How to Prove the Security of Practical Cryptosystems with Merkle-Damgård Hashing by Adopting Indifferentiability

Yusuke Naito[1], Kazuki Yoneyama[2], Lei Wang[2], and Kazuo Ohta[2]

[1] Mitsubishi Electric Corporation
Naito.Yusuke@ce.MitsubishiElectric.co.jp
[2] The University of Electro Communications
{yoneyama,wanglei,ota}@ice.uec.ac.jp

Abstract. In this paper, we show that major cryptosystems such as FDH, OAEP, and RSA-KEM are secure under a hash function MD^h with Merkle-Damgård (MD) construction that uses a random oracle compression function h. First, we propose two new ideal primitives called Traceable Random Oracle ($\mathcal{TRO}$) and Extension Attack Simulatable Random Oracle ($\mathcal{ERO}$) which are weaker than a random oracle ($\mathcal{RO}$). Second, we show that MD^h is indifferentiable from $\mathcal{LRO}$, $\mathcal{TRO}$ and $\mathcal{ERO}$, where $\mathcal{LRO}$ is Leaky Random Oracle proposed by Yoneyama et al. This result means that if a cryptosystem is secure in these models, then the cryptosystem is secure under MD^h following the indifferentiability theory proposed by Maurer et al. Finally, we prove that OAEP is secure in the $\mathcal{TRO}$ model and RSA-KEM is secure in the $\mathcal{ERO}$ model. Since it is also known that FDH is secure in the $\mathcal{LRO}$ model, as a result, major cryptosystems, FDH, OAEP and RSA-KEM, are secure under MD^h, though MD^h is not indifferentiable from $\mathcal{RO}$.

Keywords: Indifferentiability theory, Merkle-Damgård hash function, Random Oracle, Full Domain Hash (FDH) signature, Optimal Asymmetric Encryption Padding (OAEP) encryption, RSA-based key encapsulation mechanism (RSA-KEM) scheme

1 Introduction

1.1 Indifferentiability Framework

Maurer et al. [11] introduced the indifferentiable framework as a stronger notion than indistinguishability. This framework deals with the security of two systems $\mathcal{C}(\mathcal{V})$ and $\mathcal{C}(\mathcal{U})$: for cryptosystem $\mathcal{C}$, $\mathcal{C}(\mathcal{V})$ retains at least the same level of provable security of $\mathcal{C}(\mathcal{U})$ if primitive $\mathcal{V}$ is indifferentiable from primitive $\mathcal{U}$, denoted $\mathcal{V} \sqsubset \mathcal{U}$. Using this framework, we can state the following fact: if $\mathcal{C}(\mathcal{U})$ is a secure cryptosystem and the primitive $\mathcal{V}$ is indifferentiable from $\mathcal{U}$, then $\mathcal{C}(\mathcal{V})$ is secure, and if the primitive $\mathcal{V}$ is not indifferentiable from $\mathcal{U}$, there is some cryptosystem which is secure in the $\mathcal{U}$ model but insecure in the $\mathcal{V}$ model.

1.2 Cryptosystems and Applications of Indifferentiability

While many cryptosystems have been proved secure in the random oracle ($\mathcal{RO}$) model [2] (e.g. FDH [2], OAEP[3], RSA-KEM[17] and so on) where $\mathcal{RO}$ is modeled as a monolithic entity (i.e. a black box working in domain $\{0,1\}^*$), in practice it is instantiated by a hash function that is usually constructed by iterating a fixed input length primitive (e.g. a compression function). There are many architectures based on iterated hash functions. The most well-known one is Merkle-Damgård (MD) construction [7, 12]. A hash function with MD construction iterates underlying compression function $f : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^n$ is as follows.

```
$MD^f(m_1, ..., m_l)$ ($|m_i| = t, i = 1, ..., l$):
    let $y_0 = IV$ be some $n$ bit fixed value.
    for $i = 1$ to $l$ do $y_i = f(y_{i-1}, m_i)$
    return $y_l$
```

There is a significant gap between $\mathcal{RO}$ and hash functions, since hash functions are constructed from a small primitive, f, and $\mathcal{RO}$ is the the monolithic random function.

