Cryptosystems Resilient to Both Continual Key Leakages and Leakages from Hash Functions

Guangjun Fan[1], Yongbin Zhou[2], Chengyu Hu[3], Dengguo Feng[1]

[1] Trusted Computing and Information Assurance Laboratory,Institute of Software,Chinese Academy of Sciences,Beijing,China
guangjunfan@163.com , feng@tca.iscas.ac.cn

[2] State Key Laboratory of Information Security,Institute of Information Engineering,Chinese Academy of Sciences,Beijing,China
zhouyongbin@iie.ac.cn

[3] School of Computer Science and Technology,Shandong University, Jinan,China
hcy@sdu.edu.cn

Abstract. Yoneyama et al. introduced Leaky Random Oracle Model (LROM for short) at ProvSec2008 in order to discuss security (or insecurity) of cryptographic schemes which use hash functions as building blocks when leakages from pairs of input and output of hash functions occur. This kind of leakages occurs due to various attacks caused by sloppy usage or implementation. Their results showed that this kind of leakages may threaten the security of some cryptographic schemes. However, an important fact is that such attacks would leak not only pairs of input and output of hash functions, but also the secret key. Therefore, LROM is rather limited in the sense that it considers leakages from pairs of input and output of hash functions alone, instead of taking into consideration other possible leakages from the secret key simultaneously. On the other hand, many other leakage models mainly concentrate on leakages from the secret key and ignore leakages from hash functions for a cryptographic scheme exploiting hash functions in these leakage models. Some examples show that the above drawbacks of LROM and other leakage models may cause insecurity of some schemes which are secure in the two kinds of leakage model.

In this paper, we present an augmented model of both LROM and some leakage models, which both the secret key and pairs of input and output of hash functions can be leaked. Furthermore, the secret key can be leaked continually during the whole life cycle of a cryptographic scheme. Hence, our new model is more universal and stronger than LROM and some leakage models (e.g. only computation leaks model and bounded memory leakage model). As an application example, we also present a public key encryption scheme which is provably IND-CCA secure in our new model.

Keywords: Leakage Resilient Cryptography, Leaky Random Oracle Model, Public Key Cryptography, Cramer-Shoup cryptosystem.

1 Introduction

Hash functions are one of the most important building blocks of cryptographic schemes. For example, public key encryption scheme, digital signature, authenticated key exchange etc. On one hand, hash functions can be exploited to construct cryptographic schemes in the standard model (SM). For example, Cramer-Shoup cryptosystem [9] is a public key encryption scheme which is based on universal one-way hash function family in SM. On the other hand, for a cryptographic scheme in the random oracle model [1] (ROM), the random oracle is usually instantiated with hash functions.

If possible, a cryptographic scheme will be implemented on some device in practice. A fact that cannot be neglected is that any implementation of a cryptographic scheme can be threatened by attacks caused by sloppy usage or implementation (For example, physical attacks such as side-channel attacks [6,7,8] and cold boot attacks [4]). These attacks may leak sensitive information in the cryptographic scheme. In [5], Yoneyama et al. applied this view to hash functions. They considered the situation that all contents of pairs of input and output of hash functions used by a cryptographic scheme can be leaked to an adversary. These leakages may also caused by sloppy usage or implementation. A possible example of sloppy usage is that pairs of input and output of hash functions may remain in some insecure area of the memory for reusing of hash values in order to reduce computational costs or for failing to release temporary memory area, then contents of the memory may be revealed without advanced implementation attacks [5]. If a cryptographic scheme is implemented without any sloppy usage, an adversary can also try to attack the implementation and may obtain pairs of input and output of hash functions by side-channel attacks [6,7,8], cold boot attack[1] [4], and malicious Trojan Horse programs etc. Thus, even if we successfully developed exceedingly secure hash functions, such kind of leakages might be possible. In [5], Yoneyama et al. formulated Leaky Random Oracle Model (LROM) capturing this kind of leakages in order to discuss security (or insecurity) of cryptographic schemes which use hash functions as building blocks when such kind of leakages occurs. Yoneyama et al. have analyzed the security of five prevailing cryptographic schemes in LROM including Full Domain Hash [1], Optimal Asymmetric Encryption Padding [17], Cramer-Shoup cryptosystem, Kurosawa-Desmedt cryptosystem [18] and NAXOS [19].

1.1 Motivation

An important fact is that an adversary can obtain not only leakages from pairs of input and output of hash functions, but also leakages from the secret key of a cryptographic scheme from attacks caused by sloppy usage or implementation. For example, most side channel attacks target the secret key of a cryptographic scheme because the secret key is fixed in every invocation and can be revealed easier than pairs of input or output of hash functions. In addition, Halderman

[1] The intermediate computation result will be stored in memory.

et al. [4] put forward cold boot attack in which an adversary can learn a (noisy) version of the *entire* memory[1] even if no computation is going on. What's more, malicious Trojan Horse programs can be designed to obtain both the secret key and pairs of input and output of hash functions by an adversary easily. In these scenarios, an adversary can obtain both leakages from the secret key and leakages from hash functions.

On one hand, LROM only considers leakages from hash functions and assumes that the secret key of a cryptographic scheme that is secure in LROM will not be leaked to an adversary. However, in many real world settings, leakages from the secret key completely compromise the security of many cryptographic schemes. The security of any cryptographic scheme that is secure in LROM may not holds when the adversary can obtain additional leakages from the secret key. For example, if an adversary obtains the *stateless* secret key of Cramer-Shoup cryptosystem[2] completely from continual leakages, he can easily break the scheme. Therefore, LROM is very limited.

On the other hand, many other leakage models [2,3,10,11,13,15,16,20,27,28,29, 30] which mainly concentrate on leakages from the secret key are given out in recent years. However, the result of the paper [5] shows that leakages from hash functions used by a cryptographic scheme that is secure in these leakage models may threaten its security. A specific example is the public key encryption scheme in Section 4 of the paper [2] when the scheme is instantiated by a hash function[3]. If all contents of pairs of input and output of the hash function are leaked, the above scheme will not be secure any more. Therefore, this is a drawback of these leakage models which ignore leakages from hash functions[4].

In order to improve both LROM and other leakage models, we try to formulate a new leakage model in which both the secret key and pairs of input and output of hash functions can be leaked. We believe that our new model is more universal and stronger than LROM due to leakages from the secret key and some leakage models due to leakages from hash functions. We also try to build provably secure cryptographic schemes in our new model.

1.2 Our Contribution

The main contributions of this paper are two-fold as follows. First, we introduce a new leakage model which is more universal and stronger than LROM and some other leakage models. Second, we give out a public key encryption scheme that is provably secure in this new leakage model as an application example.

Our New Model For a cryptographic scheme which exploits hash functions, our new model allows an adversary to obtain both leakages from the secret key

[1] Note that the secret key must be stored in the memory.
[2] Cramer-Shoup cryptosystem is secure in LROM [5].
[3] Note that any family of pairwise independent hash functions is an average-case strong extractor [14].
[4] Here we assume the secret key is not an input or an output of hash functions used by a cryptographic scheme in these leakage models.

continually and leakages from pairs of input and output of the hash functions in the same way as LROM. Any cryptographic scheme which is secure in our new model will be also secure in LROM. However, any cryptographic scheme which is secure in LROM may not be secure in our new model (For example, the adversary can obtain the *stateless* secret key completely from continual leakages to break the scheme.). Therefore, our new model is more universal and stronger than LROM. Additionally, our new model is more universal and stronger than some other leakage models [2,3,10,13] for cryptographic schemes in these leakage models exploit hash functions (both in SM and ROM). Because these leakage models ignore leakages from hash functions and only consider *bounded* leakages from the secret key.

A Public Key Encryption Scheme in Our New Model We also construct a public key encryption scheme which is IND-CCA secure in our new model without any complex assumptions and cryptographic tools. Our new public key encryption scheme is based on Cramer-Shoup cryptosystem, Hiding Subspaces principle [11,12], and a new assumption which is equivalent to the Decisional Diffie-Hellman assumption (DDH assumption). This new scheme is better than some known practical leakage resilient public key encryption schemes because the new scheme can tolerate more leakages and has higher security level. Furthermore, the new scheme can be implemented easily in practice. Therefore, we believe it can be used widely.

1.3 Related Work

Work on tolerating leakages was initiated by Rivest and Boyko [21,22] in the context of increasing the cost of brute-force attacks on block ciphers and efficiency issues. Then exposure-resilient cryptography [23,24,25] considers simple leakage functions that reveal a subset of the bits of the secret key or the internal memory of the cryptographic device. In contrast to these works, more powerful leakage function (i.e. *efficiently computable leakage function*) that can perform some *global computation* on the secret key are used to describe leakages from the secret key. Micali et al. [26] proposed to construct and study formal models that capture this general type of leakages. This study has led to two distinct strands of work as follows.

Bounded Leakage Models This line of work considers the leakage models that allow an adversary to obtain the output of any efficiently computable leakage function f, of his choice, to the secret key SK. The unique restriction of f is that the output $f(SK)$ does not reveal the entire secret key. For example, Akavia et al. [10] restricted the output length of f is bounded by the length of the secret key, i.e. $|f(SK)| \ll |SK|$. Many other papers can be found in these leakage models [2,3,10,13,16,27,28,29].

