Universal Leaky Random Oracle

Guangjun Fan[1], Yongbin Zhou[2], Dengguo Feng[1]

[1] Trusted Computing and Information Assurance Laboratory,Institute of Software,Chinese Academy of Sciences,Beijing,China
guangjunfan@163.com , feng@is.iscas.ac.cn
[2] State Key Laboratory of Information Security,Institute of Information Engineering,Chinese Academy of Sciences,Beijing,China
zhouyongbin@iie.ac.cn

Abstract. Yoneyama et al. introduces the Leaky Random Oracle Model at ProvSec2008 to capture the leakages from the hash list of a hash function used by a cryptography construction due to various attacks caused by sloppy usages or implementations in the real world. However, an important fact is that such attacks would leak not only the hash list, but also other secret states (e.g. the secret key) outside the hash list. Therefore, the Leaky Random Oracle Model is very limited in the sense that it considers the leakages from the hash list alone, instead of taking into consideration other possible leakages from secret states simultaneously. In this paper, we present an augmented model of the Leaky Random Oracle Model. In our new model, both the secret key and the hash list can be leaked. Furthermore, the secret key can be leaked continually during the whole lifecycle of the cryptography construction. Hence, our new model is more universal and stronger than the Leaky Random Oracle Model and some other leakage models (e.g. only computation leaks model and 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: leaky random oracle model, secret key, hash list, Cramer-Shoup cryptosystem, leakage.

1 Introduction

Hash function is one of the most important building blocks of cryptographic schemes and is widely used in various schemes. For example, public key cryptosystem, digital signature, authenticated key exchange, etc.

In practical sense, hash functions is used to hide private information to other parities in the protocol. The spreading use of transaction by small electronic devices has been encouraging researchers to develop an efficient and practical security system in a limited resources environment. Due to the computational costs of hash function is lower than that of public key cryptosystem, hash function is received much attention to construct protocols for such low-power devices.

In theoretical sense, hash function is modeled as a idealized model. This idealized model is called random oracle model [1] (ROM). Mostly, proofs with ROM are easier than the model without random oracles, i.e. the standard model (SM), and can provide tighter security reductions.

Unfortunately, Canetti et al. [2,3] showed that there are digital signature schemes and public key cryptosystems which are secure in ROM but insecure if random oracles are instantiated by real hash functions. However, proving security of a cryptography construction in SM is rather hard, ROM is an important tool to design new cryptography construction as the guideline of the provable security.

If possible, a cryptography construction in ROM will be implemented on some device in practical. A fact that cannot be neglected is that any implementation of a cryptography construction (both in ROM and in SM) can be attacked by physical attacks (such as cold boot attack [4] and side-channel attacks [6,7,8]) and attacks caused by sloppy usages of its implementation. The physical attacks and sloppy usages of the implementation of a cryptography construction may leak sensitive information in the cryptography construction. Usually, the adversary could exploit the leakage information to break a cryptography construction [4,6,7,8,14]. Therefore, leakage information of a cryptography construction pose a serious threat on the security of its implementation.

For a cryptography construction in ROM, when the random oracle is instantiated in practice using a hash function, all the pairs of inputs and outputs of the hash function (the contents of the hash list of the hash function) may be leaked to adversaries due to physical attacks and sloppy usages. For example, the hash list of a hash function may remain in the memory for reuse 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 by various attacks, e.g. cold boot attack, malicious Trojan Horse programs [5]. Yoneyama et al. [5] introduces the Leaky Random Oracle Model (LROM) considering this kind of leakages of hash list. In this model, all the contents of the hash list of a hash function are leaked to the adversary. They analyzed the security of five prevailing cryptography construction in LROM. The five prevailing cryptography construction are Full Domain Hash [1], Optimal Asymmetric Encryption Padding [17], Cramer-Shoup cryptosystem [9], Kurosawa-Desmedt cryptosystem [18] and NAXOS [19]. Clearly, the secret key of a cryptography construction which is secure in LROM must be out of the hash list.

1.1 Motivation

An important fact is that the adversary not only obtains leakages of the hash list of a hash function, but also leakages of other secret states outside of the hash list such as the secret key of a cryptography construction which is secure in LROM from physical attacks and sloppy usages. In many real world settings, the leakages of the secret key completely compromises the security of many cryptography construction. However, LROM only considers the leakages of the hash list. In other words, it is very difficult to guarantee the security of any cryptography construction which is secure in LROM when the adversary can obtain some leakage bits of the secret key occur by physical attacks and sloppy usages. In these senses, LROM is very limited.