Coron et al. [6] made important observations on these cryptosystems using the indifferentiable framework. They introduced the new iterated hash function property of indifferentiability from $\mathcal{RO}$. In this framework, the underlying primitive, G, is compression function h of a random oracle or an ideal block cipher. We say that hash function H^G is indifferentiable from $\mathcal{RO}$ if there exists simulator $\mathcal{S}$ such that no distinguisher can distinguish H^G from $\mathcal{RO}$ ($\mathcal{S}$ mimics G). The distinguisher can access $\mathcal{RO}/H^G$ and $\mathcal{S}/G$; $\mathcal{S}$ can access $\mathcal{RO}$. A hash function, H^G, satisfying this property behaves like $\mathcal{RO}$. Therefore, the security of any cryptosystem is preserved when $\mathcal{RO}$ is replaced by H^G.

Coron et al. analyzed the indifferentiability of $\mathcal{RO}$ for several specific constructions. For example, they have shown that MD^h is not indifferentiable from $\mathcal{RO}$ due to the extension attack which uses the following fact: The output value $z' = MD^h(M||m)$ can be calculated by $c = h(z, m)$ where $z = MD^h(M)$, so $z' = c$. On the other hand, no S can return the output value $z' = \mathcal{RO}(M||m)$ from the query (z, m) where $z = \mathcal{RO}(M)$, since no $\mathcal{S}$ can know z' from z and m, and z' is randomly chosen. Therefore no $\mathcal{S}$ can simulate the extension attack. This result implies that MD^h does not behave like $\mathcal{RO}$ and there exists some cryptosystem that is secure in the $\mathcal{RO}$ model but insecure under MD^h. Their counter action was the proposal of several constructions such as Prefix-Free MD, chop MD, NMAC and HMAC. Hash functions with these constructions under h are indifferentiable from $\mathcal{RO}$ but the work fails to prove the important original MD cryptosystem is secure.

1.3 Is MD Construction Dead?

MD construction is among the most important blocks of modern cryptosystems [1, 6, 9]. There are two main reasons:

- MD construction is employed by many popular hash functions such as SHA-1 and SHA-256, and
- MD construction is more efficient than other iterated hash function types such as Prefix-Free MD, and chop MD.

Since MD^h is not indifferentiable from $\mathcal{RO}$, there is some cryptosystem $\mathcal{C}^*$ that is secure in the $\mathcal{RO}$ model but insecure under MD^h:

$$MD^h \not\sqsubset \mathcal{RO} \Rightarrow \exists \mathcal{C}^* \text{ s.t. } \mathcal{C}^*(\mathcal{RO}) \text{ is secure and } \mathcal{C}^*(MD^h) \text{ is insecure.}$$

Thus the important question is "can we confirm that a certain given cryptosystem is secure in the $\mathcal{RO}$ model and secure under MD^h?":

$$\text{For a set } \mathcal{C}_{\mathcal{RO}} \text{ of secure cryptosystems in the } \mathcal{RO} \text{ model, } \exists \mathcal{C}_0 \in \mathcal{C}_{\mathcal{RO}} \text{ s.t. } \mathcal{C}_0(MD^h) \text{ is secure?}$$

There might be several cryptosystems that remain secure when $\mathcal{RO}$ is replaced by MD^h. If we can confirm this for major cryptosystems which are widely used, the MD construction is still alive in the indifferentiability theory!

1.4 Our Contribution

We take following approaches to rescue cryptosystems under MD^h.

1. Find an ideal primitive $\widetilde{\mathcal{RO}}$ from which MD^h is indifferentiable.
2. Prove that cryptosystem $\mathcal{C}_0$ is secure in the $\widetilde{\mathcal{RO}}$ model.

If we can find $\widetilde{\mathcal{RO}}$ for the hash function MD^h (claim 1) and many cryptosystems are secure in the $\widetilde{\mathcal{RO}}$ model (claim 2), the MD construction is still alive in the indifferentiability theory, since these cryptosystems are secure under MD^h.

In order for $\mathcal{S}$ to be able to simulate the extension attack, it is necessary for $\mathcal{S}$ to know $z' = \mathcal{RO}(M||m)$ from (z, m) where $z = \mathcal{RO}(M)$. However, no $\mathcal{S}$ can know z' from (m, z) in the $\mathcal{RO}$ model. So we consider three models, *Leaky Random Oracle* ($\mathcal{LRO}$), *Traceable Random Oracle* ($\mathcal{TRO}$) and *Extension Attack Simulatable Random Oracle* ($\mathcal{ERO}$) model as $\widetilde{\mathcal{RO}}$, where an additional oracle is assumed for $\mathcal{S}$ to obtain $z' = \mathcal{RO}(M||m)$.

