\boldsymbol{g}_1' \otimes g_2) \ , \\
\boldsymbol{x}_3 &= w_3(g_1 \otimes \boldsymbol{g}_2) + w_3'(g_1 \otimes \boldsymbol{g}_2') \ ,
\end{aligned}$$

where elements $\boldsymbol{g}_1'$ and $\boldsymbol{g}_2'$ are linearly independent from $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, respectively. We note that at least one of w_1', w_1'', w_1''', w_2', and w_3' is non-zero, since $x \in X \setminus L$.

Under the condition that $\widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}}) = \widehat{\mathsf{hp}}$, i.e., $(\widehat{\boldsymbol{k}}_1^\top \boldsymbol{g}_1, \widehat{\boldsymbol{k}}_2^\top \boldsymbol{g}_2) = (\widehat{s}_1, \widehat{s}_2)$, the condition $\widehat{\mathsf{Hash}}(x) = \widehat{\pi}$ is equivalent to

$$\begin{aligned}
& w_1(\widehat{s}_1 \otimes \widehat{s}_2) + w_1'(\widehat{s}_1 \otimes \widehat{\boldsymbol{k}}_2^\top \boldsymbol{g}_2') + w_1''(\widehat{\boldsymbol{k}}_1^\top \boldsymbol{g}_1' \otimes \widehat{s}_2) + w_1'''(\widehat{\boldsymbol{k}}_1^\top \boldsymbol{g}_1' \otimes \widehat{\boldsymbol{k}}_2^\top \boldsymbol{g}_2') = \widehat{\pi}_1 \ , \\
& w_2(\widehat{s}_1 \otimes g_2) + w_2'(\widehat{\boldsymbol{k}}_1^\top \boldsymbol{g}_1' \otimes g_2) = \widehat{\pi}_2 \ , \\
& w_3(g_1 \otimes \widehat{s}_2) + w_3'(g_1 \otimes \widehat{\boldsymbol{k}}_2^\top \boldsymbol{g}_2') = \widehat{\pi}_3 \ .
\end{aligned}$$

Moreover, since $\boldsymbol{g}_1', \boldsymbol{g}_2'$ are linearly independent from $\boldsymbol{g}_1, \boldsymbol{g}_2$, respectively, $\widehat{\boldsymbol{k}}_1^\top \boldsymbol{g}_1'$ and $\widehat{\boldsymbol{k}}_2^\top \boldsymbol{g}_2'$ take uniformly random values over $\mathbb{G}_1$ and $\mathbb{G}_2$ independently from $\widehat{s}_1$ and $\widehat{s}_2$, respectively, assuming $\widehat{\boldsymbol{k}}_1, \widehat{\boldsymbol{k}}_2 \leftarrow \mathbb{Z}_p^2$. If we write these values as π_1', π_2', the conditions above can be rewritten as

$$w_1(\widehat{s}_1 \otimes \widehat{s}_2) + w_1'(\widehat{s}_1 \otimes \pi_2') + w_1''(\pi_1' \otimes \widehat{s}_2) + w_1'''(\pi_1' \otimes \pi_2') = \widehat{\pi}_1 \ , \tag{1}$$

$$w_2(\widehat{s}_1 \otimes g_2) + w_2'(\pi_1' \otimes g_2) = \widehat{\pi}_2 \ , \tag{2}$$

$$w_3(g_1 \otimes \widehat{s}_2) + w_3'(g_1 \otimes \pi_2') = \widehat{\pi}_3 \ . \tag{3}$$

From now on, we discuss the probabilities for these conditions to hold.

First, let us consider the case with $w_2' \neq 0$. In this case, since π_1' is uniformly random, the whole of the left-hand side of Equation (2) takes a uniformly random value. Therefore, the probability that Equation (2) holds is $1/p$. Thus the probability that the whole condition above holds is at most $1/p$, and from $1/p \leq (2p-1)/p^2$, the statement follows. For the case with $w_3' \neq 0$, we can show the statement by applying the same discussion to Equation (3). We discuss the other case $w_2' = w_3' = 0$ from now.

If $w_1''' = 0$, then either w_1' or w_1'' is non-zero. Now we also have that the left-hand side of Equation (1) takes a uniformly random value, therefore the probability for the above condition to hold is at most $1/p$. Finally, we consider the remaining case with $w_1''' \neq 0$. In this case, Equation (1) can be rewritten as

$$(\pi_1' + v_1'\widehat{s}_1) \otimes (\pi_2' + v_1''\widehat{s}_2) = \widehat{\pi}_1' \ ,$$

where $v_1' := w_1' w_1'''^{-1}$, $v_1'' := w_1'' w_1'''^{-1}$, and $\widehat{\pi}_1' := w_1'''^{-1}(\widehat{\pi}_1 - w_1(\widehat{s}_1 \otimes \widehat{s}_2)) + v_1' v_1''(\widehat{s}_1 \otimes \widehat{s}_2)$. Furthermore, as π_1', π_2' are uniformly random, $\pi_1' + v_1'\widehat{s}_1$ and $\pi_2' + v_1'\widehat{s}_2$ are also uniformly random; we denote them as π_1'' and π_2''. Under these notations, the given condition is equivalent to

$$\pi_1'' \otimes \pi_2'' = \widehat{\pi}_1' \ .$$

This condition holds with probability at most $1/p$ when $\widehat{\pi}_1' \neq 0$, and with probability $(2p-1)/p^2$ when $\widehat{\pi}_1' = 0$. Hence the statement also holds in this case. □

```
$\mathsf{KeyGen}(1^\ell)$:
    $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$
    $\widehat{\mathsf{hk}} \leftarrow \widehat{\mathsf{HashKG}}(\mathsf{pp}), \widehat{\mathsf{hp}} \leftarrow \widehat{\mathsf{ProjKG}}(\widehat{\mathsf{hk}})$
    $\widetilde{\mathsf{hk}}^{(i)} \leftarrow \widetilde{\mathsf{HashKG}}^{(i)}(\mathsf{pp})\ (i = 1, 2)$
    $\widetilde{\mathsf{hp}}^{(i)} \leftarrow \widetilde{\mathsf{ProjKG}}^{(i)}(\widetilde{\mathsf{hk}}^{(i)})\ (i = 1, 2)$
    $\mathsf{pk} = (\mathsf{hp}, \widehat{\mathsf{hp}}, \widetilde{\mathsf{hp}}^{(1)}, \widetilde{\mathsf{hp}}^{(2)})$
    $\mathsf{sk} = (\mathsf{hk}, \widehat{\mathsf{hk}}, \widetilde{\mathsf{hk}}^{(1)}, \widetilde{\mathsf{hk}}^{(2)})$, $\mathsf{ek} = (\widetilde{\mathsf{hk}}^{(1)}, \widetilde{\mathsf{hk}}^{(2)})$
    Output $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$

$\mathsf{Enc}(\mathsf{pk}, (i, m))$:
    Sample $x \in L^{(i)}$ together with its witness $w$
    $\pi \leftarrow \mathsf{ProjHash}^{(i)}(x, w)$
    When $i = 1$, $e = (mg_1, mg_2) + \pi$
    When $i = 2$, $e = mg_T + \pi$
    $\widehat{\pi} \leftarrow \widehat{\mathsf{ProjHash}}^{(i)}(x, w)$
    $\widetilde{\pi} \leftarrow \widetilde{\mathsf{ProjHash}}^{(i)}((x, e, \widehat{\pi}), w)$
    Output $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$

$\mathsf{Dec}(\mathsf{sk}, C)$: $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$
    If $\widehat{\pi} \neq \widehat{\mathsf{Hash}}^{(i)}(x)$, output $\perp$
    If $\widetilde{\pi} \neq \widetilde{\mathsf{Hash}}^{(i)}(x, e, \widehat{\pi})$, output $\perp$
    $\pi \leftarrow \mathsf{Hash}^{(i)}(x)$
    When $i = 1$, output $m = (e_1 - \pi_1)/g_1$
    When $i = 2$, output $m = (e - \pi)/g_T$

$\mathsf{Eval}(\mathsf{ek}, \mathsf{Add}^{(i^*)}, C, C')$:
    $C = (i, x, e, \widehat{\pi}, \widetilde{\pi}), C' = (i', x', e', \widehat{\pi}', \widetilde{\pi}')$
    If not $i = i' = i^*$, output $\perp$
    If $\widetilde{\pi} \neq \widetilde{\mathsf{Hash}}^{(i)}(x, e, \widehat{\pi})$, output $\perp$
    If $\widetilde{\pi}' \neq \widetilde{\mathsf{Hash}}^{(i)}(x', e', \widehat{\pi}')$, output $\perp$
    Sample $x_0 \in L^{(i)}$ together with its witness $w_0$
    $x'' = x + x' + x_0$, $e'' = e + e' + \mathsf{ProjHash}^{(i)}(x_0, w_0)$
    $\widehat{\pi}'' = \widehat{\pi} + \widehat{\pi}' + \widehat{\mathsf{ProjHash}}^{(i)}(x_0, w_0)$
    $\widetilde{\pi}'' \leftarrow \widetilde{\mathsf{Hash}}^{(i)}(x'', e'', \widehat{\pi}'')$
    Output $C'' = (i, x'', e'', \widehat{\pi}'', \widetilde{\pi}'')$
```