Continual Leakage Models This line of work considers the case where leakages are continual, i.e. a bounded amount of information about the secret key is leaked in each time period, but the overall leakages in the whole life cycle of a cryptographic scheme is unbounded. It is easy to see that to guarantee any security in this kind of leakage models the secret key must necessarily be *stateful* which means that the secret key must be updated between time-periods while the public key remains unchanged. Micali et al. [26] proposed to study security against continual leakages under the *"only computation leaks information"* axiom. Some papers [31,32] design leakage resilient cryptographic schemes under this axiom. Furthermore, there exist several papers [11,20,30] in continual leakage models without the *"only computation leaks information"* axiom.

However, none of the above work considers leakages from the secret key and leakages from pairs of input and output of hash functions simultaneously.

1.4 Organization of This Paper

The remainder of this paper is organized as follows. In section 2, we introduce some basic notations, concepts and backgrounds. We present our new leakage model in section 3. Our provable secure public key encryption scheme in our new model is introduced in section 4. In section 4, we also prove the security of this scheme. We conclude this paper in section 5.

2 Preliminaries

In this section, we first present some notations and concepts used throughout the paper. Second, we review LROM. Third, we introduce the security of Cramer-Shoup cryptosystem in SM and LROM. Finally, we introduce the computational assumption which is used in this paper.

2.1 Notations and Concepts

The *statistical distance* between two random variables X and Y over a common domain S is defined by $\mathsf{SD}(X,Y) = \frac{1}{2}\sum_{x\in S}|\Pr[X=x]-\Pr[Y=x]|$. We write $X \overset{s}{\approx}_{\epsilon} Y$ to denote $\mathsf{SD}(X,Y) \leq \epsilon$ and just plain $X \overset{s}{\approx} Y$ if the statistical distance is negligible in some security parameter. In the latter case, we say that X, Y are statistically indistinguishable.

Let Gen denotes a probabilistic polynomial-time algorithm that takes as input a security parameter and outputs a triple $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a group of order q and is generated by $g \in \mathbb{G}$.

Let $\boldsymbol{v} = (v_1, v_2, \ldots, v_n), v_i \in \mathbb{Z}_q$ is a vector, we use $g^{\boldsymbol{v}}$ to denote the vector $(g^{v_1}, g^{v_2}, \ldots, g^{v_n})$. If $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ and $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ are two vectors in $\mathbb{Z}_q^n$, we use $\langle \boldsymbol{t}, \boldsymbol{s} \rangle = t_1s_1 + t_2s_2 + \cdots + t_ns_n$ to denote the inner product of the two vectors. For a random number $r \in \mathbb{Z}_q$, $r\boldsymbol{t} = (rt_1, rt_2, \ldots, rt_n)$ is also a vector in $\mathbb{Z}_q^n$. Let vector $\mathbf{1}_{\boldsymbol{n}} = (1, 1, \ldots, 1)$, there exist n components in the vector.

If $\boldsymbol{A} \in \mathbb{Z}_p^{n\times m}$ is a $n \times m$ matrix of scalars, we use $\mathsf{colspan}(\boldsymbol{A})$ to denote the subspaces spanned by the columns of $\boldsymbol{A}$.

2.2 Leaky Random Oracle Model

In order to concentrate on effects of leakages, LROM supposes that hash functions are ideal as random oracle without considering any internal structure of real hash functions but pairs of input and output of hash functions can be leaked to an adversary [5]. In LROM, pairs of input and output of a hash function are stored in a hash list. The hash list is a virtual notation and is just used for describing leakages. The hash list is not kept in memory by an implementor without any sloppy usage. However, an adversary can also obtain all contents of the hash list by various attacks[1]. On the other hand, some sloppy usage of hash functions (Please see the example in Section 1.) causes direct leakages of the hash list. LROM is shown in the following.

Definition 1. ***(Leaky Random Oracle Model)*** *LROM is a model assuming the leaky random oracle. We suppose a hash function $H : X \to Y$ such that $x_i \in X$, $y_i \in Y$ (i is an index), and X and Y are both finite sets. Also, let $\mathcal{L}_H$ be the hash list of H. We say H is a leaky random oracle if H can be simulated by the following procedure:*

Initialization: $\mathcal{L}_H \leftarrow \perp$

Hash query: *For a hash query x_i to H, behave as follows:*

If $x_i \in \mathcal{L}_H$, then find y_i corresponding to x_i and output y_i as the answer to the hash query. If $x_i \notin \mathcal{L}_H$, then choose y_i randomly, add pair (x_i, y_i) to $\mathcal{L}_H$ and output y_i as the answer to the hash query.

Leak hash query:[2] *For a leak hash query to H, output all contents of the hash list $\mathcal{L}_H$.*

If a cryptographic scheme is not secure in LROM, the cryptographic scheme must not be secure against various attacks caused by sloppy usage or implementation when it uses any real hash functions. If a cryptographic scheme is secure in LROM, it may still be insecure when it uses real hash functions against these attacks. Therefore, Cramer-Shoup cryptosystem is considered whether it is secure in LROM in [5].

2.3 The Security of Cramer-Shoup Cryptosystem

We ignore the description of Cramer-Shoup cryptosystem and only introduce its security here. In [9], the security of Cramer-Shoup cryptosystem in SM stated in the following lemma:

Lemma 1 (Security of Cramer-Shoup cryptosystem in SM). *If the hash function H is chosen from a family of universal one-way hash functions and the*

---

[1] The attacks include side-channel attacks, cold boot attack, and malicious Trojan Horse programs etc.

[2] The leak hash query is identical to the leak query in Definition 1 in [5]. We rename the leak query as leak hash query here because we will define leakage query in Definition 2 in Section 3 of this paper.

DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA secure.

In [5], the security of Cramer-Shoup cryptosystem in LROM is analyzed. Cramer-Shoup cryptosystem is also secure in LROM.

Lemma 2 (Security of Cramer-Shoup cryptosystem in LROM). *If the DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA secure where H is modeled as a leaky random oracle.*

2.4 Computational Assumption

In this paper, we use an assumption which is equivalent to the DDH assumption as follows.

The Generalized Diffie-Hellman assumption. *The Generalized Decisional Diffie-Hellman (GDDH) assumption is that the two ensembles*

$$\{\mathbb{G}, \{g_1, \ldots, g_n\}, \{g_{n+1}, \ldots, g_{2n}\}, \{g_1^r, \ldots, g_n^r\}, \{g_{n+1}^r, \ldots, g_{2n}^r\}\},$$

$$\{\mathbb{G}, \{g_1, \ldots, g_n\}, \{g_{n+1}, \ldots, g_{2n}\}, \{g_1^{r_1}, \ldots, g_n^{r_1}\}, \{g_{n+1}^{r_2}, \ldots, g_{2n}^{r_2}\}\}$$

are computationally indistinguishable, where $(\mathbb{G}, q, g) \leftarrow \mathsf{Gen}(1^k)$, and the elements $g_1, g_2, \ldots, g_{2n} \in \mathbb{G}$ and $r, r_1, r_2 \in \mathbb{Z}_q$ are chosen independently and uniformly at random.

The GDDH assumption is not mentioned in previous work. We show that the GDDH assumption and the DDH assumption are equivalent in Theorem 1. The proof of Theorem 1 is shown in Appendix A.

Theorem 1. *The GDDH assumption and the DDH assumption are equivalent.*

3 Our New Model

In this paper, we use the same notations and assumptions as LROM. In our new model, we consider both leakages from the secret key and leakages from the hash lists of hash functions.

The secret key is *stateless* means that it is stored in memory and remains unchanged during the whole life cycle of a cryptographic scheme. It is well known that if the secret key is leaked to an adversary entirely, no leakage resilient cryptographic scheme can be designed. Therefore, for a leakage model where the secret key is stateless, only a part of information of the secret key can be leaked to the adversary during the whole life cycle of a cryptographic scheme in this leakage model.

However, this leakage scenario about the secret key is not suitable for our new model. During the whole life cycle of a cryptographic scheme, our new model allows the adversary can obtain *all* contents of the hash list as that in LROM. It is unreasonable to assume the adversary can obtain only a part of information of the secret key during the whole life cycle of the same cryptographic scheme. But the adversary can not obtain the secret key entirely. Therefore, in our new model, the secret key must be *stateful* (Like the schemes in Continual Leakage Models) and be updated before the adversary obtains enough information about the secret key to carry out various attacks. The amount of information about the secret key leaked in each time period is bounded, but the overall leakages from the secret key in the whole life cycle of a cryptographic scheme are unbounded.

In our new model, we use an efficiently computable leakage function $Leak$ to describe leakages from the secret key in each time period. The input of $Leak$ is the secret key SK. The only restriction of $Leak$ is that the output length of it is bounded by the length of the secret key (i.e. $|Leak(SK)| \ll |SK|$). The leakage function in LROM can be viewed as an identity function. If we use a simpler leakage function f which can only output a subset of the bits of the secret key, the leakage function about the secret key and the leakage function about the hash list are unified. Clearly, the two kinds of leakage functions in our leakage model are not only unified, but also more powerful than the above case. Because the leakage function $Leak$ is an efficiently computable leakage function which is more powerful than the simpler leakage function f.

To sum up, the leakage pattern about the secret key and the leakage pattern about the hash functions in our new model are unified.