Our results are summarised as follows:,

1. Introduce several $\mathcal{RO}$ variants, $\mathcal{LRO}$, $\mathcal{TRO}$ and $\mathcal{ERO}$ for $\widetilde{\mathcal{RO}}$, and prove $\mathcal{RO} \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$. Moreover prove $\mathcal{LRO} \not\sqsubset \mathcal{TRO} \not\sqsubset \mathcal{ERO}$.
2. Prove $MD^h \sqsubset \mathcal{ERO}$. Therefore $MD^h \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$ holds (claim 1).
3. Prove that OAEP is secure in the $\mathcal{TRO}$ model and RSA-KEM is secure in the $\mathcal{ERO}$ model. (It is also known that FDH is secure in the $\mathcal{LRO}$ model [18].) (claim 2)
4. Prove RSA-KEM is insecure in the $\mathcal{TRO}$ model. (It is also known that OAEP is insecure in the $\mathcal{LRO}$ model [18].)

By combining the second and third results, we can obtain the result that FDH, OAEP, and RSA-KEM are secure under MD^h adopting the indifferentiability framework.

The latter part of the first result implies that there is some cryptosystem secure in the $\mathcal{TRO}$ model but insecure in the $\mathcal{LRO}$ model, and some cryptosystem secure in the $\mathcal{ERO}$ model but insecure in the $\mathcal{TRO}$ model. The forth result gives concrete examples of practical and major cryptosystems: that is, OAEP is separating between $\mathcal{LRO}$ and $\mathcal{TRO}$, and RSA-KEM is separating between $\mathcal{TRO}$ and $\mathcal{ERO}$. Table 1 summarises the security of FDH, OAEP, and RSA-KEM in variant $\mathcal{RO}$ models.

	FDH signature	OAEP encryption	RSA-KEM scheme
Leaky Random Oracle ($\mathcal{LRO}$)	secure	insecure	insecure
Traceable Random Oracle ($\mathcal{TRO}$)	secure	secure	insecure
Extension Attack Simulatable $\mathcal{RO}$ ($\mathcal{ERO}$)	secure	secure	secure

Table 1. Security of Major Cryptosystems in Variant $\mathcal{RO}$ Models

In this paper, we succeed in proving that major cryptosystems including FDH, OAEP, and RSA-KEM are secure under MD^h! We can say that the *MD construction is still alive* !!

1.5 Related Works

Several works have introduced variants of the random oracle as follows.

In [10, 15, 14], the oracle are modeled on breaking one-way of a hash function. Since the number of input elements of a hash function is more than the number of output elements of a hash function, the probability that the additional oracle returns a unique input value where $\mathcal{S}$ can simulate the extension attack is negligible. Therefore MD^h is not indifferentiable from the oracle consisting of both $\mathcal{RO}$ and this additional oracle.

These studies were made in corresponding to recent attacks on hash functions. The goal of these studies is clearly different from our goal, because our goal is to analyse the security of cryptosystems under MD^h by "using the indifferentiability theory" while their goals are analyse cryptosystems reflecting concrete attacks of hash functions.

Coron *et al.* [6] and Chang *et al.* [5] have proven that so-called *Prefix-free MD* is indifferentiable from $\mathcal{RO}$.

1.6 Road Map of the Paper

We start with some preliminaries (MD construction, definition of the indifferentiability, and the extension attack) in section 2. In section 3, we introduce variants of $\mathcal{RO}$ model, i.e., the $\mathcal{LRO}$, $\mathcal{TRO}$, and $\mathcal{ERO}$ models, and prove both $\mathcal{RO} \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$ and $\mathcal{LRO} \not\sqsubset \mathcal{TRO} \not\sqsubset \mathcal{ERO}$. In section 4, we prove $MD^h \sqsubset \mathcal{ERO}$, which is the main result. Therefore we also obtain $MD^h \sqsubset \mathcal{TRO}$ and $MD^h \sqsubset \mathcal{LRO}$ from $\mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$, which is an answer to claim 1. We will prove that OAEP is secure in the $\mathcal{TRO}$ model in section 5, and that RSA-KEM is secure in the $\mathcal{ERO}$ model but insecure in the $\mathcal{TRO}$ model in section 6, respectively, which is an answer to claim 2. In section 7, we discuss another approach to find secure cryptosystems under MD^h. As an example, the security of OAEP under MD^h can be also proved by adopting this approach.