Figure 4: Proposed Keyed-2LHE scheme (except homomorphic multiplication)

5.4 Concrete Construction of the Proposed Scheme

We describe the concrete construction of our proposed scheme in this section. As explained in Section 5.2, our scheme (except the multiplication between ciphertexts) mostly follows the generic construction of Section 4.2. Our scheme except the multiplication is given in Figure 4.

Next, we give an explanation for the multiplication algorithm. When Eval takes ek, $f = \mathsf{Mult}$, and level-1 ciphertexts $C = (1, x, e, \widehat{\pi}, \widetilde{\pi})$, $C' = (1, x', e', \widehat{\pi}', \widetilde{\pi}')$ as inputs, it computes the following:

1. If $\widetilde{\pi} \neq \widetilde{\mathsf{Hash}}^{(1)}(x, e, \widehat{\pi})$ or $\widetilde{\pi}' \neq \widetilde{\mathsf{Hash}}^{(1)}(x', e', \widehat{\pi}')$, output $\perp$.
2. Sample $x_0 \in L^{(2)}$ together with its witness w_0.
3. Set $x'' = (\boldsymbol{x}_1 \otimes \boldsymbol{x}'_2, \boldsymbol{x}_1 \otimes e'_2, e_1 \otimes \boldsymbol{x}'_2) + x_0$.
4. Set $e'' \leftarrow e_1 \otimes e'_2 + \mathsf{ProjHash}^{(2)}(x_0, w_0)$.
5. Set $\widehat{\pi}'' \leftarrow (\widehat{\pi}_1 \otimes \widehat{\pi}'_2, \widehat{\pi}_1 \otimes e'_2, e_1 \otimes \widehat{\pi}'_2) + \widehat{\mathsf{ProjHash}}^{(2)}(x_0, w_0)$.
6. Set $\widetilde{\pi}'' \leftarrow \widetilde{\mathsf{Hash}}^{(2)}(x'', e'', \widehat{\pi}'')$.
7. Output $(2, x'', e'', \widehat{\pi}'', \widetilde{\pi}'')$.

Let us confirm the correctness of the multiplication. Let C and C' be level-1 ciphertexts of plaintexts m and m', respectively. The operation of adding x_0 or its hash values does not affect correctness (since it is just for rerandomization), therefore we ignore it here. In Steps 3 and 4 of Mult, the second and the third components of x'' and the value of e'' can be represented as

$$\begin{aligned} \boldsymbol{x}_1 \otimes e_2' &= \boldsymbol{x}_1 \otimes (m' g_2 + \boldsymbol{k}_2^\top \boldsymbol{x}_2') \ , \\ e_1 \otimes \boldsymbol{x}_2' &= (m g_1 + \boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes \boldsymbol{x}_2' \ , \\ e_1 \otimes e_2' &= (m g_1 + \boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes (m' g_2 + \boldsymbol{k}_2^\top \boldsymbol{x}_2') \ . \end{aligned}$$

Hence, when we apply Dec to the evaluated ciphertext, we indeed obtain

$$\begin{aligned} &(e'' - \mathsf{Hash}^{(2)}(\boldsymbol{x}_1 \otimes \boldsymbol{x}_2', \boldsymbol{x}_1 \otimes e_2', e_1 \otimes \boldsymbol{x}_2'))/g_T \\ &= (e_1 \otimes e_2' + (\boldsymbol{k}_1 \otimes \boldsymbol{k}_2)^\top (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2') - \boldsymbol{k}_1^\top (\boldsymbol{x}_1 \otimes e_2') - \boldsymbol{k}_2^\top (e_1 \otimes \boldsymbol{x}_2'))/g_T \\ &= ((m g_1 + \boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes (m' g_2 + \boldsymbol{k}_2^\top \boldsymbol{x}_2') + (\boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes (\boldsymbol{k}_2^\top \boldsymbol{x}_2') \\ &\qquad - (\boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes (m' g_2 + \boldsymbol{k}_2^\top \boldsymbol{x}_2') - (m g_1 + \boldsymbol{k}_1^\top \boldsymbol{x}_1) \otimes (\boldsymbol{k}_2^\top \boldsymbol{x}_2'))/g_T \\ &= ((m g_1) \otimes (m' g_2))/g_T = m m' \ . \end{aligned}$$

The remaining task is to see that $\widehat{\pi}''$ is a correct hash value, and this can be checked from

$$\widehat{\pi}_1 \otimes \widehat{\pi}_2' = (\widehat{\boldsymbol{k}}_1^\top \boldsymbol{x}_1) \otimes (\widehat{\boldsymbol{k}}_2^\top \boldsymbol{x}_2') = (\widehat{\boldsymbol{k}}_1 \otimes \widehat{\boldsymbol{k}}_2)^\top (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2') \ ,$$

$$\begin{aligned} \widehat{\pi}_1 \otimes e_2' &= (\widehat{\boldsymbol{k}}_1^\top \boldsymbol{x}_1) \otimes e_2' = \widehat{\boldsymbol{k}}_1^\top (\boldsymbol{x}_1 \otimes e_2') \ , \\ e_1 \otimes \widehat{\pi}_2' &= e_1 \otimes (\widehat{\boldsymbol{k}}_2^\top \boldsymbol{x}_2') = \widehat{\boldsymbol{k}}_2^\top (e_1 \otimes \boldsymbol{x}_2') \ . \end{aligned}$$

Summarizing, the correctness of the multiplication holds.

We have the following theorem for the security of our proposed scheme. As it can be proved in almost the same manner as the original security proof of the KH-PKE scheme in [12], we describe the detailed proof of the theorem in Appendix A.

Theorem 2. *The Keyed-2LHE scheme constructed above is KH-CCA secure under SXDH assumption.*

6 Efficiency Evaluations

In this section, we evaluate the efficiency of our proposed scheme. All the evaluations given in this section are based on the case where hash functions Γ in the construction of $\widetilde{\boldsymbol{P}}^{(1)}$ and $\widetilde{\boldsymbol{P}}^{(2)}$ are collision resistant (i.e., the case where $\widetilde{\boldsymbol{P}}$ is first-adaptive computationally universal$_2$), and the case where $n = 1$ in the construction of $\widetilde{\boldsymbol{P}}^{(1)}$ and $\widetilde{\boldsymbol{P}}^{(2)}$ (similarly to the DDH-based instantiation of KH-PKE in [12]). Also, we assume that all the necessary values of pairings between secret keys, public keys, or public parameters are computed in advance during the key generation, and the resulting values are involved in secret keys or public keys.