We call our new model *Continual Key Leakages and Hash Function Leakages Model* (KHLM for short). As an example, we consider a public key encryption scheme which achieves IND-CCA security in KHLM. Similarly, we can define a IND-CPA secure public key encryption scheme, a signature scheme which is existentially unforgeable under an adaptive chosen-message attack etc. in KHLM. A public key encryption scheme in KHLM consists of the following algorithms:

- **KeyGen**(1^k)**:** Takes as input the security parameter k and outputs a public key PK, a secret key SK (denoted by SK_0) and an update key UK.
- **Encrypt**(PK, M)**:** The input is a public key PK and a message M. The output is a ciphertext CT.
- **Decrypt**(SK_i, CT)**:** The input is a secret key SK_i and a ciphertext CT. The output is a decrypted message M.
- **Update**(UK, SK_i)**:** The input is an update key UK and an old secret key SK_i. The output is an updated secret key SK_{i+1}.

Note that the output of **Update**(UK, SK_i) (i.e. SK_{i+1}) and SK_i are corresponding to the same public key PK. This means that for a ciphertext CT which is encrypted by PK ($CT =$**Encrypt**(PK, M)), we have

$$\textbf{Decrypt}(SK_i, CT)=\textbf{Decrypt}(SK_{i+1}, CT)=M.$$

Moreover, the size of the secret key should be unchanged after the update process (i.e. $|SK_i| = |SK_{i+1}|$). The secret key space of a public key should be large enough. Let $L(k)$ be a function of the security parameter and $L(k) \ll |SK_i|$.

Our new model is as follows. In Definition 2, we simply assume the scheme Π uses one hash function. For a scheme uses more than one hash function, our new model allows the hash lists of all hash functions can be leaked similarly.

Definition 2. *We say that a public key encryption scheme Π is $L(k)$-IND-CCA secure in KHLM if for any probabilistic polynomial time adversary $\mathcal{A}$, it holds that*

$$Adv_{\Pi,\mathcal{A}}^{LCCA}(k) = \left| Pr[Expt_{\Pi,\mathcal{A}}^{LCCA}(0) = 1] - Pr[Expt_{\Pi,\mathcal{A}}^{LCCA}(1) = 1] \right|$$

is negligible in k, where $Expt_{\Pi,\mathcal{A}}^{LCCA}(b)$ is defined as follows:

- *Let $\mathcal{L}_H$ denotes the hash list of a hash function H used by Π. Initialization: $\mathcal{L}_H \leftarrow \perp$*
- *Challenger chooses $(PK, UK, SK_0) \leftarrow KeyGen(1^k)$ and sends PK to $\mathcal{A}$.*
- *The adversary $\mathcal{A}$ may ask for the following four queries:*
  Leakage query: *Each such query consists of an efficiently computable leakage function $Leak : \{0,1\}^{|SK|} \rightarrow \{0,1\}^{L(k)}$ with $L(k)$ bits output. On the i^{th} such query with $Leak_i$, the challenger gives the value $Leak_i(SK_i)$ to $\mathcal{A}$ and computes the updated secret key $SK_{i+1} \leftarrow Update(UK, SK_i)$.*
  Hash query: *For a hash query a_i to H, behave as follows:*
  If $a_i \in \mathcal{L}_H$, then find b_i corresponding to a_i from $\mathcal{L}_H$ and output b_i to $\mathcal{A}$. If $a_i \notin \mathcal{L}_H$, then choose b_i randomly, add pair (a_i, b_i) to $\mathcal{L}_H$ and output b_i to $\mathcal{A}$.
  Leak hash query: *For a leak hash query to H, output all contents of the hash list $\mathcal{L}_H$ to $\mathcal{A}$.*
  Decryption query: *For a decryption query with a ciphertext CT, decrypt CT with the current secret key SK_i and output $Decrypt(SK_i, CT)$ to $\mathcal{A}$.*
- *At some point $\mathcal{A}$ gives the challenger two messages M_0, M_1 and $|M_0| = |M_1|$. The challenger computes $CT^* \leftarrow Encrypt(PK, M_b)$. Then the challenger sends CT^* to $\mathcal{A}$.*
- *The adversary $\mathcal{A}$ can not ask the leakage query after he gets CT^*. The adversary $\mathcal{A}$ can also ask the hash query, the leak hash query, and the decryption query. But he can not ask the decryption query with CT^*.*
- *The adversary $\mathcal{A}$ outputs a bit b'. If $b' = b$, then the experiment outputs 1, otherwise, the experiment outputs 0.*

In KHLM, we assume the update process of the secret key is leak-free. We note that this security property may be motivated in practice, by thinking of a device storing the secret key SK_i as being used "in the field" for some amount of time, during which the secret key can leak up to $L(k)$ bits, but then always being brought back to a "secure base" where the secret key SK_i is updated to SK_{i+1} using an update key UK which is stored securely and externally. In [12] and [33], the same assumption was introduced.

Note that, the adversary in KHLM is not allowed to ask the leakage query after the challenge phase. This restriction is necessary as many other leakage models [2,10,11,13,20,30]: the adversary can encode the decryption algorithm, the challenge ciphertext, and the to messages M_0 and M_1 into a leakage function that outputs the bit b. Additionally, in KHLM, we assume the randomness used by the challenger to compute $CT^* \leftarrow Encrypt(PK, M_b)$ can not be leaked to the adversary even if a part of information of it. Otherwise, the adversary can break the security easily[1] and no leakage resilient cryptographic scheme can be designed in this leakage model. As far as we know, no leakage model allows leakages from this randomness occur.

In KHLM, the adversary can get not only leakages from hash functions, but also continual leakages from the secret key. Hence, our new model is more universal and stronger than both LROM and some leakage models in [2,10,13] (In KHLM, the adversary asks the leakage query non-adaptively (See [10] for the definition and more details.). For a chosen-plaintext attack adversary, the security definition of a public key encryption scheme in our new model is equivalent to a security definition in which the adversary can ask the leakage query adaptively [10]. However, it is not clear that whether the same equivalence holds when we extend the definition to consider a chosen-ciphertext attack adversary. If the equivalence holds, KHLM is also stronger than the leakage models in [2,10,13] where a chosen-ciphertext attack adversary is considered.).

In the next section, we will present a public key encryption scheme which is $L(k)$-IND-CCA secure in our new model.

4 A Provably Secure Public Key Encryption Scheme in Our New Model

In this section, we first introduce our public key encryption scheme in KHLM and then prove the security of it. Our public key encryption scheme in KHLM is denoted by PKE and is based on Cramer-Shoup cryptosystem. The PKE is shown in the following.

KeyGen: The key generation algorithm runs as follows. It generates a k bit prime q where k is the security parameter. Let $\mathbb{G}$ is a group of prime order q. The generator of $\mathbb{G}$ is g. It chooses $\boldsymbol{A_1}, \boldsymbol{A_2}$ uniformly and independently at random from $\mathbb{Z}_q^{n\times(n-1)}$ (denoted by $\boldsymbol{A_1}, \boldsymbol{A_2} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{n\times(n-1)}$) and two random vectors $\boldsymbol{t} = (t_1, t_2, \ldots, t_n), t_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n$ and $\boldsymbol{s} = (s_1, s_2, \ldots, s_n), s_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n$ satisfying $ker(\boldsymbol{t}) = \mathsf{colspan}(\boldsymbol{A_1})$ and $ker(\boldsymbol{s}) = \mathsf{colspan}(\boldsymbol{A_2})$. This requirement can be satisfied easily without negligible probability. Let $t = \sum_{i=1}^n t_i \bmod q$ and $s = \sum_{i=1}^n s_i \bmod q$ and $g_1 = g^t, g_2 = g^s$. It generates five vectors $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}$ uniformly and independently at random from $\mathbb{Z}_q^n$. Let x_1, x_2, y_1, y_2, z be five numbers in $\mathbb{Z}_q$. The five numbers satisfy $\langle \boldsymbol{t}, \boldsymbol{x_1}\rangle \bmod q = tx_1 \bmod q$, $\langle \boldsymbol{s}, \boldsymbol{x_2}\rangle \bmod q = sx_2 \bmod q$, $\langle \boldsymbol{t}, \boldsymbol{y_1}\rangle \bmod q = ty_1 \bmod q$, $\langle \boldsymbol{s}, \boldsymbol{y_2}\rangle \bmod q = sy_2 \bmod q$, and $\langle \boldsymbol{t}, \boldsymbol{z}\rangle \bmod q = tz \bmod q$. The group elements $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$ are computed. A hash function H is chosen from the family of universal one-way hash functions.

[1] For example, the adversary can obtain a part of information of Encrypt$(PK, M_0; r)$ (r is the randomness.) which can be used to determine b.

It chooses $\boldsymbol{\beta_1}, \boldsymbol{\beta_3}, \boldsymbol{\beta_5} \in ker(\boldsymbol{t})$ and $\boldsymbol{\beta_2}, \boldsymbol{\beta_4} \in ker(\boldsymbol{s})$ uniformly and independently at random. Let matrix $UP = [\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta_3}, \boldsymbol{\beta_4}, \boldsymbol{\beta_5}]^\top$ denotes a $n \times 5$ matrix. The public key PK is $(g^{\boldsymbol{t}}, g^{\boldsymbol{s}}, c, d, h, H)$. The secret key SK (i.e. SK_0) is a $n \times 5$ matrix and $SK = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^\top + UP$. The update key UK is $(\boldsymbol{t}, \boldsymbol{s})$.