The proofs that FDH/OAEP/RSA-KEM are secure in the variants $\mathcal{RO}$ models are also described in Appendixes as well as that of the main theorem.

2 Preliminaries

2.1 Merkle-Damgård Construction

We first give a short description of Merkle-Damgård (MD) construction. The function $MD^f : \{0,1\}^* \rightarrow \{0,1\}^n$ is built by iterating a compression function $f : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^n$ as follows.

– $MD^f(M)$:
  1. calculate $M' = pad(M)$ where pad is a padding function such that $pad : \{0,1\}^* \rightarrow (\{0,1\}^t)^*$.
  2. calculate $c_i = f(c_{i-1}, m_i)$ for $i = 1, ..., l$ where for $i = 1, ..., l$, $|m_i| = t$, $M' = m_1||...||m_l$ and c_0 is an initial value (s.t. $|c_0| = n$).
  3. return c_n

In this paper we ignore the above padding function but this implies no loss of generality, so hereafter we discuss $MD^f : (\{0,1\}^t)^* \rightarrow \{0,1\}^n$. And we use a random oracle compression function h as f where $h : \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^n$ and h is a random function. So we discuss about the hash function MD^h with MD construction using h.

2.2 Indifferentiability Framework for Hash Functions

The indifferentiability framework generalizes the fundamental concept of the indistinguishability of two crypto systems $\mathcal{C}(\mathcal{U})$ and $\mathcal{C}(\mathcal{V})$ where $\mathcal{C}(\mathcal{U})$ is the cryptosystem $\mathcal{C}$ invoking the underlying primitive $\mathcal{U}$ and $\mathcal{C}(\mathcal{V})$ is the cryptosystem $\mathcal{C}$ invoking the underlying primitive $\mathcal{V}$. $\mathcal{U}$ and $\mathcal{V}$ have two interfaces: public and private interfaces. Adversaries can only access the public interface and honest parties (e.g. the cryptosystem $\mathcal{C}$) can only access the private interface. Hereafter, $\mathcal{U}$ is recognized as $\mathcal{RO}$ and $\mathcal{V}$ is recognized as MD^h.

We denote the private interface of the system $\mathcal{X}$ by $\mathcal{X}^1$ and the public interface of the system $\mathcal{X}$ by $\mathcal{X}^2$. The definition of the indifferentiability is as follows.

Definition 1. *$\mathcal{V}$ is indifferentiable from $\mathcal{U}$, denote $\mathcal{V} \sqsubset \mathcal{U}$, if for any distinguisher $\mathcal{D}$ with binary output (0 or 1) there is a simulator $\mathcal{S}$ such that the advantage $|Pr[\mathcal{D}^{\mathcal{V}^1,\mathcal{V}^2} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{U}^1,\mathcal{S}(\mathcal{U}^2)} \Rightarrow 1]|$ is negligible in the security parameter k.*

This definition will allow us to use the construction MD^h instead of $\mathcal{RO}$ in *any* cryptosystem which is secure in the $\mathcal{RO}$ model and retains the same level of provable security due to the indifferentiability theory of Maurer et al. [11].

There exist a private interface of MD^h and a public interface of h in the MD^h model, while there exist both private and public interfaces of $\mathcal{RO}$ in the $\mathcal{RO}$ model.

2.3 Extension Attack

Coron et al. showed that MD^h is not indifferentiable from $\mathcal{RO}$ using the extension attack. The extension attack is the attack for MD^h where we can calculate a new hash value from some hash value. Namely $z' = MD^h(M||m)$ can be calculated from only z and m by $z' = h(z, m)$ where $z = MD^h(M)$. Note that z' can be calculated without using M. The distinguishing attack using the extension attack is as follows. Let $\mathcal{O}_1$ be MD^h or $\mathcal{RO}$ and let $\mathcal{O}_2$ be h or S. First, a distinguisher poses M to $\mathcal{O}_1$ and gets z from $\mathcal{O}_1$. Second, he poses (z, m) to $\mathcal{O}_2$ and gets c from $\mathcal{O}_2$. Finally, he poses $M||m$ to $\mathcal{O}_1$ and gets z' from $\mathcal{O}_1$.