6.1 Size of Keys and Ciphertexts

A comparison of key and ciphertext sizes among our scheme, the state-of-the-art 2LHE scheme [3], and the original KH-PKE scheme [12] is shown in Table 1. The four numbers in each cell denote the numbers of elements in $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. Here the KH-PKE scheme means its instantiation based on DDH assumption shown in Section 5.3 of [12]. It is natural that our scheme is less efficient than the other two schemes as shown in the table, since our scheme achieves stronger security than [3] and realizes stronger functionality than [12]. We expect that the overhead of our scheme compared to the other two schemes is within an acceptable range from practical viewpoints.

Table 1: Comparison of key and ciphertext sizes (numbers of elements in $\mathbb{Z}_p$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ in this order)

	Ours	2LHE [3]	KH-PKE [12]
sk	32, 0, 0, 0	2, 0, 0, 0	8, 0, 0, 0
pk	0, 4, 4, 9	0, 2, 2, 4	0, 4, -, -
ek	0, 2, 2, 6	—	0, 2, -, -
Level-1 Ciphertext	0, 5, 5, 0	0, 2, 2, 0	0, 5, -, -
Level-2 Ciphertext	0, 0, 0, 15	0, 0, 0, 4	—

Table 2: Numbers of operations in our proposed scheme; here "Op" and "Exp" denote addition and scalar multiplication, respectively, over $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ in this order (note that now $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are regarded as additive groups), "P" denotes pairing, and "DL" denotes computation of discrete logarithm (with restricted exponent)

		Op	Exp	P	DL
KeyGen		4, 4, 15	8, 8, 24	0	0
Enc	Level-1	2, 2, 0	7, 7, 0	0	0
	Level-2	0, 0, 6	0, 0, 21	0	0
Dec	Level-1	4, 2, 0	6, 4, 0	0	1
	Level-2	0, 0, 18	0, 0, 24	0	1
Eval	$\mathsf{Add}^{(1)}$	11, 11, 0	10, 10, 0	0	0
	$\mathsf{Add}^{(2)}$	0, 0, 41	0, 0, 38	0	0
	Mult	2, 2, 19	4, 4, 22	12	0

6.2 Computational Costs

Table 2 shows the numbers of addition and scalar multiplication (note that now the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are regarded as additive groups), pairing, and computation of discrete logarithm (with restricted exponent; see Section 6.3 for the details) performed in each algorithm for our proposed scheme. Here the three numbers in each cell for "Op" and "Exp" are those for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Note that operations on $\mathbb{Z}_p$ are omitted here, since they are much faster compared to the operations over the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$.

Table 3 shows a comparison of roughly estimated computation times for each algorithms in our proposed scheme with the 2LHE scheme [3] and the KH-PKE scheme [12]. In the time estimate, we started with the benchmark result in [3] using the Barreto–Naehrig (BN) curves over 462-bit prime field. Then by comparing the numbers of operations (such as ones in Table 2) among the three schemes, we calculated the running times estimated from the result of [3]. Although the estimated times are deduced in such a rough manner (for example, it does not include computation times for hash functions Γ in $\widetilde{\boldsymbol{P}}$, discrete logarithms for decryption, and operations on $\mathbb{Z}_p$; and it does not take into consideration that the underlying group for the KH-PKE scheme in [12] can be more efficient than the other two schemes as it does not require pairings), it still suggests that our proposed scheme would be feasible in practical implementation. On the other hand, some algorithms in our scheme related to the group $\mathbb{G}_T$ (namely, Dec for level-2 ciphertexts, $\mathsf{Add}^{(2)}$, and Mult) are particularly less efficient than the others, which is a major point to be improved in future research.

6.3 Remarks

In Dec algorithm of our proposed scheme, we need to compute division by group element g_1 or g_T, which corresponds to computing the discrete logarithm when the groups are written multiplicatively. Therefore, to execute decryption efficiently, restriction of the plaintext space for input ciphertexts is essential (which also was the case with the previous schemes, i.e., 2LHE in [3] and DDH-based KH-PKE in [12]). For the concrete method regarding the discrete logarithm computation, we refer to the previous paper [3].

On the other hand, it is worth noting that our scheme has the following feature; if a party has only the

Table 3: Comparison of roughly estimated computation times (ms)

		Ours	2LHE [3]	KH-PKE [12]
KeyGen		7.968	0.983	0.378
Enc	Level-1	1.346	0.577	0.330
	Level-2	5.578	1.326	—
Dec	Level-1	5.950	0.416	2.494
	Level-2	30.343	3.797	—
Eval	$\mathsf{Add}^{(1)}$	8.481	0.402	2.688
	$\mathsf{Add}^{(2)}$	34.170	1.107	—
	Mult	48.163	10.799	—

part $\widetilde{\mathsf{hk}}^{(1)}$ of the evaluation key $\mathsf{ek} = (\widetilde{\mathsf{hk}}^{(1)}, \widetilde{\mathsf{hk}}^{(2)})$, then the party can only compute addition between level-1 ciphertexts but not addition between level-2 ciphertexts nor multiplication. This feature to only allow some subclass of operations by giving the evaluation key partially is a new feature that previous schemes did not possess, though we are leaving the formal definition for security with such partial exposure of the evaluation key as a future research topic.

Acknowledgements

This research was partially supported by the Ministry of Internal Affairs and Communications SCOPE Grant Number 182103105, JST CREST Grant Number JPMJCR19F6, and JSPS KAKENHI Grant Number 19H01109.

References

[1] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 69–100. Springer, 2015.

[2] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[3] Nuttapong Attrapadung, Goichiro Hanaoka, Shigeo Mitsunari, Yusuke Sakai, Kana Shimizu, and Tadanori Teruya. Efficient two-level homomorphic encryption in prime-order bilinear groups and A fast implementation in WebAssembly. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 685–697. ACM, 2018.

[4] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475. Springer, 2013.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory*, 6(3):13:1–13:36, 2014.

[7] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[8] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1518–1529. ACM, 2015.

[9] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2013.

[10] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[11] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[12] Keita Emura, Goichiro Hanaoka, Koji Nuida, Go Ohtake, Takahiro Matsuda, and Shota Yamada. Chosen ciphertext secure keyed-homomorphic public-key cryptosystems. *Des. Codes Cryptogr.*, 86(8):1623–1683, 2018.

[13] Keita Emura, Goichiro Hanaoka, Go Ohtake, Takahiro Matsuda, and Shota Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 32–50. Springer, 2013.

[14] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 44–61. Springer, 2010.

[15] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to UC-PAKE and more. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 630–655. Springer, 2015.

[16] Junzuo Lai, Robert H. Deng, Changshe Ma, Kouichi Sakurai, and Jian Weng. CCA-secure keyed-fully homomorphic encryption. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 70–98. Springer, 2016.

[17] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2014.

[18] Manoj Prabhakaran and Mike Rosulek. Homomorphic encryption with CCA security. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 667–678. Springer, 2008.

[19] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[20] Hoeteck Wee. KDM-security via homomorphic smooth projective hashing. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 159–179. Springer, 2016.

A Proof of Theorem 2

In this appendix, we give a proof of Theorem 2. Before starting the proof, let us define some terminology to make the argument simpler. For a ciphertext $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$, we say:

- C is *i-regular*, if $x \in L^{(i)}$.
- C is $\widetilde{\boldsymbol{P}}^{(i)}$*-consistent*, if $\widetilde{\pi} = \widetilde{\mathsf{Hash}}^{(i)}(x, e, \widehat{\pi})$ holds. (Similarly for $\widehat{\boldsymbol{P}}^{(i)}$.)
- C is $\widetilde{\boldsymbol{P}}^{(i)}$*-forging*, if it is $\widetilde{\boldsymbol{P}}^{(i)}$-consistent but not i-regular. (Similarly for $\widehat{\boldsymbol{P}}^{(i)}$.)