Remark 1. *The values $t, s, x_1, x_2, y_1, y_2, z$ should be deleted after the key generation algorithm so that the adversary can not obtain them by attacks caused by sloppy usage or implementation.*
Remark 2. *For convenience, we use the same symbol $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^\top$ to denote every secret key $SK_i, (i = 0, 1, 2, \ldots)$. Note that, for any $i \neq j$, we have $SK_i \neq SK_j$ except negligible probability.*

Encrypt: The encryption algorithm runs as follows. For an input message $M \in \mathbb{G}$, it chooses $r \in \mathbb{Z}_p$ at random. It computes $u_1 = g^{r\langle \boldsymbol{t}, \boldsymbol{1_n} \rangle} = g_1^r$, $u_2 = g^{r\langle \boldsymbol{s}, \boldsymbol{1_n} \rangle} = g_2^r$, $e = h^r M$, $\alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$. It outputs a ciphertext $(g^{r\boldsymbol{t}}, g^{r\boldsymbol{s}}, e, v)$.

Decrypt: The decryption algorithm runs as follows. Given a ciphtertext $(g^{r\boldsymbol{t}}, g^{r\boldsymbol{s}}, e, v)$, it computes $u_1 = g^{\langle r\boldsymbol{t}, \boldsymbol{1_n} \rangle}$, $u_2 = g^{\langle r\boldsymbol{s}, \boldsymbol{1_n} \rangle}$, $\alpha = H(u_1, u_2, e)$ and verifies whether $g^{\langle r\boldsymbol{t}, \boldsymbol{x_1} \rangle + \alpha \langle r\boldsymbol{t}, \boldsymbol{y_1} \rangle + \langle r\boldsymbol{s}, \boldsymbol{x_2} \rangle + \alpha \langle r\boldsymbol{s}, \boldsymbol{y_2} \rangle} = v$ holds or not by using $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}]$. If the verification holds, then it outputs the message $M = e / g^{\langle r\boldsymbol{t}, \boldsymbol{z} \rangle}$ by using $\boldsymbol{z}$. Else if, it rejects the decryption as an invalid ciphertext $\perp$.

Update: The update algorithm runs as follows. Given an old secret key $SK_i = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^\top$, it chooses $\boldsymbol{\beta_1}, \boldsymbol{\beta_3}, \boldsymbol{\beta_5} \in ker(\boldsymbol{t})$ and $\boldsymbol{\beta_2}, \boldsymbol{\beta_4} \in ker(\boldsymbol{s})$ uniformly and independently at random. Let matrix $UP = [\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta_3}, \boldsymbol{\beta_4}, \boldsymbol{\beta_5}]^\top$ be a $n \times 5$ matrix. The new updated secret key is $SK_{i+1} = SK_i + UP$. It outputs SK_{i+1}.

Since we have $g^{\langle r\boldsymbol{t}, \boldsymbol{x_1} \rangle + \langle r\boldsymbol{s}, \boldsymbol{x_2} \rangle} = g_1^{rx_1} g_2^{rx_2} = c^r$, $g^{\langle r\boldsymbol{t}, \boldsymbol{y_1} \rangle + \langle r\boldsymbol{s}, \boldsymbol{y_2} \rangle} = g_1^{ry_1} g_2^{ry_2} = d^r$, and $g^{\langle r\boldsymbol{t}, \boldsymbol{z} \rangle} = g_1^{rz} = h^r$. The test performed by the decryption algorithm will pass and the output will be $e/h^r = M$. Therefore, the correctness of PKE can be verified. Second, we verify that the updated secret key can also decrypt a ciphertext correctly. For example, let's consider the vector $\boldsymbol{x_1}$. It is clear that $g_1^{\langle \boldsymbol{t}, \boldsymbol{x_1} + \boldsymbol{\beta_1} \rangle} = g^{\langle \boldsymbol{t}, \boldsymbol{x_1} \rangle + \langle \boldsymbol{t}, \boldsymbol{\beta_1} \rangle} = g^{\langle \boldsymbol{t}, \boldsymbol{x_1} \rangle}$, because $\boldsymbol{\beta_1} \in ker(\boldsymbol{t})$. Similarly, $\boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}$ can be updated correctly.

Theorem 2 establishes the security of the scheme PKE. We show the proof of Theorem 2 in Appendix B.

Theorem 2. *If the hash function H is chosen from a family of universal one-way hash functions and the GDDH assumption of the group $\mathbb{G}$ holds, then the PKE is $L(k)$-IND-CCA secure in KHLM, as long as $L(k) < (n-4)log(q) - \omega(log(k))$.*

Our scheme PKE is much stronger than the scheme in Section 4 of the paper [2] because our scheme can tolerate more leakages and has higher security level. If the equivalence of adaptive leakages and non-adaptive leakages holds for a chosen-ciphertext attack adversary, our scheme *PKE* is still better than the scheme in Section 6.3 of the paper [3] because our scheme can tolerate continual leakages from the secret key and has higher tolerance leakage rate.

5 Conclusion and Future Work

In this paper, we introduce a new leakage model in which both the secret key and the hash lists of hash functions can be leaked. Moreover, the secret key can be leaked continually and refreshed. Therefore, our new model is more universal and stronger than the LROM and some other leakage models [2,10,13]. We also present a new public key encryption scheme *PKE* which is $L(k)$-IND-CCA secure in this new model. In future work, one may try to consider additional leakages from the key generation process and/or the update process. Leakage resilient signature scheme in KHLM is also expected.

References

1. M. Bellare, P. Rogaway.: Random Oracles are Practical: A Paradigm for Designing Eifficient Protocols. ACM Conference on Computer and Communications Security 1993. pp.62-73, 1993.
2. M. Naor, G. Segev.: Public-Key Cryptosystems Resilient to Key Leakage. CRYPTO2009, LNCS 5677, pp.18-35, 2009. (See [3] for the full version.)
3. M. Naor, G. Segev.: Public-Key Cryptosystems Resilient to Key Leakage. Cryptology ePrint Archive, Report 2009/105 (2009).
4. J.A. Halderman, S.D. Schoen, H. Nadia, W. Clarkson, W. Paul, JA. Calandrino, AJ. Feldman, J. Appelbaum, and EW. Felten.: Lest We Remember: Cold-Boot Attacks on Encryption Keys. 17th USENIX Security Symposium,pp.45-60,2008.
5. Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta.: Leaky Random Oracle (Extended Abstract). ProvSec 2008, LNCS 5324, pp.226-240, 2008.
6. P. Kocher, J. Jaffe, and B. Jun.: Differential Power Analysis. CRYPTO1999, LNCS 1666, PP.388-397, 1999.
7. K. Gandol, C. Mourtel, and F. Olivier.: Electromagnetic Analysis: Concrete Results. CHES2001, LNCS 2162, pp.251-261, 2001.
8. Paul C. Kocher.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO1996, LNCS 1109, pp.104-113, 1996.
9. R. Cramer, V. Shoup.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. CRYPTO1998, LNCS 1462, pp.13-25, 1998.
10. A. Akavia, S. Goldwasser, and V. Vaikuntanathan.: Simultaneous hardcore bits and cryptography against memory attacks. TCC2009, LNCS 5444, pp.474-495, 2009.
11. Z. Brakerski, Y.T. Kalai, J. Katz, and V. Vaikuntanathan.: Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage. FOCS2010, pp.501-510, 2010.
12. S. Agrawal, Y. Dodis, V. Vaikuntanathan, and D. Wichs.: On Continual Leakage of Discrete Log Representations. IACR Eprint Archive Report 2012/367.
13. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs.: Efficient Public-Key Cryptography in the Presence of Key Leakage. ASIACRYPT2010, LNCS 6477, pp.613-631, 2010.

14. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing, 38(1):97-139, 2008.
15. S. Dziembowski, S. Faust.: Leakage-Resilient Cryptography from the Inner-Product Extractor. ASIACRYPT2011, LNCS 7073, pp.702-721, 2011.
16. S. Halevi, H. Lin.: After-the-Fact Leakage in Public-Key Encryption. TCC2011, LNCS 6597, pp.107-124, 2011.
17. M. Bellare, P. Rogaway.: Optimal Asymmetric Encryption. EUROCRYPT1994, LNCS 950, pp.92-111, 1995.
18. K. Kurosawa, Y. Desmedt.: A New Paradigm of Hybrid Encryption Scheme. CRYPTO2004, LNCS 3152, pp.426-442, 2004.
19. B. LaMacchia, K. Lauter, and A. Mityagin.: Stronger Security of Authenticated Key Exchange. Provsec2007, pp.1-16, 2007.
20. A. Lewko, M. Lewko, and B. Waters.: How to Leak on Key Updates. STOC2011, pp.725-734.
21. R.L. Rivest.: All-or-Nothing Encryption and the Package Transform. FSE1997, LNCS 1267, pp.210-218, 1997.
22. V. Boyko.: On the Security Properties of OAEP as an All-or-Nothing Transform. CRYTO1999, LNCS 1666, pp.503-518, 1999.
23. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai.: Exposure-Resilient Functions and All-or-Nothing Transforms. EUROCRYPT2000, LNCS 1807, pp.453-469, 2000.
24. Y. Dodis, A. Sahai, and A. Smith.: On Perfect and Adaptive Security in Exposure-Resilient Cryptography. EUROCRYPT2001, LNCS 2045, pp.301-324, 2001.
25. Y. Ishai, A. Sahai, and D. Wagner.: Private circuits: Securing hardware against Probing attacks. CRYPTO2003, LNCS 2729, pp.463-481, 2003.
26. S. Micali, L. Reyzin.: Physically Observable Cryptography (Extented Abstract). TCC2004, LNCS 2951, pp.278-296, 2004.
27. Y. Dodis, Y.T. Kalai, and S. Lovett.: On Cryptography with Auxiliary Input. STOC2009, pp.621-630, 2009.
28. J. Katz, V. Vaikuntanathan.: Signature Schemes with Bounded Leakage Resilience. ASIACRYPT2009, LNCS 5912, pp.703-720, 2009.
29. Y. Dodis, S. Goldwasser, Y.T. Kalai, C. Peikert, V. Vaikuntanathan.: Public-Key Encryption Schemes with Auxiliary Inputs.
30. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs.: Cryptography Against Continuous Memory Attacks. FOCS2010, pp.511-520, 2010.
31. E. Kiltz, K. Pietrzak.: Leakage Resilient ElGamal Encryption. ASIACRYPT2010, LNCS 6477, pp.595-612, 2010.
32. S. Dziembowski, K. Pietrzak.: Leakage-Resilient Cryptography. FOCS2008, pp.293-302, 2008.
33. D. Wichs.: Cryptographic Resilience to Continual Information Leakage. PHD Thesis.