In order to improve the limitation of LROM, we formulate a new leakage model in this paper. In our new model, both the secret key and the hash list of a hash function can be leaked. We believe that our new model is more universal and stronger than LROM due to leakages of the secret key and some other leakage models [10,13] when a cryptography construction in these models is based on random oracle. Furthermore, we try to construct a provably secure cryptography scheme 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. Our new model captures not only leakages of the hash list of a hash function used by a cryptography construction which is secure in LROM, but also leakages of the secret key of the cryptography construction. Second, we give a public key encryption scheme that is provably secure in this new model.

Universal Leaky Random Oracle Model Our new model named Universal Leaky Random Oracle Model (ULROM) allows the adversary to obtain both leakages of the secret key of a cryptography construction which is secure in LROM and leakages of the hash list of a hash function used by the cryptography construction simultaneously. We model the query in order to obtain leakage of the secret key of a cryptography construction as *leakage query*. As that in L-ROM, we model the query in order to obtain a hash value to the (leaky) random oracle as *hash query* and the special query in order to obtain all the contents of the hash list as *leak hash query*[1]. Any cryptography construction which is secure in ULROM will be also secure in LROM. However, it is very difficult to guarantee the security in ULROM for any cryptography construction which is secure in LROM. Our new model is also more universal and stronger than some other models [10,13] when a cryptography construction in these models is based on random oracle.

A Leakage Resilient Public Key Encryption Scheme in ULROM We also construct a leakage resilient public key encryption scheme in ULROM. This scheme is based on Cramer-Shoup cryptosystem and does not use any additional complex assumption and cryptography tool. In fact, it is a variant of the Cramer-Shoup cryptosystem with a different way of implementation. In our new leakage model, this scheme is IND-CCA secure even if the secret key is leaked *continually*.

[1] We rename the leak query in [5] as leak hash query because we define leakage query in our new model. Note that the leak hash query and the leak query in [5] are the same.

1.3 Related Works

Cold boot attack [4] and side-channel attacks [5] will leak sensitive information of a cryptography construction. In [10], a new class of strong side-channel attacks named memory attacks is defined. Moreover, memory attacks generalized the cold boot attack and side-channel attacks. Semantic Security Against Adaptive Memory Attacks Model and Semantic Security Against Non-Adaptive Memory Attacks Model are also presented in [10]. In [11], the Continual Memory Leakage Model (CMLM) is introduced. CMLM is stronger than the above two leakage models in [10], because the secret key can be refreshed and leaked continually. Our new model (ULROM) is based on CMLM and LROM.

1.4 Organization of This Paper

In section 2, we introduce some basic notation and concept. We present our new leakage model (ULROM) in section 3. Our provable secure public key encryption scheme in ULROM is introduced in section 4. In this section, we also prove the security of the scheme. We conclude this paper in section 5.

2 Preliminaries

In this section, we first present some symbols and notations used throughout the paper. Second, we review the Leaky Random Oracle Model. Third, Cramer-Shoup cryptosystem and the security of it are introduced. Finally, we introduce the basic assumptions which are used in this paper.

2.1 Symbols and Notations

Statistical Indistinguishability The statistical distance between two random variables X,Y is defined by

$$\mathbf{SD}(X,Y) = \frac{1}{2}\sum_{x}\Big|\Pr[X=x] - \Pr[Y=x]\Big|.$$

We write $X \overset{s}{\approx}_{\epsilon} Y$ to denote $\mathbf{SD}(X,Y) \le \epsilon$ and just plain $X \overset{s}{\approx} Y$ if the statistical distance is negligible in the security parameter. In the latter case, we say that X, Y are statistically indistinguishable.

Let *Gen* be 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 $\langle \boldsymbol{t}, \boldsymbol{s} \rangle = t_1 s_1 + t_2 s_2 + \cdots + t_n s_n$ 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 $\mathbb{Z}_q^n$.

2.2 Leaky Random Oracle Model

Yoneyama et al. [5] introduces the Leaky Random Oracle Model. We introduce LROM in Definition 1. LROM is trivially stronger than ROM [5]. There also exists separation between LROM and SM [5]. In [5], the difference between LROM and ROM under randomness revealing is shown.