To deal with both levels of ciphertexts at the same time, we just say "C is regular", "C is $\widetilde{\boldsymbol{P}}$-consistent", and "$C$ is $\widetilde{\boldsymbol{P}}$-forging", omitting the level of the ciphertext.

Let us prove an important lemma in the security proof:

Lemma 2 (source ciphertext hiding property)**.** *Let $i \in \{1, 2\}$. Assume that level-i ciphertexts $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$ and $C' = (i, x', e', \widehat{\pi}', \widetilde{\pi}')$ satisfy the following:*

(*) *C and C' are $\widehat{\boldsymbol{P}}^{(i)}$-consistent and $\widetilde{\boldsymbol{P}}^{(i)}$-consistent ciphertexts for the same plaintext, and the distributions of C and C' are identical and independent with each other.*

Moreover, let $C'' = (i, x'', e'', \widehat{\pi}'', \widetilde{\pi}'')$ be a $\widehat{\boldsymbol{P}}^{(i)}$-consistent and $\widetilde{\boldsymbol{P}}^{(i)}$-consistent level-$i$ ciphertext. Under these assumptions, for any operation $f \in \{\mathsf{Add}^{(1)}, \mathsf{Add}^{(2)}, \mathsf{Mult}\}$, the outputs of $\mathsf{Eval}(f, C, C'')$ and $\mathsf{Eval}(f, C', C'')$ satisfy the condition $(*)$ (unless the evaluation is not rejected).

Proof. The $\widehat{\boldsymbol{P}}$-consistency and the fact that the outputs of the Eval are ciphertexts for the same plaintext follow immediately from the correctness of Eval. Also, since $\widetilde{\pi}$ in the output of Eval is calculated using ek, $\widetilde{\boldsymbol{P}}$-consistency holds. Therefore, it suffices to prove that the two outputs of Eval in the statement follow identical and independent distributions.

First, let us consider the case with $i = 1$ and $f = \mathsf{Add}^{(1)}$. In this case, the second component of $\mathsf{Eval}(f, C, C'')$ and $\mathsf{Eval}(f, C', C'')$ can be represented as $x + x'' + r$ and $x' + x'' + r'$, respectively, where $r, r' \leftarrow L^{(1)}$ are the random values sampled for the rerandomization. From the assumption in the statement, x and x' follow identical and independent distributions. Hence, the results of the operations follow identical and independent distributions as well. The same argument holds for the case with $i = 2$ and $f = \mathsf{Add}^{(2)}$.

Finally, let us consider the case with $i = 1$ and $f = \mathsf{Mult}$. Let $x = (\boldsymbol{x}_1, \boldsymbol{x}_2)$, $e = (e_1, e_2)$, $x' = (\boldsymbol{x}'_1, \boldsymbol{x}'_2)$, and $e' = (e'_1, e'_2)$. In this case, the second component of $\mathsf{Eval}(f, C, C'')$ and $\mathsf{Eval}(f, C', C'')$ can be represented as

$$(\boldsymbol{x}_1 \otimes \boldsymbol{x}''_2 + \boldsymbol{r}_1, \boldsymbol{x}_1 \otimes e''_2 + \boldsymbol{r}_2, e_1 \otimes \boldsymbol{x}''_2 + \boldsymbol{r}_3) \ ,$$
$$(\boldsymbol{x}'_1 \otimes \boldsymbol{x}''_2 + \boldsymbol{r}'_1, \boldsymbol{x}'_1 \otimes e''_2 + \boldsymbol{r}'_2, e'_1 \otimes \boldsymbol{x}''_2 + \boldsymbol{r}'_3) \ ,$$

respectively, where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$, $\boldsymbol{r}'_1$, $\boldsymbol{r}_2$, and $\boldsymbol{r}'_3$ are random values sampled from $L^{(2)}$ for the rerandomization. From the assumption of the statement, x and x' follow identical and independent distributions. Hence the two values above follow identical and independent distributions. Furthermore, since C and C' have the same plaintext, e and e' follow identical and independent distributions. Hence, the results of the operations follow identical and independent distributions as well.

Combining the discussions for all the cases, the statement holds. □

From now on, we prove Theorem 2 by game-hopping. The idea for the game-hopping is almost identical to the security proof in [12]. Note that since we only deal with prime-order groups, some part of the proof can be simplified compared to [12]; in detail, a critical integer (in the sense of Definition 7 in [12]) does not exist in our case. Let T_i be the event that the adversary wins the game i.

A.1 Preliminary Part

We start the security proof with a preliminary game-hopping.

Game pre-0 This is the original KH-CCA game.

Game pre-1 In this game, we prepare another list List' at the beginning of the guess stage. This list will record the history of evaluations. When the List consists of $\kappa+1$ elements $\mathsf{List} = (C_0, C_1, \ldots, C_\kappa)$ $(C_0 = C^*)$, List' will have κ elements $((D_1, D'_1, f_1), \ldots, (D_\kappa, D'_\kappa, f_\kappa))$. Here, each of D_i and D'_i is either a ciphertext or an index in $\{0, \ldots, i-1\}$, and f_i is one of the operations $\mathsf{Add}^{(1)}$, $\mathsf{Add}^{(2)}$, and Mult. The tuple (D_i, D'_i, f_i) in the List' means that the ciphertext C_i was obtained by applying the operation f_i to the ciphertexts D_i and D'_i (where, if D_i or D'_i is an index $j \in \{0, \ldots, i-1\}$, then it represents the ciphertext C_j in List).

In the situation above, let us assume that ciphertexts C, C' and an operation f are queried to Eval oracle. If this query is not rejected and if at least one of C and C' is already in the List, we modify the behavior of the Eval oracle in the following manner. First, we discuss the case with $C \in \mathsf{List}$ and $C' \notin \mathsf{List}$. In this case, Eval oracle works as follows assuming that i is the smallest index satisfying $C_i = C$.

1. Instead of the challenge ciphertext C^*, Eval oracle computes a new ciphertext $\overline{C^*} = \mathsf{Enc}(i^*, m^*_b)$. Note that $\overline{C^*}$ has the same plaintext as the actual challenge ciphertext C^*. We call this new ciphertext a **source ciphertext**.

2. Eval oracle re-calculates all elements of List according to the information recorded in List', using source ciphertext $\overline{C^*}$ instead of challenge ciphertext C^*. Let us denote re-calculated elements of List as $\overline{C}_0, \ldots, \overline{C}_\kappa$ $(\overline{C}_0 = \overline{C^*})$.

3. Since $C = C_i \in \mathsf{List}$, C is also re-calculated in the previous step. Eval oracle evaluates f between $\overline{C}_i$ (instead of C) and C', and obtains the result C''.

4. Eval oracle appends $C_{\kappa+1} := C''$ to the List and $(D_{\kappa+1}, D'_{\kappa+1}, f_{\kappa+1}) := (i, C', f)$ to the List', and outputs C'' as the response to Eval query.

For the case with $C \notin \mathsf{List}$ and $C' \in \mathsf{List}$, Eval oracle executes the same procedure where the role of C and C' are exchanged. When $C, C' \in \mathsf{List}$, Eval oracle executes the above procedure where both C and C' are re-calculated during the procedure (and both $D_{\kappa+1}$ and $D'_{\kappa+1}$ will be indices). We refer to the above process as the $(\kappa+1)$-th **refreshing process**, and the corresponding query as the $(\kappa+1)$-th **refreshing query**.