Appendix A: The Proof of Theorem 1

Proof. We will prove Theorem 1 by the following two claims.
Claim 1.1 *The GDDH assumption implies the DDH assumption.*

Proof. Let $\mathcal{A}$ be an adversary who can break the DDH assumption. We can construct an adversary $\mathcal{B}$ who can break the GDDH assumption using $\mathcal{A}$. The adversary $\mathcal{B}$ is as follows. When $\mathcal{B}$ gets an input ensemble S_1:

$$\{\mathbb{G}, \{g_1, \ldots, g_n\}, \{g_{n+1}, \ldots, g_{2n}\}, \{g_1^r, \ldots, g_n^r\}, \{g_{n+1}^r, \ldots, g_{2n}^r\}\},$$

he sends $\{\mathbb{G}, g_1, g_{n+1}, g_1^r, g_{n+1}^r\}$ to $\mathcal{A}$ and runs $\mathcal{A}$ as a subroutine. When $\mathcal{A}$ outputs $b \in \{0, 1\}$, then $\mathcal{B}$ outputs b. When $\mathcal{B}$ gets an input ensemble S_2:

$$\{\mathbb{G}, \{g_1, \ldots, g_n\}, \{g_{n+1}, \ldots, g_{2n}\}, \{g_1^{r_1}, \ldots, g_n^{r_1}\}, \{g_{n+1}^{r_2}, \ldots, g_{2n}^{r_2}\}\},$$

he sends $\{\mathbb{G}, g_1, g_{n+1}, g_1^{r_1}, g_{n+1}^{r_2}\}$ to $\mathcal{A}$ and runs $\mathcal{A}$ as a subroutine. When $\mathcal{A}$ outputs $b \in \{0, 1\}$, then $\mathcal{B}$ outputs b. Clearly, we have

$$\Pr[\mathcal{B}(S_1) = 1] = \Pr[\mathcal{A}(\mathbb{G}, g_1, g_{n+1}, g_1^r, g_{n+1}^r) = 1],$$

$$\Pr[\mathcal{B}(S_2) = 1] = \Pr[\mathcal{A}(\mathbb{G}, g_1, g_{n+1}, g_1^{r_1}, g_{n+1}^{r_2}) = 1].$$

Due to $\mathcal{A}$ can break the DDH assumption, then $\mathcal{B}$ can break the GDDH assumption. Therefore, Claim 1.1 holds. □

Claim 1.2 *The DDH assumption implies the GDDH assumption.*

Proof. Let $\mathcal{A}$ be an adversary who can break the GDDH assumption. We can construct an adversary $\mathcal{B}$ who can break the DDH assumption using $\mathcal{A}$. The adversary $\mathcal{B}$ is as follows. When $\mathcal{B}$ gets an input ensemble $\{\mathbb{G}, g_1, g_2, g_1^r, g_2^r\}$ (r is chosen uniformly at random from $\mathbb{Z}_q$), he chooses $a_i, b_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n-1$ independently and uniformly at random and computes $\eta_1 = g_1, \eta_i = g_1^{a_{i-1}}, \eta_i^r = g_1^{ra_{i-1}}, \eta_{n+1} = g_2, \eta_{n+i} = g_2^{b_{i-1}}, \eta_{n+1}^r = g_2^r, \eta_{n+i}^r = g_2^{rb_{i-1}}, i = 2, \ldots, n$. Thus, $\mathcal{B}$ has the ensemble S_1:

$$\{\mathbb{G}, \{\eta_1, \ldots, \eta_n\}, \{\eta_{n+1}, \ldots, \eta_{2n}\}, \{\eta_1^r, \ldots, \eta_n^r\}, \{\eta_{n+1}^r, \ldots, \eta_{2n}^r\}\}$$

and sends it to the adversary $\mathcal{A}$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. When $\mathcal{A}$ outputs $b \in \{0, 1\}$, then $\mathcal{B}$ outputs b. Similarly, when $\mathcal{B}$ gets an input ensemble $\{\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2}\}$ (r_1, r_2 are chosen uniformly at random from $\mathbb{Z}_q$), he chooses $a_i, b_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n-1$ independently and uniformly at random and computes $\eta_1 = g_1, \eta_i = g_1^{a_{i-1}}, \eta_i^{r_1} = g_1^{r_1 a_{i-1}}, \eta_{n+1} = g_2, \eta_{n+i} = g_2^{b_{i-1}}, \eta_{n+1}^{r_2} = g_2^{r_2}, \eta_{n+i}^{r_2} = g_2^{r_2 b_{i-1}}, i = 2, \ldots, n$. Thus, $\mathcal{B}$ has the ensemble S_2:

$$\{\mathbb{G}, \{\eta_1, \ldots, \eta_n\}, \{\eta_{n+1}, \ldots, \eta_{2n}\}, \{\eta_1^{r_1}, \ldots, \eta_n^{r_1}\}, \{\eta_{n+1}^{r_2}, \ldots, \eta_{2n}^{r_2}\}\}$$

and sends it to the adversary $\mathcal{A}$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. When $\mathcal{A}$ outputs $b \in \{0, 1\}$, then $\mathcal{B}$ outputs b. Clearly, we have

$$\Pr[\mathcal{B}(\mathbb{G}, g_1, g_{n+1}, g_1^r, g_{n+1}^r) = 1] = \Pr[\mathcal{A}(S_1) = 1],$$

$$\Pr[\mathcal{B}(\mathbb{G}, g_1, g_{n+1}, g_1^{r_1}, g_{n+1}^{r_2}) = 1] = \Pr[\mathcal{A}(S_2) = 1].$$

Due to $\mathcal{A}$ can break the GDDH assumption, it is clearly that $\mathcal{B}$ can break the DDH assumption. Therefore, the Claim 1.2 holds. □

This concludes the proof of the theorem. □

Appendix B: The Proof of Theorem 2

Proof. We define a new experiment $Expt_{\Pi,\mathcal{A}}^{RLCCA}(b)$ for a public key encryption Π and any probabilistic polynomial time adversary $\mathcal{A}$. The experiment $Expt_{\Pi,\mathcal{A}}^{RLCCA}(b)$ is identical to the experiment $Expt_{\Pi,\mathcal{A}}^{LCCA}(b)$ except that the challenger chooses random numbers (denoted by UR_i) which has the same size as the secret key and sends $Leak_i(UR_i)$ to the adversary in every leakage query. The real secret key SK is updated normally. For our scheme PKE, it holds that

$$Adv_{\mathsf{PKE},\mathcal{A}}^{LCCA}(k) = \left|\Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{LCCA}(0)=1] - \Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{LCCA}(1)=1]\right|$$

$$\leq \left|\Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{LCCA}(0)=1] - \Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{RLCCA}(0)=1]\right|$$

$$+\left|\Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{RLCCA}(0)=1] - \Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{RLCCA}(1)=1]\right|$$

$$+\left|\Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{RLCCA}(1)=1] - \Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{LCCA}(1)=1]\right|.$$

We will prove this theorem by the following three claims.

Claim 2.1 *If the GDDH assumption of the group $\mathbb{G}$ holds and $L(k) < (n-4)log(q) - \omega(log(k))$, it holds that*

$$\left|\Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{LCCA}(0)=1] - \Pr[Expt_{\mathsf{PKE},\mathcal{A}}^{RLCCA}(0)=1]\right| < \mu_1(k)$$

, where $\mu_1(k)$ is negligible in k.

Proof. By the following lemma, as long as $L(k) < (n-4)log(q) - \omega(log(k))$, the leakages from the real secret key SK_i is distinguishable with the leakages from UR_i for any leakage function $Leak_i$.