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

2.3 Cramer-Shoup Cryptosystem is secure in LROM

Cramer-Shoup cryptosystem [9] is based on the DDH assumption and universal one-way hash function family. The description of Cramer-Shoup cryptosystem is as follows:

Key generation: For input security parameter k, generate a k-bit prime q. Let $\mathbb{G}$ is a group of prime order q. Choose $g_1, g_2 \in \mathbb{G}$ randomly and generate a secret key $sk = (x_1, x_2, y_1, y_2, z) \in \mathbb{Z}_q^5$ and public information (c, d, h) such that $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. Next, a hash function H is chosen from the family of universal one-way hash functions. The public key is $pk = (g_1, g_2, c, d, h, H)$ and the secret key is sk.

Encryption: Given a message $m \in \mathbb{G}$, it chooses $r \in \mathbb{Z}_q$ at random. Then it computes

$$u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}.$$

The ciphertext is (u_1, u_2, e, v).

Decryption: Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$ and tests if

$$u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} = v.$$

If this condition does not hold, the decryption algorithm outputs $\perp$; otherwise, it outputs $m = e/u_1^z$.

In [9], the security of Cramer-Shoup cryptosystem in standard model 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 DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA.*

In [5], the security of Cramer-Shoup cryptosystem in LROM is analysed. 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 where H is modeled as a leaky random oracle.*

2.4 Basic Assumptions

In this section, we introduce the assumptions used in this paper.

The Decisional Diffie-Hellman assumption. *The Decisional Diffie-Hellman (DDH) assumption is that the ensembles $\{\mathbb{G}, g_1, g_2, g_1^r, g_2^r\}$ and $\{\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2}\}$ are computationally indistinguishable, where $\mathbb{G} \leftarrow Gen(1^k)$, and the elements $g_1, g_2 \in \mathbb{G}$ and $r, r_1, r_2 \in \mathbb{Z}_q$ are chosen independently and uniformly at random.*

In this paper, we use an assumption which is equivalent to the DDH assumption. The assumption is in the following.

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

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

and

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

are computationally indistinguishable, where $\mathbb{G} \leftarrow Gen(1^k)$, 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 will show that the GDDH assumption and the DDH assumption are equivalent.

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

The proof of Theorem 1 is in Appendix B.

3 Universal Leaky Random Oracle Model

In [5], the Leaky Random Oracle Model only assumes that all the contents of the hash list of a hash function are leaked. For a cryptography construction which is secure in LROM, the secret key of the cryptography construction must be out of the hash list of a hash function used by the cryptography construction. However, in many real world settings, the the adversary not only obtains leakages of the hash list of a hash function, but also leakages of the other secret states outside the hash list such as the secret key of a cryptography construction which is secure in LROM from physical attacks and sloppy usages. Our new model considers these two kinds of leakages simultaneously. In our new model, the adversary can obtain both leakages of the hash list of a hash function and leakages of the secret key of a cryptography construction which is secure in LROM. Note that, we consider *continual* leakages of the secret key in our new model. Our new model is called Universal Leaky Random Oracle Model (ULROM).

As an example, we consider a public key encryption scheme which achieves IND-CCA security in ULROM. Similarly, one can define a IND-CPA secure public key encryption scheme or a signature scheme which is existentially unforgeable under an adaptive chosen-message attack in ULROM. A public key encryption scheme in ULROM 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 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, CT):** The input is a secret key SK and a ciphertext CT. The output is a decrypted message M.
– **Update(UK, SK):** The input is an update key UK and an old secret key SK. The output is an updated secret key SK'.

Note that the output of **Update(UK, SK)** i.e. SK' and SK are correspond to the same public key PK. This means that for a ciphertext CT which is encrypted by PK ($CT =$**Encrypt(PK, M)**), we have

$$\textbf{Decrypt}(SK, CT) = \textbf{Decrypt}(SK', CT) = M.$$

Note that the size of the secret key remains unchanged by update operations, i.e. $|SK| = |SK'|$. Moreover, the space of secret keys corresponding to the same public key should be large enough.

Let $L = L(k)$ be a function of the security parameter.