Game pre-2 In this game, the challenger computes the hash values for $\boldsymbol{P}, \widehat{\boldsymbol{P}}, \widetilde{\boldsymbol{P}}$ using Hash instead of $\mathsf{ProjHash}$. Note that this modification is possible due to the fact that the challenger owns the secret key sk.

We evaluate the differences between the winning probability of the adversary for each game.

In Game pre-1, since the challenge ciphertext C^* and the source ciphertext $\overline{C^*}$ have the same plaintext, their distributions are identical, while being independent. Hence, by applying the source ciphertext hiding property recursively, it can be shown that the output distribution of the refreshing query does not differ from that of Game pre-0. Moreover, since the output distributions of Hash and $\mathsf{ProjHash}$ are identical, the challenge ciphertext computed in Game pre-2 has exactly the same distribution as that of Game pre-1.

Therefore, the winning probability of the adversary does not change in the preliminary game-hopping, that is, we have

$$\Pr[T_{\text{pre-0}}] = \Pr[T_{\text{pre-1}}] = \Pr[T_{\text{pre-2}}] \enspace .$$

A.2 Main Part

We move to the main part of the game-hopping. Let Q_{ref} denote the number of refreshing queries that the adversary asks. Note that $Q_{\mathsf{ref}} = O(\mathsf{poly}(\ell))$, since the adversary is a PPT algorithm. The outline of the game-hopping is given as follows:

Game 0 This game is identical to the Game pre-2.

Game 1 In this game, the third element $\overline{e^*}$ of the source ciphertext $\overline{C^*}$ is replaced with a uniformly random element of $\Pi^{(i^*)}$, in the first refreshing query.

Game 2 In this game, the third element $\overline{e^*}$ of the source ciphertext $\overline{C^*}$ is replaced with a uniformly random element of $\Pi^{(i^*)}$, in the second refreshing query.

$\vdots$

Game Q_{ref} In this game, the third element $\overline{e^*}$ of the source ciphertext $\overline{C^*}$ is replaced with a uniformly random element of $\Pi^{(i^*)}$, in the Q_{ref}-th refreshing query.

Game $(Q_{\mathsf{ref}}+1)$ In this game, the third element e^* of the challenge ciphertext C^* is replaced with a uniformly random element of $\Pi^{(i^*)}$.

In Game $(Q_{\mathsf{ref}}+1)$, it is obvious that the adversary wins with the probability $\Pr[T_{Q_{\mathsf{ref}}+1}] = 1/2$, since all the information of the plaintext m_b^* (hence the information on b) are removed.

To evaluate the differences between adversary's winning probability, we consider a further game-hopping between Game $(\kappa-1)$ and Game κ $(1 \le \kappa \le Q_{\mathsf{ref}})$ as follows:

SubGame κ.0 This is identical to Game $(\kappa-1)$.

SubGame κ.1 In this game, the second component $\overline{x^*}$ of the source ciphertext $\overline{C^*}$ is chosen uniformly at random from $X'^{(i^*)} \setminus L^{(i^*)}$ instead of $L^{(i^*)}$, in the κ-th refreshing process.

SubGame κ.2 In this game, the challenger computes the hash values for the regular inputs using $\mathsf{ProjHash}$ instead of Hash in Enc, Dec, and Eval. Here the challenger obtains the corresponding witness by an exhaustive search. (Note that now the challenger becomes not PPT, but this does not affect the following game-hopping.)

SubGame $\kappa.3$ In this game, Eval and Dec oracles also rejects inputs that are irregular and not in the List. We refer to this modified rule as an **enhanced rejection rule**, and the original rule as an **original rejection rule**.

SubGame $\kappa.4$ In this game, the third element of the source ciphertext is calculated by $\overline{e^*} \leftarrow \Pi^{(i^*)}$, during the κ-th refreshing process.

SubGame $\kappa.5$ In this game, the modification made in SubGame $\kappa.3$ is restored, i.e., we change the rejection rule from enhanced to original.

SubGame $\kappa.6$ In this game, the modification made in SubGame $\kappa.2$ is restored, i.e., all the hash values in Enc, Dec, and Eval are computed using Hash (and now the challenger becomes PPT again).

SubGame $\kappa.7$ In this game, the modification made in SubGame $\kappa.1$ is restored, i.e., this game is identical to Game κ.

Moreover, we also consider the game-hopping from SubGame $(Q_{\mathsf{ref}}+1).0$ to SubGame $(Q_{\mathsf{ref}}+1).7$ in a similar manner. In these subgames, the elements of the challenge ciphertext C^* are replaced instead of the source ciphertext.

Let us evaluate the differences between the adversary's winning probability. It is obvious that the probability does not change in the modification from SubGame $\kappa.1$ to SubGame $\kappa.2$ and from SubGame $\kappa.5$ to SubGame $\kappa.6$, since the output distributions of Enc, Dec, and Eval in those games are identical. Therefore, from the triangle inequality, we have

$$\begin{aligned} |\Pr[T_0] - \Pr[T_{Q_{\mathsf{ref}}+1}]| &= \left| \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa-1}] - \Pr[T_\kappa]) \right| \\ &= \left| \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{i=1}^{7} (\Pr[T_{\kappa.i-1}] - \Pr[T_{\kappa.i}]) \right| \\ &= |\delta_1 + \delta_1' + \delta_2 + \delta_2' + \delta_3| \\ &\le |\delta_1| + |\delta_1'| + |\delta_2| + |\delta_2'| + |\delta_3| \ , \end{aligned} \tag{4}$$

where δ_1, δ_1', δ_2, δ_2', and δ_3 are defined as

$$\begin{aligned} \delta_1 &= \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.0}] - \Pr[T_{\kappa.1}]) \ , \quad & \delta_1' &= \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.6}] - \Pr[T_{\kappa.7}]) \ , \\ \delta_2 &= \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.2}] - \Pr[T_{\kappa.3}]) \ , \quad & \delta_2' &= \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.4}] - \Pr[T_{\kappa.5}]) \ , \\ \delta_3 &= \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.3}] - \Pr[T_{\kappa.4}]) \ . \end{aligned}$$

From now on, we evaluate each term in the right-hand side of the inequality (4).

A.3 Evaluation of δ_1 and δ_1'

To evaluate δ_1, we construct an algorithm $\mathcal{B}$ against the hardness of the subset membership problem of $\boldsymbol{P}^{(i^*)}$, using the adversary $\mathcal{A}$ for the KH-CCA game as a black box. $\mathcal{B}$ takes x^* as an input, and works as follows:

1. $\mathcal{B}$ chooses κ uniformly at random from $1 \le \kappa \le Q_{\mathsf{ref}} + 1$.

2. $\mathcal{B}$ simulates the KH-CCA game against $\mathcal{A}$ where, for the κ chosen above, it does the following.

- Case with $\kappa \neq Q_{\mathsf{ref}} + 1$: Until the $(\kappa - 1)$-th refreshing query, $\mathcal{B}$ chooses the third element $\overline{e^*}$ of the source ciphertext uniformly at random from $\Pi^{(i^*)}$. In the κ-th refreshing process, $\mathcal{B}$ sets its input x^* as the second component of the source ciphertext.
- Case with $\kappa = Q_{\mathsf{ref}} + 1$: In all the refreshing queries, $\mathcal{B}$ chooses the third element $\overline{e^*}$ of the source ciphertext uniformly at random from $\Pi^{(i^*)}$. Then B sets its input x^* as the second component of the challenge ciphertext.

3. $\mathcal{B}$ outputs 1 if the bit it chose as the challenger for the KH-CCA game coincides with $\mathcal{A}$'s output, and outputs 0 otherwise.