Lemma 3 *Let $n \geq m$, $m \geq l_1$, and $m \geq l_2$ be integers. Let $Leak : \{0,1\}^* \to \{0,1\}^{L(k)}$ be some arbitrary function and let $\boldsymbol{X_1} \in \mathbb{Z}_q^{n\times l_1}$, $\boldsymbol{X_2} \in \mathbb{Z}_q^{n\times l_2}$ be arbitrary matrixes. For randomly sampled $\boldsymbol{A_1} \xleftarrow{*} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{E_1} \xleftarrow{*} \mathbb{Z}_q^{m\times l_1}$, $\boldsymbol{UR_1} \xleftarrow{*} \mathbb{Z}_q^{n\times l_1}$, $\boldsymbol{A_2} \xleftarrow{*} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{E_2} \xleftarrow{*} \mathbb{Z}_q^{m\times l_2}$, $\boldsymbol{UR_2} \xleftarrow{*} \mathbb{Z}_q^{n\times l_2}$ we have:*

$$(Leak(\boldsymbol{A_1E_1} + \boldsymbol{X_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}) \overset{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{UR_2}), \boldsymbol{A_1}, \boldsymbol{A_2}),$$

as long as $(m - max\{l_1, l_2\})log(q) - L(k) = \omega(log(k))$, $n = poly(k)$, and $q = k^{\omega(1)}$.

Proof. This lemma can be proved based on Corollary 8 in the paper of S. Agrawal et al. [12]. We prove the lemma by the following two lemmas.

Lemma 4 *Let $n \geq m$, $m \geq l_1$, and $m \geq l_2$ be integers. Let $Leak : \{0,1\}^* \to \{0,1\}^{L(k)}$ be some arbitrary function and let $\boldsymbol{X_1} \in \mathbb{Z}_q^{n\times l_1}$, $\boldsymbol{X_2} \in \mathbb{Z}_q^{n\times l_2}$ be arbitrary matrixes. For randomly sampled $\boldsymbol{A_1} \xleftarrow{*} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{E_1} \xleftarrow{*} \mathbb{Z}_q^{m\times l_1}$, $\boldsymbol{UR_1} \xleftarrow{*} \mathbb{Z}_q^{n\times l_1}$, $\boldsymbol{A_2} \xleftarrow{*} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{E_2} \xleftarrow{*} \mathbb{Z}_q^{m\times l_2}$ we have:*

$$(Leak(\boldsymbol{A_1E_1}+\boldsymbol{X_1}, \boldsymbol{A_2E_2}+\boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}) \overset{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{A_2E_2}+\boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}),$$

as long as $(m-l_1)log(q) - L(k) = \omega(log(k))$, $n = poly(k)$, *and* $q = k^{\omega(1)}$.
Proof. We first prove that

$$(Leak(\boldsymbol{A_1E_1} + \boldsymbol{X_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}) \stackrel{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}).$$

Now we assume that there is some function $Leak$ and an (unbounded) distinguisher D that has a non-negligible distinguishing advantage for the two distributions

$$(Leak(\boldsymbol{A_1E_1} + \boldsymbol{X_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}) \stackrel{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}).$$

Then we can define a function $Leak'$ and a distinguisher D' which breaks the problem of Corollary 8 in [12]. The matrixes $\boldsymbol{X_2}$, $\boldsymbol{A_2}$, and $\boldsymbol{E_2}$ are chosen uniformly and independently at random and satisfy the requirement of Lemma 4. Let $\boldsymbol{A_2E_2} + \boldsymbol{X_2}$ be a fixed matrix. Given $\boldsymbol{C} = \boldsymbol{A_1E_1} + \boldsymbol{X_1}$ or $\boldsymbol{C} = \boldsymbol{UR_1}$, the function $Leak'$ outputs $ans := Leak(\boldsymbol{C}, \boldsymbol{A_2E_2} + \boldsymbol{X_2})$. The distinguisher D' is given $(ans, \boldsymbol{A_1})$ and outputs $D(ans, \boldsymbol{A_1})$. The distinguisher D' has the same distinguishing advantage as D. Therefore, indistinguishability holds as long as $L(k)$ satisfies the requirement. Then, using the fact that applying the same function to two distributions cannot increase their statistical distance, we obtain

$$(Leak(\boldsymbol{A_1E_1}+\boldsymbol{X_1}, \boldsymbol{A_2E_2}+\boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}) \stackrel{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{A_2E_2}+\boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}).$$

□

Similarly, we can prove the following lemma and we ignore the proof here due to space reason.

Lemma 5 *Let* $n \geq m$, $m \geq l_1$, *and* $m \geq l_2$ *be integers. Let* $Leak : \{0,1\}^* \rightarrow \{0,1\}^{L(k)}$ *be some arbitrary function and let* $\boldsymbol{X_2} \in \mathbb{Z}_q^{n\times l_2}$ *be an arbitrary matrix. For randomly sampled* $\boldsymbol{UR_1} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{n\times l_1}$, $\boldsymbol{A_1} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{A_2} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{n\times m}$, $\boldsymbol{E_2} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{d\times l_2}$, $\boldsymbol{UR_2} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{n\times l_2}$ *we have:*

$$(Leak(\boldsymbol{UR_1}, \boldsymbol{A_2E_2} + \boldsymbol{X_2}), \boldsymbol{A_1}, \boldsymbol{A_2}) \stackrel{s}{\approx} (Leak(\boldsymbol{UR_1}, \boldsymbol{UR_2}), \boldsymbol{A_1}, \boldsymbol{A_2}),$$

as long as $(m-l_2)log(q) - L(k) = \omega(log(k))$, $n = poly(k)$, *and* $q = k^{\omega(1)}$.

From Lemma 4 and Lemma 5, we can see that Lemma 3 holds. □

Note that, in our scheme, the matrixes $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ in the key generation algorithm are chosen uniformly and independently at random from $\mathbb{Z}_q^{n\times(n-1)}$. In the update algorithm, the old secret key $SK_i = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^\top$. The new secret key SK_{i+1} is generated by $\boldsymbol{A_1E_1} + [\boldsymbol{x_1}, \boldsymbol{y_1}, \boldsymbol{z}]^\top$ and $\boldsymbol{A_2E_2} + [\boldsymbol{x_2}, \boldsymbol{y_2}]^\top$, where $\boldsymbol{E_1} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{m\times l_1}$ and $\boldsymbol{E_2} \stackrel{*}{\leftarrow} \mathbb{Z}_q^{m\times l_2}$. Furthermore, similar to the situation of Cramer-Shoup cryptosystem, the secret key can not be leaked from the decryption query except negligible probability and can not be obtained from the public information and leakages from hash lists. Therefore, by Lemma 3, the leakages from the real secret key SK_i and the leakages from a random matrix UR_i in $\mathbb{Z}_p^{n\times 5}$ can not be distinguished as long as $L(k) < (n-4)log(q) - \omega(log(k))$, $(i = 0, 1, \ldots)$. In this way, Claim 2.1 is proved.□

Claim 2.2 *If the GDDH assumption of the group* $\mathbb{G}$ *holds, we have*

$$\left|\Pr[Expt^{RLCCA}_{\mathsf{PKE},\mathcal{A}}(0)=1]-\Pr[Expt^{RLCCA}_{\mathsf{PKE},\mathcal{A}}(1)=1]\right|<\mu_2(k)$$

, where $\mu_2(k)$ *is negligible in* k*.*

For space reason, the details of the proof of Claim 2.2 is shown in Appendix C. The crucial reasons of success of the proof are as follows. On one hand, the leakage query in the two experiments does not leak the real secret key SK_i and only leaks UR_i. For a random matrix $UR_i=[\boldsymbol{ur_1},\ldots,\boldsymbol{ur_5}]^\top$, the probability of $\langle \boldsymbol{t},\boldsymbol{ur_1}\rangle \bmod q = tx_1 \bmod q$ equals to $1/q$ which is negligible in k. Therefore, the adversary obtains even a part of leakage information about the real secret key with negligible probability $1-(1-1/q)^5$. On the other hand, the leaky hash query in KHLM cannot be advantage of the adversary. □

Claim 2.3 *If the GDDH assumption of the group* $\mathbb{G}$ *holds and* $L(k)<(n-4)log(q)-\omega(log(k))$*, it holds that*

$$\left|\Pr[Expt^{LCCA}_{\mathsf{PKE},\mathcal{A}}(1)=1]-\Pr[Expt^{RLCCA}_{\mathsf{PKE},\mathcal{A}}(1)=1]\right|<\mu_3(k)$$

, where $\mu_3(k)$ *is negligible in* k*.*

Proof. The proof of Claim 2.3 is similar to the proof of Claim 2.1. □

Therefore, our new scheme PKE is $L(k)$-IND-CCA secure in KHLM. □

Appendix C: The Proof of Claim 2.2

Proof. Assume that $Adv^{RLCCA}_{\mathsf{PKE},\mathcal{A}}(k)$ is non-negligible and the hash family is universal one-way. Then there exists an adversary $\mathcal{A}$ that can break the scheme PKE. We will show how to use the adversary $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the GDDH assumption. Define the set $\mathbf{D}$ as follows $\{(\{g_1,\ldots,g_n\},\{g_{n+1},\ldots,g_{2n}\},\{g_1^r,\ldots,g_n^r\},\{g_{n+1}^r,\ldots,g_{2n}^r\})|g_1,\ldots,g_{2n}\overset{*}{\leftarrow}\mathbb{G},r\overset{*}{\leftarrow}\mathbb{Z}_q\}$ and the set $\mathbf{R}$ as follows $\{(\{g_1,\ldots,g_n\},\{g_{n+1},\ldots,g_{2n}\},\{g_1^{r_1},\ldots,g_n^{r_1}\},\{g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}\})|g_1,\ldots,g_{2n}\overset{*}{\leftarrow}\mathbb{G},r_1,r_2\overset{*}{\leftarrow}\mathbb{Z}_q\}$. If the input of the adversary $\mathcal{B}$ comes from $\mathbf{D}$, the simulation of $\mathcal{B}$ will be nearly perfect, and so the adversary $\mathcal{A}$ will have a non-negligible advantage in guessing the hidden bit b. If the input of $\mathcal{B}$ comes from $\mathbf{R}$, then the adversary $\mathcal{A}$'s view is essentially independent of b, and therefore the adversary $\mathcal{A}$'s advantage is negligible. Therefore, $\mathcal{B}$ can distinguish $\mathbf{D}$ from $\mathbf{R}$ with non-negligible advantage which contradicts with the GDDH assumption.