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

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

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

– *A random function H is chosen. Let $\mathcal{L}_H$ denotes the hash list of H. Initialization: $\mathcal{L}_H \leftarrow \perp$*
– *Challenger chooses $(PK, UK, SK_1) \leftarrow KeyGen(1^k)$.*
– *The adversary may ask for the following four queries:*
***Leakage query**: Each such query consists of a function $Leak : \{0,1\}^* \to \{0,1\}^L$ with L bits output. On the i^{th} such query $Leak_i$, 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 as the answer to the hash query.
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 as the answer to the hash query.
***Leak hash query:** For a leak hash query to H, output all contents of the hash list $\mathcal{L}_H$.*
***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 the adversary $\mathcal{A}$.*

– *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 the adversary $\mathcal{A}$.*
– *The adversary $\mathcal{A}$ can not ask leakage query after he gets CT^*. The adversary $\mathcal{A}$ can also ask the hash query and the leak hash query. The adversary $\mathcal{A}$ can also ask the decryption query. But he cannot ask the decryption query with CT^*.*
– *The adversary $\mathcal{A}$ outputs a bit b'. If $b' = b$, the experiment outputs 1, otherwise, the experiment outputs 0.*

Note that the leaky hash query and the leakage query are essentially different in ULROM. On one hand, the secret key must be out of the hash list for a cryptography construction that is secure in LROM, which means that the secret key can not be leaked from leak hash query. On the other hand, the leaky hash query leaks all the contents of the hash list, while the leakage query only leaks a part of the secret key between two updates.

Leaky Random Oracle Model in [5] only allows the adversary to obtain leakages of the hash list. Therefore, it is very hard to guarantee the security of a cryptography construction which is secure in LROM when the secret key that is outside the hash list could be leaked.

In recent years, many leakage models [10,11,13,15,16] are given out. Undoubtedly, if one try to design a cryptography construction with ROM in these models, he may get simpler proof and tighter security reductions. The paper [5] shows that the leakages of the hash list of a hash function which is used to instantiated the ROM may threaten the security of a cryptography construction that is secure in ROM. However, the above models mainly consider the leakages about the secret key and do not consider the leakages about the hash list when a cryptography construction in these models uses ROM. Therefore, this is a weakness of the models. We do not know the relation between the models [10,11,13,15,16] and LROM. This is a valuable question.

In our new model, the adversary can get both leakages of the hash list and *continual* leakages of the secret key. Therefore, our new model is more universal and stronger than Leaky Random Oracle Model and some leakage models [10,13]. For other leakage models [11,15,16], although they consider some additional leakages such as the leakage during key generation process, they do not consider the leakages of hash list when a cryptography construction in these models uses hash function.

In next section, we present a public key encryption scheme which is IND-CCA secure in ULROM.

4 A Provably Secure Public Key Encryption Scheme in ULROM

In this section, we first introduce our public key encryption scheme in ULROM and then prove the security of it.

Let vector $\mathbf{1_n} = (1, 1, \ldots, 1)$, there exists n components in the vector.

Our public key encryption scheme in ULROM is denoted by PKE_{ULROM} and is based on Cramer-Shoup cryptosystem. The PKE_{ULROM} is shown in the following.

KeyGen: For input security parameter k, generate a k bit prime q. Let $\mathbb{G}$ is a group of prime order q. The generator of $\mathbb{G}$ is g. Choose $t, s \in \mathbb{Z}_q$ uniformly at randomly. Let $g_1 = g^t, g_2 = g^s$. Generating five random numbers $(x_1, x_2, y_1, y_2, z) \in \mathbb{Z}_q^5$ and compute (c, d, h) such that $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. Next, choose a hash function H from a family of universal one-way hash functions. Computing $t_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n$ at random such that $\sum_{i=1}^{n} t_i \bmod q = t$. Similarly, computing $s_i \in \mathbb{Z}_q, i = 1, 2, \ldots, n$ at random such that $\sum_{i=1}^{n} s_i \bmod q = s$. Denote vector $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ and vector $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$.

Computing a random vector $\boldsymbol{x_1} = (x_{11}, x_{12}, \ldots, x_{1n})$, where $x_{1i} \in \mathbb{Z}_q,\ i = 1, 2, \ldots, n$ such that $\langle \boldsymbol{t}, \boldsymbol{x_1} \rangle \bmod q = tx_1 \bmod q$. Computing a random vector $\boldsymbol{x_2} = (x_{21}, x_{22}, \ldots, x_{2n})$, where $x_{2i} \in \mathbb{Z}_q,\ i = 1, 2, \ldots, n$ such that $\langle \boldsymbol{s}, \boldsymbol{x_2} \rangle \bmod q = sx_2 \bmod q$.