If $\mathcal{B}$'s input x^* was an element of $L^{(i^*)}$, the simulated game is identical to SubGame κ.0. On the other hand, if x^* was an element of $X'^{(i^*)} \setminus L^{(i^*)}$, the simulated game is identical to SubGame κ.1. Therefore, we have

$$\Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow L^{(i^*)}] = \frac{1}{Q_{\mathsf{ref}} + 1} \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \Pr[T_{\kappa.0}] \ ,$$

$$\Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow X'^{(i^*)} \setminus L^{(i^*)}] = \frac{1}{Q_{\mathsf{ref}} + 1} \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \Pr[T_{\kappa.1}] \ ,$$

and hence

$$\begin{aligned} |\delta_1| &= \left| \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.0}] - \Pr[T_{\kappa.1}]) \right| \\ &= (Q_{\mathsf{ref}} + 1) \left| \Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow L^{(i^*)}] - \Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow X'^{(i^*)} \setminus L^{(i^*)}] \right| \ . \end{aligned}$$

Since the subset membership problem of $\boldsymbol{P}^{(i^*)}$ is hard relative to $X'^{(i^*)}$,

$$\left| \Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow L^{(i^*)}] - \Pr[\mathcal{B}(x^*) = 1 \mid x^* \leftarrow X'^{(i^*)} \setminus L^{(i^*)}] \right|$$

is negligible. Furthermore, from the fact that $Q_{\mathsf{ref}} = \mathsf{poly}(\ell)$, it holds that $|\delta_1|$ is negligible in ℓ.

Also, $|\delta'_1|$ can be shown to be negligible in a similar manner. To conclude, it is shown that both $|\delta_1|$ and $|\delta'_1|$ are negligible.

A.4 Evaluation of δ_2 and δ'_2

We give evaluation for δ_2 and δ'_2 in this section. For simplicity, here we only deal with the case where both $\widetilde{\boldsymbol{P}}^{(1)}$ and $\widetilde{\boldsymbol{P}}^{(2)}$ are information-theoretically universal$_2$. For the evaluation for the case where $\widetilde{\boldsymbol{P}}^{(1)}$ and $\widetilde{\boldsymbol{P}}^{(2)}$ only have first-adaptive computationally universal$_2$ property, see Appendix B.

To evaluate δ_2, let us consider another game-hopping between SubGame κ.2 and SubGame κ.3. Let Q be a number of Dec or Eval queries that the adversary makes.

SubSubGame κ.3.0 This game is identical to SubGame κ.2.

SubSubGame κ.3.1 In this game, we apply the enhanced rejection rule up to the first Dec or Eval query.

$\vdots$

SubSubGame $\kappa.3.Q$ In this game, we apply the enhanced rejection rule up to the Q-th Dec or Eval query, i.e., this game is identical to SubGame κ.3.

For $1 \le \rho \le Q$, let $R^{(\kappa.3.\rho)}$ be an event that the ρ-th query is not rejected in the original rejection rule but is rejected in the enhanced rejection rule in the SubSubGame $\kappa.3.\rho$. By applying triangle inequality and Lemma 1, δ_2 can be evaluated as

$$\begin{aligned} |\delta_2| &= \left| \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} (\Pr[T_{\kappa.2}] - \Pr[T_{\kappa.3}]) \right| \\ &= \left| \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} (\Pr[T_{\kappa.3.(\rho-1)}] - \Pr[T_{\kappa.3.\rho}]) \right| \\ &\le \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \left| \Pr[T_{\kappa.3.(\rho-1)}] - \Pr[T_{\kappa.3.\rho}] \right| \\ &\le \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R^{(\kappa.3.\rho)}] \ . \end{aligned} \tag{5}$$

We evaluate the right-hand side of the inequality (5) from now on.

First, let us assume $1 \le \kappa \le Q_{\mathsf{ref}}$. We divide $R^{(\kappa.3.\rho)}$ into several cases depending on the situation where the ρ-th query was made. When the ρ-th query was a Dec query (we denote its input as C), we consider the following four cases. Note that for all the cases, the input C is $\widehat{\boldsymbol{P}}$-forging and $\widetilde{\boldsymbol{P}}$-forging, $C \notin \mathsf{List}$, and if the query is a Dec query in the guess stage, RevEK is not queried before.

- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.1}$: The case where the ρ-th query was a Dec query in the find stage.
- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.2}$: The case where the ρ-th query was a Dec query made in the guess stage, before the κ-th refreshing query.
- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.3}$: The case where the ρ-th query was a Dec query made in the guess stage, after the κ-th refreshing query, and the reply to the κ-th refreshing query was regular.
- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}$: The case where the ρ-th query was a Dec query made in the guess stage, after the κ-th refreshing query, and the reply to the κ-th refreshing query was irregular.

When the ρ-th query was an Eval query (we denote its input as C, C'), we consider the following three cases. Note that for all the cases, the inputs C and C' are both $\widehat{\boldsymbol{P}}$-consistent, and at least one of C and C' is irregular and not in the List. Also note that the RevEK has not been queried yet.

- $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}$: The case where the ρ-th query was a κ-th refreshing query or an Eval query made before the κ-th refreshing query.
- $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}$: The case where the ρ-th query was an Eval query made after the κ-th refreshing query, and the reply to the κ-th refreshing query was regular.
- $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.3}$: The case where the ρ-th query was an Eval query made after the κ-th refreshing query, and the reply to the κ-th refreshing query was irregular.

Since all the cases above are disjoint, we have

$$\Pr[R^{(\kappa.3.\rho)}] = \sum_{i=1}^{4} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.i}] + \sum_{i=i}^{3} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.i}] \ . \tag{6}$$

From now on, we evaluate the right-hand side of this equation more precisely. Note that in SubSubGame $\kappa.3.\rho$, replies of Eval oracles before the ρ-th query are always regular, except for the κ-th refreshing query.

When the ρ-th query is a Dec query, the probability for the rejection can be evaluated as follows:

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.1}]$: Just before making this query, the only information related to the secret key $\widehat{\mathsf{hk}}$ that the adversary has gained is the public key $\widehat{\mathsf{hp}}$. Therefore, since $\widehat{\boldsymbol{P}}$ is universal$_1$, the probability that the adversary calculates the correct value of $\widehat{\pi}$ corresponding to an irregular ciphertext is negligible.

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.2}]$: Since this query is made before the κ-th refreshing query, the only information related to the secret key $\widehat{\mathsf{hk}}$ that the adversary has gained is the public key $\widehat{\mathsf{hp}}$ just before this query. Therefore, since $\widehat{\boldsymbol{P}}$ is universal$_1$, the probability that the adversary calculates the correct value of $\widehat{\pi}$ corresponding to an irregular ciphertext is negligible.

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.3}]$: Although this query was made after the κ-th refreshing query, since the reply to the κ-th refreshing query was regular, the only information related to the secret key $\widehat{\mathsf{hk}}$ that the adversary has gained is the public key $\widehat{\mathsf{hp}}$ just before this query. Therefore, since $\widehat{\boldsymbol{P}}$ is universal$_1$, the probability that the adversary calculates the correct value of $\widehat{\pi}$ corresponding to an irregular ciphertext is negligible.

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}]$: Just before this query, the information related to the secret key $\widetilde{\mathsf{hk}}$ that the adversary has gained is the public key $\widetilde{\mathsf{hp}}$ and the reply to the κ-th refreshing query, which is a $\widetilde{\boldsymbol{P}}$-forging ciphertext. Therefore, since $\widetilde{\boldsymbol{P}}$ is universal$_2$, the probability that the adversary calculates the correct value of $\widetilde{\pi}$ corresponding to an irregular ciphertext is negligible.

When the ρ-th query is an Eval query, the probability for the rejection can be evaluated as follows:

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}]$: Just before making this query, the only information related to the secret key $\widetilde{\mathsf{hk}}$ that the adversary has gained is the public key $\widetilde{\mathsf{hp}}$. Therefore, since $\widetilde{\boldsymbol{P}}$ is universal$_1$, the probability that the adversary calculates the correct value of $\widetilde{\pi}$ corresponding to an irregular ciphertext is negligible.