We now give the details of $\mathcal{B}$. The input to $\mathcal{B}$ is

$$(\mathbb{G},\{g_1,\ldots,g_n\},\{g_{n+1},\ldots,g_{2n}\},\{g_1^{r_1},\ldots,g_n^{r_1}\},\{g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}\}).$$

The adversary $\mathcal{B}$ chooses vectors $\boldsymbol{x_1}=(x_{11},\ldots,x_{1n})\in\mathbb{Z}_q^n$, $\boldsymbol{x_2}=(x_{21},\ldots,x_{2n})\in\mathbb{Z}_q^n$, $\boldsymbol{y_1}=(y_{11},\ldots,y_{1n})\in\mathbb{Z}_q^n$, $\boldsymbol{y_2}=(y_{21},\ldots,y_{2n})\in\mathbb{Z}_q^n$, $\boldsymbol{z_1}=(z_{11},\ldots,z_{1n})\in\mathbb{Z}_q^n$, $\boldsymbol{z_2}=(z_{21},\ldots,z_{2n})\in\mathbb{Z}_q^n$ independently and uniformly at random. Then the adversary $\mathcal{B}$ computes $c=g_1^{x_{11}}g_2^{x_{12}}\cdots g_n^{x_{1n}}g_{n+1}^{x_{21}}g_{n+2}^{x_{22}}\cdots g_{2n}^{x_{2n}}$, $d=g_1^{y_{11}}g_2^{y_{12}}\cdots g_n^{y_{1n}}g_{n+1}^{y_{21}}g_{n+2}^{y_{22}}\cdots g_{2n}^{y_{2n}}$, $h=g_1^{z_{11}}g_2^{z_{12}}\cdots g_n^{z_{1n}}g_{n+1}^{z_{21}}g_{n+2}^{z_{22}}\cdots g_{2n}^{z_{2n}}$. The adversary $\mathcal{B}$ also chooses a hash function H at random. The adversary $\mathcal{B}$ sends $\{(g_1, \ldots, g_n), (g_{n+1}, \ldots, g_{2n}), c, d, h, H\}$ as the public key to $\mathcal{A}$. And the secret key is the matrix $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z_1}, \boldsymbol{z_2}]^\top$. Note that the adversary $\mathcal{B}$'s key generation algorithm is slightly different from the key generation algorithm of the actual cryptosystem. In the latter, we essentially fix $\boldsymbol{z_2} = \boldsymbol{0}$.

The adversary $\mathcal{B}$ answers the leakage query as follows: chooses $UR_i \in \mathbb{Z}_q^{n\times 5}$ uniformly at random, and sends $Leak_i(UR_i)$ to $\mathcal{A}$. Note that, the matrix UR_i is sampled uniformly at random from $\mathbb{Z}_q^{n\times 5}$. Therefore, $Leak_i(UR_i)$ leaks no information about the actual secret key $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z_1}, \boldsymbol{z_2}]^\top$ except negligible probability.

The adversary $\mathcal{B}$ can answer the hash query and the leaky hash query normally. Note that the leaky hash query in KHLM cannot be advantage of the adversary. The reason is that all input and output of the hash function H are publicly known to the adversary because a ciphertext contains (u_1, u_2, e) which is the input to the hash function.

The adversary $\mathcal{B}$ answers the decryption query as follows: For a decryption query $((g_1^{r_1'}, \ldots, g_n^{r_1'}), (g_{n+1}^{r_2'}, \ldots, g_{2n}^{r_2'}), e', v')$ from $\mathcal{A}$, asks the hash query $(g_1^{r_1'} g_2^{r_1'} \cdots g_n^{r_1'}, g_{n+1}^{r_2'} g_{n+2}^{r_2'} \cdots g_{2n}^{r_2'}, e', v')$ to H, obtains α' (the output of H) and verifies whether

$$g_1^{r_1' x_{11}} \cdots g_n^{r_1' x_{1n}} g_1^{\alpha' r_1' y_{11}} \cdots g_n^{\alpha' r_1' y_{1n}} g_{n+1}^{r_2' x_{21}} \cdots g_{2n}^{r_2' x_{2n}} g_{n+1}^{\alpha' r_2' y_{21}} \cdots g_{2n}^{\alpha' r_2' y_{2n}} = v'$$

holds or not by using $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}]$. If the verification holds, then output the message $m = e'/(g_1^{r_1' z_{11}} g_2^{r_1' z_{12}} \cdots g_n^{r_1' z_{1n}} g_{n+1}^{r_2' z_{21}} g_{n+2}^{r_2' z_{22}} \cdots g_{2n}^{r_2' z_{2n}})$ by using $[\boldsymbol{z_1}, \boldsymbol{z_2}]$. Else if, reject the decryption as an invalid ciphertext $\perp$.

When the adversary $\mathcal{B}$ obtains two message M_0 and M_1 from $\mathcal{A}$, he chooses $b \in \{0, 1\}$ at random, and computes

$$e = g_1^{r_1 z_{11}} g_2^{r_1 z_{12}} \cdots g_n^{r_1 z_{1n}} g_{n+1}^{r_2 z_{21}} g_{n+2}^{r_2 z_{22}} \cdots g_{n+2}^{r_2 z_{2n}} M_b,$$

$$\alpha = H(g_1^{r_1} g_2^{r_1} \cdots g_n^{r_1}, g_{n+1}^{r_2} g_{n+2}^{r_2} \cdots g_{n+2}^{r_2}, e),$$

$$v = g_1^{r_1 x_{11}} \cdots g_n^{r_1 x_{1n}} g_1^{\alpha r_1 y_{11}} \cdots g_n^{\alpha r_1 y_{1n}} g_{n+1}^{r_2 x_{21}} \cdots g_{2n}^{r_2 x_{2n}} g_{n+1}^{\alpha r_2 y_{21}} \cdots g_{2n}^{\alpha r_2 y_{2n}},$$

and sends $(\{g_1^{r_1}, \ldots, g_n^{r_1}\}, \{g_{n+1}^{r_2}, \ldots, g_{2n}^{r_2}\}, e, v)$ as the challenge ciphertext to $\mathcal{A}$.

Let g denotes the generator of the group $\mathbb{G}$. We know that there exist $t_i \in \mathbb{Z}_q$ such that $g_i = g^{t_i}, i = 1, \ldots, n$. There exist $s_i \in \mathbb{Z}_q$ such that $g_{n+i} = g^{s_i}, i = 1, \ldots, n$. Let $\sum_{i=1}^n t_i \bmod q = t$ and $\sum_{i=1}^n s_i \bmod q = s$, there also exist $x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{Z}_q$ such that

$$t_1 x_{11} + t_2 x_{12} + \cdots + t_n x_{1n} \equiv t x_1 \bmod q, s_1 x_{21} + s_2 x_{22} + \cdots + s_n x_{2n} \equiv s x_2 \bmod q$$
$$t_1 y_{11} + t_2 y_{12} + \cdots + t_n y_{1n} \equiv t y_1 \bmod q, s_1 y_{21} + s_2 y_{22} + \cdots + s_n y_{2n} \equiv s y_2 \bmod q$$
$$t_1 z_{11} + t_2 z_{12} + \cdots + t_n z_{1n} \equiv t z_1 \bmod q, s_1 z_{21} + s_2 z_{22} + \cdots + s_n z_{2n} \equiv s z_2 \bmod q.$$

The adversary $\mathcal{B}$ does not know $t_1, \ldots, t_n, s_1, \ldots, s_n, t, s, x_1, x_2, y_1, y_2, z_1, z_2$. However, these values are really existent. Due to the vectors $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z_1}, \boldsymbol{z_2}$ are chosen independently and uniformly at random, the values $x_1, x_2, y_1, y_2, z_1, z_2$

are chosen independently and uniformly at random from $\mathbb{Z}_q$. The adversary $\mathcal{B}$ can answer $\mathcal{A}$'s all queries correctly without knows these values.

When the input of the adversary $\mathcal{B}$ comes from $\mathbf{D}$, the challenge ciphertext is a perfectly legitimate ciphertext; however, when the input of the adversary $\mathcal{B}$ comes from $\mathbf{R}$, the challenge ciphertext will not be legitimate, in the sense that $r_1 \neq r_2$. Claim 2.2 now follows immediately from the following two lemmas.