Computing a random vector $\boldsymbol{y_1} = (y_{11}, y_{12}, \ldots, y_{1n})$, where $y_{1i} \in \mathbb{Z}_q,\ i = 1, 2, \ldots, n$ such that $\langle \boldsymbol{t}, \boldsymbol{y_1} \rangle \bmod q = ty_1 \bmod q$. Computing a random vector $\boldsymbol{y_2} = (y_{21}, y_{22}, \ldots, y_{2n})$, where $y_{2i} \in \mathbb{Z}_q,\ i = 1, 2, \ldots, n$ such that $\langle \boldsymbol{s}, \boldsymbol{y_2} \rangle \bmod q = sy_2 \bmod q$.

Computing a random vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$, where $z_i \in \mathbb{Z}_q,\ i = 1, 2, \ldots, n$ such that $\langle \boldsymbol{t}, \boldsymbol{z} \rangle \bmod q = tz \bmod q$.

The public key pk is $(g^{\boldsymbol{t}}, g^{\boldsymbol{s}}, c, d, h, H)$. The secret key sk is a $n \times 5$ matrix, i.e. $sk = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^T$. The update key uk is $(\boldsymbol{t}, \boldsymbol{s})$.

Encrypt: For input a message $m \in \mathbb{G}$, choose $r \in \mathbb{Z}_p$ at random, compute $u_1 = g^{r\langle \boldsymbol{t}, \boldsymbol{1_n} \rangle} = g_1^r$, $u_2 = g^{r\langle \boldsymbol{s}, \boldsymbol{1_n} \rangle} = g_2^r$, $e = h^r m$, $\alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$. Output a ciphertext $(g^{r\boldsymbol{t}}, g^{r\boldsymbol{s}}, e, v)$.

Decrypt: Given a ciphtertext $(g^{r\boldsymbol{t}}, g^{r\boldsymbol{s}}, e, v)$, compute $u_1 = g^{\langle r\boldsymbol{t}, \boldsymbol{1_n} \rangle}$, $u_2 = g^{\langle r\boldsymbol{s}, \boldsymbol{1_n} \rangle}$, $\alpha = H(u_1, u_2, e)$ and verify 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 output the message $m = e / g^{\langle r\boldsymbol{t}, \boldsymbol{z} \rangle}$ by using $\boldsymbol{z}$. Else if, reject the decryption as an invalid ciphertext $\perp$.

Update: Let $sk = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z}]^T$ be a $n \times 5$ matrix denotes the old secret key. Chooses $\beta_1, \beta_3, \beta_5 \in ker(\boldsymbol{t})$ and $\beta_2, \beta_4 \in ker(\boldsymbol{s})$ uniformly at random.

Let matrix $up = [\beta_1, \beta_2, \beta_3, \beta_4, \beta_5]^T$ be a $n \times 5$ matrix. Let the new updated secret key be $sk' = sk + up$. Outputs sk'.

We first verify the correctness of the scheme. We have

$$g^{\langle r\boldsymbol{t}, \boldsymbol{x_1}\rangle + \langle r\boldsymbol{s}, \boldsymbol{x_2}\rangle} = g_1^{rx_1} g_2^{rx_2} = c^r.$$

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

The following theorem establishes the security of the scheme:

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_{ULROM} is IND-CCA secure in Universal Leaky Random Oracle model, as long as $L < (n-5)log(q) - \omega(log(k))$.*

Proof. We define a new experiment $Expt_{\Pi,\mathcal{A}}^{RandomLeakageCCA}(b)$ for a public key encryption Π and any probabilistic polynomial time adversary $\mathcal{A}$. The experiment $Expt_{\Pi,\mathcal{A}}^{RandomLeakageCCA}(b)$ is identical to the experiment $Expt_{\Pi,\mathcal{A}}^{LeakageCCA}(b)$ except that in the *leakage query*, the challenger chooses random numbers (denoted by UR_i) with the same size of the secret key and sends $Leak_i(UR_i)$ to the adversary. For our scheme, in the experiment $Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(b)$, when the adversary sends a leakage function $Leak_i$ to the leakage oracle, the challenger chooses a matrix $UR_i \in \mathbb{Z}_q^{n\times 5}$ uniformly at random and sends $Leak_i(UR_i)$ to the adversary.