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}]$: Although this query was made after the κ-th refreshing query, since the reply to the κ-th refreshing query was regular, the only information related to the secret key $\widetilde{\mathsf{hk}}$ that the adversary has gained is the public key $\widetilde{\mathsf{hp}}$ just before this query. Therefore, since $\widetilde{\boldsymbol{P}}$ is universal$_1$, the probability that the adversary calculates the correct value of $\widetilde{\pi}$ corresponding to an irregular ciphertext is negligible.

- $\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.3}]$: Just before this query, the information related to the secret key $\widetilde{\mathsf{hk}}$ that the adversary has gained is the public key $\widetilde{\mathsf{hp}}$ and the reply to the κ-th refreshing query, which is a $\widetilde{\boldsymbol{P}}$-forging ciphertext. Therefore, since $\widetilde{\boldsymbol{P}}$ is universal$_2$, the probability that the adversary calculates the correct value of $\widetilde{\pi}$ corresponding to an irregular ciphertext is negligible.

As a conclusion, it has been shown that the right-hand side of the equation (6) can be bounded using the same negligible function for all values of κ and ρ.

Next, for the case with $\kappa = Q_{\mathsf{ref}} + 1$, the event $R^{(\kappa.3.\rho)}$ can be divided into four cases below:

- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.1}$: The case whete the ρ-th query is a Dec query in the find stage.

- $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.2}$: The case whete the ρ-th query is a Dec query in the guess stage.

- $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}$: The case whete the ρ-th query is an Eval query in the find stage.

- $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}$: The case whete the ρ-th query is an Eval query in the guess stage.

The probability for these events can be shown to be negligible in the same manner as the case with $1 \le \kappa \le Q_{\mathsf{ref}}$. Hence the right-hand side of the equation (6) is also negligible in this case.

Combining the above discussion and the fact that Q and Q_{ref} are both $O(\mathsf{poly}(\ell))$, the whole of the right-hand side of (5) is negligible, which implies that $|\delta_2|$ is negligible. The negligibility of $|\delta_2'|$ can be shown in a similar manner.

A.5 Evaluation of δ_3

We evaluate δ_3 in this section. First, let us consider the case with $1 \le \kappa \le Q_{\mathsf{ref}}$. During the κ-th refreshing query in the SubGame κ.3, the third component of the source ciphertext is computed as $\overline{e^*} \leftarrow m_b^* + \mathsf{Hash}(\overline{x^*})$, where $\overline{x^*}$ is a random element from $X'^{(i^*)} \setminus L^{(i^*)}$. Since the only information related to the secret key hk that the adversary has gained just before this query is the public key hp, from the smoothness of $\boldsymbol{P}^{(i)}$ relative to $X'^{(i^*)} \setminus L^{(i^*)}$, the adversary's winning probability changes only negligibly even if we replace $\mathsf{Hash}(\overline{x^*})$ with a uniformly random element of $\Pi^{(i^*)}$ in the computation of $\overline{e^*}$. After applying this modification, $\overline{e^*}$ itself is a uniformly random element of $\Pi^{(i^*)}$. Therefore, $|\Pr[T_{\kappa.3}] - \Pr[T_{\kappa.4}]|$ can be bounded using common negligible function for all values of κ. By considering the same modification for the case with $\kappa = Q_{\mathsf{ref}} + 1$, we obtain

$$|\delta_3| \le \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} |\Pr[T_{\kappa.3}] - \Pr[T_{\kappa.4}]| = \mathrm{negl}(\ell) \ .$$

A.6 Conclusion of the Security Proof

We have shown in the previous subsections that $|\delta_1|$, $|\delta_1'|$, $|\delta_2|$, $|\delta_2'|$, and $|\delta_3|$ are all negligible. Hence, from the inequality (4), we have

$$|\Pr[T_0] - \Pr[T_{Q_{\mathsf{ref}}+1}]| = \mathrm{negl}(\ell) \ .$$

Moreover, since $\Pr[T_0] = \Pr[T_{\text{pre-0}}]$ and $\Pr[T_{Q_{\mathsf{ref}}+1}] = 1/2$, we have

$$|\Pr[T_{\text{pre-0}}] - 1/2| = \mathrm{negl}(\ell) \ .$$

This completes the proof of Theorem 2.

B Evaluation of δ_2 and δ_2' for the Case with Computational Universal$_2$ Property

In this section, we give an evalution of δ_2 and δ_2' for the case where $\widetilde{\boldsymbol{P}}^{(1)}$ and $\widetilde{\boldsymbol{P}}^{(2)}$ only satisfy first-adaptive computationally universal$_2$ property.

Even in this case, we can evaluate δ_2 and δ_2' by a game-hopping from SubGame κ.2 to SubGame κ.3, similar to the one in Section A.4. From the equations (5) and (6), we have

$$|\delta_2| \le \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \left(\sum_{i=1}^{4} \Pr[R_{\mathsf{Dec}.i}^{(\kappa.3.\rho)}] + \sum_{i=i}^{3} \Pr[R_{\mathsf{Eval}.i}^{(\kappa.3.\rho)}] \right) \ . \tag{7}$$

The term $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R_{\mathsf{Dec}.i}^{(\kappa.3.\rho)}]$ $(i = 1, 2, 3)$ in the right-hand side of the inequality can be evaluated in exactly the same way as the argument in Section A.4, by applying universal$_1$ property of $\widehat{\boldsymbol{P}}$. From now on, we evaluate the rest of the terms by applying first-adaptive computationally universal$_2$ property of $\widetilde{\boldsymbol{P}}$.

B.1 Evaluation of $R_{\mathsf{Dec}.4}^{(\kappa.3.\rho)}$ and $R_{\mathsf{Eval}.3}^{(\kappa.3.\rho)}$

To evaluate $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R_{\mathsf{Dec}.4}^{(\kappa.3.\rho)}]$, we construct an adversary $\mathcal{B}$ against first-adaptive computationally universal$_2$ property of $\widetilde{\boldsymbol{P}}$ in the following manner:

1. $\mathcal{B}$ receives the public key $\widetilde{\mathsf{hp}}$ from the challenger of the first-adaptive computationally universal$_2$ game. Then $\mathcal{B}$ generates public and secret key pairs $(\mathsf{hp}, \mathsf{hk})$ and $(\widehat{\mathsf{hp}}, \widehat{\mathsf{hk}})$ using the key generation algorithm of each HPS.

2. $\mathcal{B}$ chooses κ and ρ uniformly at random from the range $1 \le \kappa \le Q_{\mathsf{ref}} + 1$ and $1 \le \rho \le Q$, respectively.

3. $\mathcal{B}$ simulates SubSubGame $\kappa.3.\rho$ up to ρ-th query. If $1 \leq \kappa \leq Q_{\mathsf{ref}}$, $\mathcal{B}$ simulates the κ-th refreshing query as follows:

   (a) $\mathcal{B}$ chooses the second component $\overline{x^*}$ of the source ciphertext uniformly at random from $X'^{(i^*)} \setminus L^{(i^*)}$, and computes the components $(x, e, \widehat{\pi})$ in the output of the refreshing query using $\overline{x^*}$ (in the same way as an ordinary refreshing query).

   (b) $\mathcal{B}$ queries $(x, e, \widehat{\pi})$ to Hash oracle and checks if the oracle returns $\perp$. If not, $\mathcal{B}$ aborts. (This operation is for checking that the reply to the refreshing query is irregular, since this has to be the case with the event $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}$.)

   (c) $\mathcal{B}$ submits $(x, e, \widehat{\pi})$ to the challenger of the first-adaptive computationally universal$_2$ game, and receives $\widetilde{\pi}$ as the reply. Then $\mathcal{B}$ sends back $(x, e, \widehat{\pi}, \widetilde{\pi})$ as the reply to the refreshing query.