If $\mathcal{O}_1 = MD^h$ and $\mathcal{O}_2 = h$, then $z' = c$, however, if $\mathcal{O}_1 = \mathcal{RO}$ and $\mathcal{O}_2 = S$, then $z' \neq c$. This is because no simulator can obtain the output value of $\mathcal{RO}(M||m)$ from just (z, m) and the output value of $\mathcal{RO}(M||m)$ is independently and randomly defined from c. Therefore, MD^h is *not* indifferentiable from $\mathcal{RO}$.

3 Variants of Random Oracles

In this section, we will introduce several variants of random oracles in order for $\mathcal{S}$ to simulate the extension attack described above, and show relationships among these oracles within the indifferentiability framework.

3.1 Motivation of New Primitives

In order for $\mathcal{S}$ to simulate the extension attack, it is helpful for $\mathcal{S}$ to obtain $z' = \mathcal{RO}(M||m)$ from (z, m) where $z = \mathcal{RO}(M)$. However, no $\mathcal{S}$ can know z' from (m, z) in the $\mathcal{RO}$ model. So we consider variants of random oracles by adding a new primitive which $\mathcal{S}$ can use in order to simulate the extension attack.

Random Oracle with additional Primitive($\widetilde{\mathcal{RO}}$) We can model $\widetilde{\mathcal{RO}}$ by combining both $\mathcal{RO}$ and an additional primitive I which outputs some information with which $\mathcal{S}$ can simulate the extension attack. Note that if $I = \mathsf{null}$ then $\widetilde{\mathcal{RO}}$ is $\mathcal{RO}$.

There exist a private interface with $\mathcal{RO}$ and two public interfaces of $\mathcal{RO}$ and I in the $\widetilde{\mathcal{RO}}$ model, while there exist both private and public interfaces of $\mathcal{RO}$ in the $\mathcal{RO}$ model. We can prove that $\mathcal{RO} \sqsubset \widetilde{\mathcal{RO}}$, where $\mathcal{S}$ just forwards queries to the public interface of $\mathcal{RO}$ and responds with $\mathcal{RO}$'s output.

Leaky Random Oracle ($\mathcal{LRO}$) The first model is *Leaky Random Oracle* ($\mathcal{LRO}$) model which was proposed by Yoneyama et al. [18]. $\mathcal{LRO}$ consists of $\mathcal{RO}$ and Leaky Oracle ($\mathcal{LO}$) which has the functions of leaking of all input-output pairs in the list of $\mathcal{RO}$. By using $\mathcal{LRO}$ as $\widetilde{\mathcal{RO}}$, we can construct $\mathcal{S}$ which can simulate the extension attack, since $\mathcal{S}$ can know M from z by calling $\mathcal{LO}$ and can know z' by querying $M||m$ to $\mathcal{RO}$. Thought they proved that FDH is secure in the $\mathcal{LRO}$ model, they did not discuss the indifferentiability for MD^h in [18].

Traceable Random Oracle ($\mathcal{TRO}$) Unfortunately, there are several practical cryptosystems which are secure in the $\mathcal{RO}$ model but insecure in $\mathcal{LRO}$ model. It was proved that OAEP is insecure in the $\mathcal{LRO}$ model in [18]. So we will introduce more suitable variant of $\mathcal{RO}$ than $\mathcal{LRO}$ where OAEP becomes secure.

$\mathcal{LRO}$ leaks too much information for simulating the extension attack. The important information for $\mathcal{S}$ in order to simulate the extension attack is the pair (M, z) in the list of $\mathcal{RO}$. As the second variant of $\mathcal{RO}$, we will propose a new primitive *Traceable Random Oracle* ($\mathcal{TRO}$) which consists of $\mathcal{RO}$ and Trace Oracle ($\mathcal{TO}$) as an additional oracle. For query z, $\mathcal{TO}$ returns M if the pair (M, z) exists in the list of $\mathcal{RO}$ where $z = \mathcal{RO}(M)$, and returns $\perp$ otherwise.