For our scheme PKE_{ULROM} it holds that

$$\begin{aligned} Adv_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(k) = \Big|&\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0) = 1] - \\ &\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(1) = 1]\Big| \\ \le \Big|&\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0) = 1]\Big| \\ +\Big|&\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(1) = 1]\Big| \\ +\Big|&\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(1) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(1) = 1]\Big|. \end{aligned}$$

We will prove this theorem by the following three claims.
Claim 2.1 *As long as $L < (n-5)log(q) - \omega(log(k))$, it holds that*

$$\Big|\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0) = 1]\Big| < \mu_1(k)$$

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

Proof. By the following lemma, as long as $L < (n-5)log(q) - \omega(log(k))$, the leakages of the real secret key sk_i is distinguishable with the leakages of random matrix UR_i for any leakage function $Leak_i$.
Lemma 3 *(Dual Subspace Hiding) Let $n \geq d \geq u$ be integers. Let $Leak : \{0,1\}^* \to \{0,1\}^L$ be some arbitrary function. For randomly sampled $C \overset{*}{\leftarrow} \mathbb{Z}_q^{n \times d}$, $E \overset{*}{\leftarrow} \mathbb{Z}_q^{d \times u}$, $UR \overset{*}{\leftarrow} \mathbb{Z}_q^{n \times u}$, we have:*

$$(Leak(CE), C) \overset{s}{\approx} (Leak(UR), C)$$

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

In our scheme, the secret key sk is a $n \times 5$ matrix in $\mathbb{Z}_p^{n \times 5}$. sk can be decomposed into the product of two matrix C and E easily, where C is a matrix in $\mathbb{Z}_q^{n \times n}$ and E is a matrix in $\mathbb{Z}_q^{n \times 5}$. Therefore, as long as $L < (n-5)log(q) - \omega(log(k))$, the leakages of the secret key and the leakages of a random matrix in $\mathbb{Z}_p^{n \times 5}$ can not be distinguished. Lemma 3 was first formulated by Z.Brakerski et al. [11] and was improved by S.Agrawal et al. [12]. Without loss of generality, assume that the attacker makes exactly T^1 leakage queries. In $Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0)$, the adversary obtains $\{Leak_i(sk_i)\}_{i=1}^T$ from leakage queries. In $Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0)$, the adversary obtains $\{Leak_i(UR_i)\}_{i=1}^T$ from leakage queries, where UR_i are sampled uniformly at random from $\mathbb{Z}_q^{n \times 5}$. The difference between $Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0)$ and $Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0)$ is only the leakage information from the *leakage query*. By lemma 3, $\{Leak_i(sk_i)\}_{i=1}^T$ and $\{Leak_i(UR_i)\}_{i=1}^T$ are statistically indistinguishable as long as $L < (n-5)log(q) - \omega(log(k))$. Hence, $Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(0)$ and $Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0)$ are statistically indistinguishable. Therefore, Claim 2.1 holds.□
Claim 2.2 *If the GDDH assumption of the group $\mathbb{G}$ holds, we have*

$$\left|\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(0) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(1) = 1]\right| < \mu_2(k)$$

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

We show the proof of Claim 2.2 in Appendix A. The main idea of the proof is that the leakage queries in the two experiments leak no information about the real secret key. Therefore, the adversary obtains no information about the real secret key. Furthermore, the Cramer-Shoup cryptosystem is IND-CCA secure in LROM (Lemma 2). Although our scheme PKE_{ULROM} is a variant of the Cramer-Shoup cryptosystem with a different way of implementation, the principle of theory of our scheme is identical to the Cramer-Shoup cryptosystem except that the basic assumptions of the two schemes are different[1]. Hence the Claim 2.2 holds.
Claim 2.3 *As long as $L < (n-5)log(q) - \omega(log(k))$, it holds that*

$$\left|\Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{LeakageCCA}(1) = 1] - \Pr[Expt_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA}(1) = 1]\right| < \mu_3(k)$$

[1] Note that $T = poly(k)$.
[1] But the two assumptions are equivalent. See section 2 for more details.

, 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_{ULROM} is IND-CCA secure in ULROM. □

The result of [11], can be used to show that *any* scheme that is secure against continual leakage, can tolerate $O(logk)$ leakage from each update process, and thus our scheme can tolerate such leakage as well.

5 Conclusion and Future Work

In this paper, we introduce a new leakage model based on Leaky Random Oracle Model and other leakage models [10,11,13]. In this new model, both the secret key and the hash list of a hash function used by a cryptography construction which is secure in LROM can be leaked. Moreover, the secret key can be leaked continually and refreshed. Therefore, we believe that our new model is more universal and stronger than the Leaky Random Oracle Model. We also present a new public key encryption scheme (PKE_{ULROM}) which is IND-CCA secure in this new model. In future work, one may try to consider additional leakage in the key generation process. Leakage resilient signature scheme in our new leakage model is also expected. LROM is independent from SM [5]. However, we do not clearly know the relation between ULROM and SM. Hence, what is the relation between ULROM and SM is an important question.