Lemma 6 *When the adversary $\mathcal{B}$'s input comes from $\mathbf{D}$, the joint distribution of the adversary $\mathcal{A}$'s view and the hidden bit b is statistically indistinguishable from that in the actual attack.*

Proof. Consider the joint distribution of the adversary $\mathcal{A}$'s view and the bit b when the input comes from $\mathbf{D}$. In this case, the challenge ciphertext is correct, because $g_1^{rx_{11}} \cdots g_n^{rx_{1n}} g_{n+1}^{rx_{21}} \cdots g_{2n}^{rx_{2n}} = c^r$, $g_1^{ry_{11}} \cdots g_n^{ry_{1n}} g_{n+1}^{ry_{21}} \cdots g_{2n}^{ry_{2n}} = d^r$, and $g_1^{rz_{11}} \cdots g_n^{rz_{1n}} g_{n+1}^{rz_{21}} \cdots g_{2n}^{rz_{2n}} = h^r$; indeed, these equations imply that $e = h^r M_b$ and $v = c^r d^{r\alpha}$, and α itself is already of the right from. To complete the proof, we will show that the output of the decryption oracle has the right distribution. We call $C = ((g_1^{r_1'}, g_2^{r_1'}, \ldots, g_n^{r_1'}), (g_{n+1}^{r_2'}, g_{n+2}^{r_2'}, \ldots, g_{2n}^{r_2'}), e', v')$ a valid ciphertext if and only if $r_1' = r_2'$. We call C is a type 1 invalid ciphertext if $r_1' \neq r_2'$. Other possible invalid ciphertexts are called type 2 invalid ciphertext. Note that if a ciphertext is valid, with $(g_1^{r'}, g_2^{r'}, \ldots, g_n^{r'})$ and $(g_{n+1}^{r'}, g_{n+2}^{r'}, \ldots, g_{2n}^{r'})$, then $h^{r'} = g_1^{r'z_{11}} g_2^{r'z_{12}} \cdots g_n^{r'z_{1n}} g_{n+1}^{r'z_{21}} g_{n+2}^{r'z_{22}} \cdots g_{2n}^{r'z_{2n}}$; therefore, the decryption oracle outputs $e/h^{r'}$, just as it should. Consequently, the lemma follows immediately from the following:

Claim C.1 *The decryption oracle in both an actual attack against the cryptosystem and in an attack against simulator $\mathcal{B}$ rejects all invalid ciphertexts, except with negligible probability.*

Proof. We now prove this claim by considering the distribution of the point $\mathbf{P} = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, conditioned on the adversary's view. We know that there exists $w \in \mathbb{Z}_q$ such that $g^s = g^{wt}$. Let $log()$ denote $log_{g^t}()$. From the adversary's view, $\mathbf{P}$ is a random point on the plane $\mathcal{P}$ formed by intersecting the hyperplanes $log(c) = x_1 + wx_2$ (1) and $log(d) = y_1 + wy_2$ (2). These two equations come from the public key. The challenge ciphertext dose not constrain $\mathbf{P}$ any further, as the hyperplane defined by $log(v) = rx_1 + wrx_2 + \alpha ry_1 + \alpha wry_2$ (3) contains $\mathcal{P}$. Now suppose the adversary $\mathcal{A}$ submits a type 1 invalid ciphertext

$$((g_1^{r_1'}, g_2^{r_1'}, \ldots, g_n^{r_1'}), (g_{n+1}^{r_2'}, g_{n+2}^{r_2'}, \ldots, g_{2n}^{r_2'}), e', v')$$

to the decryption oracle, where $r_1' \neq r_2'$. The decryption oracle will reject, unless $\mathbf{P}$ happens to lie on the hyperplane $\mathcal{H}$ defined by $log(v') = r_1'x_1 + wr_2'x_2 + \alpha' r_1' y_1 + \alpha' w r_2' y_2$ (4) where $\alpha' = H(g_1^{r_1'} g_2^{r_1'} \cdots g_n^{r_1'}, g_{n+1}^{r_2'} g_{n+2}^{r_2'} \cdots g_{2n}^{r_2'}, e')$. Note that the equations (1), (2), and (4) are linearly independent, and so $\mathcal{H}$ intersects the plane $\mathcal{P}$ at a line. It follows that the first time the adversary submits a type 1 invalid ciphertext, the decryption oracle rejects with probability $1 - 1/q$. This rejection actually constrains the point $\mathbf{P}$, puncturing the $\mathcal{H}$ at a line. Therefore, for $i = 1, 2, \ldots$, the i^{th} invalid ciphertext submitted by the adversary will be rejected with probability at least $1 - 1/(q - i + 1)$. From this it follows that the decryption oracle rejects all type 1 invalid ciphertexts, except with negligible probability. For the case of type 2 invalid ciphertext, we can prove the lemma similarly and space doesn't permit to show the proof. □

Lemma 7 *When adversary $\mathcal{B}$'s input comes from* $\mathbf{R}$, *the distribution of the hidden bit b is (essentially) independent from the adversary $\mathcal{A}$'s view.*

Proof. The input of the adversary $\mathcal{B}$ is

$$(\{g_1,\ldots,g_n\},\{g_{n+1},\ldots,g_{2n}\},\{g_1^{r_1},\ldots,g_n^{r_1}\},\{g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}\}).$$

We may assume that $r_1 \neq r_2$, because this occurs except with negligible probability. The lemma follows immediately from the following two claims.

Claim C.2 *If the decryption oracle rejects all invalid ciphertexts during the attack, then the distribution of the hidden bit b is independent of the adversary's view.*

Proof. To see this, consider the point $\mathbf{Q} = (z_1, z_2) \in \mathbb{Z}_q^2$. At the beginning of the attack, this is a random point on the line $log(h) = z_1 + wz_2$, (5) determined by the public key. Moreover, if the decryption oracle only decrypts valid ciphertext $((g_1^{r'}, g_2^{r'}, \ldots, g_n^{r'}), (g_{n+1}^{r'}, g_{n+2}^{r'}, \ldots, g_{2n}^{r'}), e', v')$, then the adversary obtains only linearly dependent relations $r' log(h) = r'z_1 + r'wz_2$. Thus, no further information about $\mathbf{Q}$ is leaked. Consider now the challenge ciphertext sent by adversary $\mathcal{B}$ to adversary $\mathcal{A}$. We have that $e = \gamma \cdot M_b$, where $\gamma = g_1^{r_1 z_{11}} g_2^{r_1 z_{12}} \cdots g_n^{r_1 z_{1n}} g_{n+1}^{r_2 z_{21}} g_{n+2}^{r_2 z_{22}} \cdots g_{n+2}^{r_2 z_{2n}}$. Now, consider the equation $log(\gamma) = r_1 z_1 + wr_2 z_2$ (6). Clearly, equation (5) and equation (6) are linearly independent, and so the conditional distribution of γ conditioning on b and everything in the adversary's view other than e is uniform. In other words, γ is a perfect one-time pad. It follows that b is independent of the adversary $\mathcal{A}$'s view.

Claim C.3 *The decryption oracle will reject all invalid ciphertexts, except with negligible probability.*

Proof. We study the distribution of $\mathbf{P} = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, conditioned on the adversary $\mathcal{A}$'s view. From the adversary $\mathcal{A}$'s view, this is a random point on the line $\mathcal{L}$ formed by intersecting the hyperplanes (1), (2), and $log(v) = r_1 x_1 + wr_2 x_2 + \alpha r_1 y_1 + \alpha wr_2 y_2$ (7). Now assume that the adversary submits a type 1 invalid ciphertext

$$((g_1^{r'_1},\ldots,g_n^{r'_1}),(g_{n+1}^{r'_2},\ldots,g_{2n}^{r'_2}),e',v') \neq ((g_1^{r_1},\ldots,g_n^{r_1}),(g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}),e,v),$$

where $r'_1 \neq r'_2$. Let $\alpha' = H(g_1^{r'_1} \cdots g_n^{r'_1}, g_{n+1}^{r'_2} \cdots g_{2n}^{r'_2}, e')$.

There are three cases we should consider.

Case 1. $((g_1^{r'_1},\ldots,g_n^{r'_1}),(g_{n+1}^{r'_2},\ldots,g_{2n}^{r'_2}),e') = ((g_1^{r_1},\ldots,g_n^{r_1}),(g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}),e)$

In this case, the hash values are the same, but $v' \neq v$ implies that the decryption oracle will certainly reject.

Case 2. $((g_1^{r'_1},\ldots,g_n^{r'_1}),(g_{n+1}^{r'_2},\ldots,g_{2n}^{r'_2}),e') \neq ((g_1^{r_1},\ldots,g_n^{r_1}),(g_{n+1}^{r_2},\ldots,g_{2n}^{r_2}),e)$ and $\alpha' \neq \alpha$.

The decryption oracle will reject unless the point $\mathbf{P}$ lies on the hyperplane $\mathcal{H}$ defined by (4). However, the equations (1), (2), (7), and (4) are linearly independent. Thus, $\mathcal{H}$ intersects the line $\mathcal{L}$ at a point, from which it follows (as in the proof of Lemma 4) that the decryption oracle rejects, except with negligible probability.

Case 3. $((g_1^{r_1'}, \ldots, g_n^{r_1'}), (g_{n+1}^{r_2'}, \ldots, g_{2n}^{r_2'}), e') \neq ((g_1^{r_1}, \ldots, g_n^{r_1}), (g_{n+1}^{r_2}, \ldots, g_{2n}^{r_2}), e)$ and $\alpha' = \alpha$. We argue that if this happens with non-negligible probability, then in fact, the family of hash functions is not universal one-way. Therefore, there exists a contradiction. For the case of type 2 invalid ciphertext, we can prove the lemma similarly and space doesn't permit to show the proof. □

Therefore, Claim 2.2 holds. □