   Furthermore, if $\kappa = Q_{\mathsf{ref}} + 1$, $\mathcal{B}$ computes the challenge ciphertext in the following manner:

   (a) $\mathcal{B}$ chooses the second component x^* of the challenge ciphertext uniformly at random from $X'^{(i^*)} \setminus L^{(i^*)}$, and uses it to compute the third and the fourth components $e^*, \widehat{\pi}^*$.

   (b) $\mathcal{B}$ submits $(x^*, e^*, \widehat{\pi}^*)$ to the challenger of the first-adaptive computationally universal$_2$ game and receives $\widetilde{\pi}^*$ as its reply. Then $\mathcal{B}$ sets $(x^*, e^*, \widehat{\pi}^*, \widetilde{\pi}^*)$ as the challenge ciphertext.

4. Finally, $\mathcal{B}$ works as follows to the ρ-th query:

   (a) $\mathcal{B}$ aborts if the ρ-th query is not a Dec query. We assume that the ρ-th query was a Dec query with input $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$ from now on.

   (b) If the input C of the query was already in the List, $\mathcal{B}$ aborts.

   (c) $\mathcal{B}$ outputs the second, the third, and the fourth components $(x, e, \widehat{\pi})$ of C and the fifth component $\widetilde{\pi}$ of C as the output of the first-adaptive computationally universal$_2$ game.

We also assume that $\mathcal{B}$ aborts if the event $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}$ becomes impossible to achieve during the simulation.

It is obvious that the simulation above is correct. In this situation, $\mathcal{B}$ wins the first-adaptive computationally universal$_2$ game if and only if the event $R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}$ happens. Therefore, the value

$$\frac{1}{Q(Q_{\mathsf{ref}}+1)} \sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}]$$

and the winning probability in the first-adaptive computationally universal$_2$ game are exactly the same.

Moreover, let $\mathcal{B}'$ be the algorithm that is obtained by modifying $\mathcal{B}$ in the following manner: in Step 3 of $\mathcal{B}$, $\mathcal{B}'$ chooses the second component of the source ciphertxt (or a challenge ciphertext) from the distribution over X that is indistinguishable from the uniform distribution over $X'^{(i^*)} \setminus L^{(i^*)}$, instead of directly sampling from $X'^{(i^*)} \setminus L^{(i^*)}$. The difference between the winning probabilities of $\mathcal{B}$ and $\mathcal{B}'$ is negligible due to the approximate samplability. Also, the winning probability of $\mathcal{B}'$ is negligible, since $\widetilde{\boldsymbol{P}}$ is first-adaptive computationally universal$_2$. Therefore, it follows that $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Dec}.4}]$ is negligible.

We can evaluate $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1} \sum_{\rho=1}^{Q} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.3}]$ in almost the same manner. We only need to modify Step 4 in $\mathcal{B}$ as follows:

4. Finally, $\mathcal{B}$ works as follows to the ρ-th query:

   (a) $\mathcal{B}$ aborts if the ρ-th query is not an Eval query. We assume that the ρ-th query was an Eval query with inputs $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$ and $C' = (i', x', e', \widehat{\pi}', \widetilde{\pi}')$, from now on.

   (b) If both C and C' are already involved in the List, $\mathcal{B}$ aborts. We assume that $C \notin \mathsf{List}$, without loss of generality.

   (c) $\mathcal{B}$ outputs $(x, e, \widehat{\pi})$ in the input of the query and $\widetilde{\pi}$ as the outputs of first-adaptive computationally universal$_2$ game.

B.2 Evaluation of $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}$ and $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}$

To evaluate $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1}\sum_{\rho=1}^{Q} Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}]$ and $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1}\sum_{\rho=1}^{Q} \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}]$, we construct an adversary $\mathcal{B}$ against first-adaptive computationally universal$_2$ property of $\widetilde{\boldsymbol{P}}$, as in the previous section. The description of $\mathcal{B}$ is given below.

1. $\mathcal{B}$ receives the public key $\widetilde{\mathsf{hp}}$ from the challenger of the first-adaptive computationally universal$_2$ game. Then $\mathcal{B}$ generates public and secret key pairs $(\mathsf{hp}, \mathsf{hk})$ and $(\widehat{\mathsf{hp}}, \widehat{\mathsf{hk}})$ using the key generation algorithm of each HPS.
2. $\mathcal{B}$ chooses κ and ρ uniformly at random from the range $1 \le \kappa \le Q_{\mathsf{ref}} + 1$ and $1 \le \rho \le Q$, respectively.
3. $\mathcal{B}$ simulates SubSubGame $\kappa.3.\rho$ up to ρ-th query. If $1 \le \kappa \le Q_{\mathsf{ref}}$, $\mathcal{B}$ simulates the κ-th refreshing query as follows:
   (a) $\mathcal{B}$ chooses the second component $\overline{x^*}$ of the source ciphertext uniformly at random from $X'^{(i^*)} \setminus L^{(i^*)}$, and computes the components $(x, e, \widehat{\pi})$ in the output of the refreshing query using $\overline{x^*}$ (in the same way as an ordinary refreshing query).
   (b) $\mathcal{B}$ queries $(x, e, \widehat{\pi})$ to Hash oracle and aborts if the oracle returns $\perp$. Otherwise, $\mathcal{B}$ answers the refreshing query with $(x, e, \widehat{\pi}, \widetilde{\pi})$, where $\widetilde{\pi}$ is the reply to the Hash query that $\mathcal{B}$ made.
4. $\mathcal{B}$ chooses a tuple $(x, e, \widehat{\pi})$ uniformly at random, and submits it to the challenger of the first-adaptive computationally universal$_2$ game.
5. Finally, $\mathcal{B}$ works as follows to the ρ-th query:
   (a) $\mathcal{B}$ aborts if the ρ-th query is not an Eval query. We assume that the ρ-th query was an Eval query with inputs $C = (i, x, e, \widehat{\pi}, \widetilde{\pi})$ and $C' = (i', x', e', \widehat{\pi}', \widetilde{\pi}')$, from now on.
   (b) If both C and C' are already involved in the List, $\mathcal{B}$ aborts. We assume that $C \notin \mathsf{List}$, without loss of generality.
   (c) $\mathcal{B}$ outputs $(x, e, \widehat{\pi})$ in the input of the query and $\widetilde{\pi}$ as the outputs of first-adaptive computationally universal$_2$ game.

We also assume that $\mathcal{B}$ aborts if the event $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}$ or $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}$ becomes impossible to achieve during the simulation.

It is obvious that the above simulation is correct. In this situation, $\mathcal{B}$ wins the first-adaptive computationally universal$_2$ game if and only if the event $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}$ or $R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}$ happens, except for the case where the random value that $\mathcal{B}$ submits coincides with the value that $\mathcal{B}$ outputs (which happens with a negligible probability). Therefore, the value

$$\frac{1}{Q(Q_{\mathsf{ref}}+1)}\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1}\sum_{\rho=1}^{Q}(\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}] + \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}])$$

and $\mathcal{B}$'s winning probability in the first-adaptive computationally universal$_2$ game have only negligible difference. Thus, by applying approximate samplability of $X'^{(i^*)} \setminus L^{(i^*)}$ in the same manner as Section B.1, the fact that $\sum_{\kappa=1}^{Q_{\mathsf{ref}}+1}\sum_{\rho=1}^{Q}(\Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.1}] + \Pr[R^{(\kappa.3.\rho)}_{\mathsf{Eval}.2}])$ is negligible follows from the first-adaptive computationally universal$_2$ property of $\widetilde{\boldsymbol{P}}$.

B.3 Conclusion of the Proof

From the above arguments, it follows that the right-hand side of the inequality (7) is negligible. Hence, $|\delta_2|$ is also negligible in this case. The negligibility of $|\delta_2'|$ can be shown in a similar manner.