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. R. Canetti, O. Goldreich, and S. Halevi.: The Random Oracle Methodology, Revisited (Preliminary Version). STOC1998, pp.131-140,1998.
3. R. Canetti, O. Goldreich, and S. Halevi.: The Random Oracle Methodology, Revisited. J.ACM 51(4), pp.557-594,2004.
4. J.A. Halderman, SD. 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. S. Dziembowski, K. Pietrzak.: Leakage-Resilient Cryptography. FOCS2008, pp.293-302, 2008.
14. M. Medwed, E. Oswald.: Template Attacks on ECDSA. WISA2008, LNCS 5379, pp.14-27, 2009.
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.

Appendix A: Proof of Claim 2.2

Proof. Equivalently, we redefine the advantage of an adversary as follows:

$$Adv^{RandomLeakageCCA'}_{PKE_{ULROM},\mathcal{A}}(k) = 2\left|\Pr[Expt^{RandomLeakageCCA'}_{PKE_{ULROM},\mathcal{A}}(k) = 1] - \frac{1}{2}\right|,$$

where $Expt^{RandomLeakageCCA'}_{PKE_{ULROM},\mathcal{A}}$ is as follows:

- A random function H is chosen. Let $\mathcal{L}_H$ denotes the hash list of H. Initialization: $\mathcal{L}_H \leftarrow \perp$
- Challenger chooses $(PK, UK, SK) \leftarrow KeyGen(1^k)$.
- The adversary may ask for the following four queries:
  Leakage query: Each such query consists of a function $Leak : \{0,1\}^* \rightarrow \{0,1\}^L$ with L bits output. On the ith such query $Leak_i$, the challenger gives the value $Leak_i(UR_i)$ to $\mathcal{A}$, where $UR_i \overset{*}{\leftarrow} \mathbb{Z}_q^{n\times 5}$ and is sampled uniformly at random.
  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 as the answer to the hash query.
  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 as the answer to the hash query.
  Leak hash query: For a leak hash query to H, output all contents of the hash list $\mathcal{L}_H$.
  Decryption query: For a decryption query with a ciphertext CT, decrypts CT with the secret key SK and sends $Decrypt(SK, CT)$ to the adversary $\mathcal{A}$.

- At some point $\mathcal{A}$ gives the challenger two messages M_0, M_1 and $|M_0| = |M_1|$. The challenger chooses $b \in \{0,1\}$ uniformly at random and computes $CT^* \leftarrow Encrypt(PK, M_b)$. Then the challenger sends CT^* as the challenge ciphertext to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ can not ask leakage query after he gets CT^*. The adversary $\mathcal{A}$ can also ask the hash query and the leak hash query. The adversary $\mathcal{A}$ can also ask the decryption query. But he cannot ask the decryption query with CT^*.
- The adversary $\mathcal{A}$ outputs a bit b'. If $b' = b$, the experiment outputs 1, otherwise, the experiment outputs 0.

If $Adv_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA'}(k)$ is negligible, then Claim 2.2 can be proved. We will prove that $Adv_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA'}(k)$ is negligible in the following.

Assume that $Adv_{PKE_{ULROM},\mathcal{A}}^{RandomLeakageCCA'}(k)$ is non-negligible and the hash family is universal one-way. Then there exists an adversary $\mathcal{A}$ that can break the scheme PKE_{ULROM}. 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

$\{(\{g_1, \ldots, g_n\}, \{g_{n+1}, \ldots, g_{2n}\}, \{g_1^{r_1}, \ldots, g_n^{r_1}\}, \{g_{n+1}^{r_2}, \ldots, g_{2n}^{r_2}\}) | g_1, \ldots, g_{2n} \overset{*}{\leftarrow} \mathbb{G}, r_1, r_2 \overset{*}{\leftarrow} \mathbb{Z}_q\}$.

We will show that 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. We will also show that 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

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

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 $\{(g_1, \dots, g_n), (g_{n+1}, \dots, g_{2n}), c, d, h, H\}$ as the public key to $\mathcal{A}$. The secret key is $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z_1}, \boldsymbol{z_2}]^T$.

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, due to UR_i is sampled uniformly at random from $\mathbb{Z}_q^{n\times 5}$, it has no relation with the actual secret key. Therefore, $Leak_i(UR_i)$ leaks no information about the actual secret key $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{z_1}, \boldsymbol{z_2}]^T$.

The adversary $\mathcal{B}$ answers the *hash query* and *leaky hash query* normally. Note that *leaky hash query* in ULROM cannot be advantage of adversaries. The reason is that all inputs and outputs of hash function H are publicly known because a ciphertext contains (u_1, u_2, e) which are the inputs to the hash function. Naturally, adversaries can know the input and the output in each session.

The adversary $\mathcal{B}$ answers the *decryption query* as follows: For a decryption query $((g_1^{r'_1}, \dots, g_n^{r'_1}), (g_{n+1}^{r'_2}, \dots, g_{2n}^{r'_2}), e', v')^1$ 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, obtain α' and verify 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 $(\{g_1^{r_1}, \dots, g_n^{r_1}\}, \{g_{n+1}^{r_2}, \dots, g_{2n}^{r_2}\}, e, v)$ as the challenge ciphertext to $\mathcal{A}$.

Let g be the generator of the group $\mathbb{G}$. We know that there exist $t_i \in \mathbb{Z}_q$ such that $g_i = g^{t_i}, i = 1, \dots, n$. There exist $s_i \in \mathbb{Z}_q$ such that $g_{i+n} = g^{s_i}, i = 1, \dots, n$. 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

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

^{1} If $r'_1 = r'_2$, then the ciphertext is valid.

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. The adversary $\mathcal{B}$ can answer $\mathcal{A}$'s all queries correctly without knows these values. Due to 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 from $\mathbb{Z}_q^n$, the values $\{x_1, x_2, y_1, y_2, z_1, z_2\}$ are chosen independently and uniformly at random from $\mathbb{Z}_q$.

As we will see, when the input to adversary $\mathcal{B}$ comes from $\mathbf{D}$, the challenge ciphertext is a perfectly legitimate ciphertext; however, when the input to 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 4 *When the adversary $\mathcal{B}$'s input comes from $\boldsymbol{D}$, 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 $((g_1^{r_1'}, g_2^{r_1'}, \ldots, g_n^{r_1'}), (g_{n+1}^{r_2'}, g_{n+2}^{r_2'}, \ldots, g_{2n}^{r_2'}), e', v')$ a valid ciphertext if $r_1' = r_2'$ (an invalid ciphertext if $r_1' \neq r_2'$). 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 A.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 $log()$ deonte $log_{g^t}()$.

From the adversary's view, $\mathbf{P}$ is a random point on the plane $\mathcal{P}$ formed by intersecting the hyperplanes

$$log(c) = x_1 + wx_2 \text{ (1) and } log(c) = y_1 + wy_2 \text{ (2).}$$

These two equations come from the public key. The challenge ciphertext dose not constrain $\mathbf{P}$ any further, as the hyperplane defined by

$$log(v) = rx_1 + wrx_2 + \alpha ry_1 + \alpha rwy_2 \tag{3}$$

contains $\mathcal{P}$.

Now suppose the adversary $\mathcal{A}$ submits an 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 **P** happens to lie on the hyperplane $\mathcal{H}$ defined by

$$log(v') = r'_1 x_1 + w r'_2 x_2 + \alpha' r'_1 y_1 + \alpha' r'_2 y_2, \tag{4}$$

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 an invalid ciphertext, the decryption oracle rejects with probability $1 - 1/q$. This rejection actually constrains the point **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 invalid ciphertexts, except with negligible probability.

Lemma 5 *When adversary $\mathcal{B}$'s input comes from $\boldsymbol{R}$, 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 A.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

$$log(h) = z_1 + w z_2, \tag{5}$$

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 $r' log(h) = r' z_1 + r' w z_2$. Thus, no further information about **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

$$log(\gamma) = r_1 z_1 + w r_2 z_2 \tag{6}$$

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

Proof. We study the distribution of $P = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, 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

$$log(v) = r_1x_1 + wr_2x_2 + \alpha r_1y_1 + \alpha wr_2y_2. \tag{7}$$

Now assume that the adversary submits an 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 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. This can be verified by observing that

$$det \begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1 & wr_2 & \alpha r_1 & \alpha wr_2 \\ r'_1 & wr'_2 & \alpha' r'_1 & \alpha' wr'_2 \end{pmatrix} = w^2(r_2 - r_1)(r'_2 - r'_1)(\alpha - \alpha') \neq 0.$$

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.

Therefore, Claim 2.2 holds. □

Appendix B: 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

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

and

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

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 $\{\mathbb{G}, g_1, g_2, g_1^r, g_2^r\}$, 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 $\{\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2}\}$, 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

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

and

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

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. □