By using $\mathcal{TRO}$ as $\widetilde{\mathcal{RO}}$, we will construct $\mathcal{S}$ which can simulate the extension attack, since $\mathcal{S}$ obtains M by using $\mathcal{TO}$ and can know z' by querying $M||m$ to $\mathcal{RO}$. We will prove that OAEP is secure in the $\mathcal{TRO}$ model in Section 5.

We also prove that $\mathcal{TRO} \sqsubset \mathcal{LRO}$ and $\mathcal{LRO} \not\sqsubset \mathcal{TRO}$. This means that any secure cryptosystem in the $\mathcal{LRO}$ model is also secure in the $\mathcal{TRO}$ model and there exists some cryptosystem secure in the $\mathcal{TRO}$ model but insecure in the $\mathcal{LRO}$ model. Since it was proved that OAEP is insecure in the $\mathcal{LRO}$ model [18], OAEP is an evidence of the separation between $\mathcal{TRO}$ and $\mathcal{LRO}$ models. Note that FDH is secure in the $\mathcal{TRO}$ model since it is secure in the $\mathcal{LRO}$ model (see Appendix D).

Extension Attack Simulatable Random Oracle ($\mathcal{ERO}$) Again, there are however several practical cryptosystems which are secure in the $\mathcal{RO}$ model but insecure $\mathcal{TRO}$ model. We will show an concrete attack against RSA-KEM in the $\mathcal{TRO}$ model, that is, RSA-KEM is insecure there. Therefore we will introduce more suitable $\mathcal{RO}$ variant than $\mathcal{TRO}$ where RSA-KEM becomes secure.

$\mathcal{TRO}$ leaks too much information for simulating the extension attack yet. The important information for $\mathcal{S}$ is only the value z' such that $z' = \mathcal{RO}(M||m)$. Note that the value of the pair (M, z) is unnecessary for $\mathcal{S}$. As the third variant of $\mathcal{RO}$, we will propose a new primitive *Extension Attack Simulatable Random Oracle* ($\mathcal{ERO}$) which consists of $\mathcal{RO}$ and Extension Attack Simulatable Oracle ($\mathcal{EO}$) as an additional oracle. For query (m, z), if the pair (M, z) is in the list of $\mathcal{RO}$ where $z = \mathcal{RO}(M)$, then $\mathcal{EO}$ queries $M||m$ to $\mathcal{RO}$, receives z' and returns z', and otherwise $\mathcal{EO}$ returns $\perp$.

By using $\mathcal{ERO}$ as $\mathcal{RO}$, we will construct $\mathcal{S}$ which can simulate the extension attack, since S obtains $z' = \mathcal{RO}(M||m)$ by using $\mathcal{EO}$ without $(M||m)$. We can prove that RSA-KEM is secure in the $\mathcal{ERO}$ model in Section 6.

We also prove that $\mathcal{ERO} \sqsubset \mathcal{TRO}$ and $\mathcal{TRO} \not\sqsubset \mathcal{ERO}$. Therefore, RSA-KEM is an evidence of the separation between $\mathcal{ERO}$ and $\mathcal{TRO}$. Note that FDH and OAEP are secure in the $\mathcal{ERO}$ model because of the transitivity of the indifferentiability.

3.2 Definition of Variants of Random Oracles

The definition of $\mathcal{RO} : \{0,1\}^* \to \{0,1\}^n$ is as follows. $\mathcal{RO}$ has initially the empty hash list $\mathcal{L}_{\mathcal{RO}}$. On a query M, if $\exists (M, z) \in \mathcal{L}_{\mathcal{RO}}$, it returns z. Otherwise it chooses $z \in \{0,1\}^n$ at random, $\mathcal{L}_{\mathcal{RO}} \leftarrow (M, z)$ and returns z.

$\mathcal{LRO}$ was proposed by Yoneyama et al. [18]. The definition of $\mathcal{LRO}$ is as follows. $\mathcal{LRO}$ consists of $\mathcal{RO}$ and $\mathcal{LO}$. On a leak query to $\mathcal{LO}$, $\mathcal{LO}$ outputs all contents of $\mathcal{L}_{\mathcal{RO}}$. We can define $\mathcal{S}$ that can simulate the extension attack by using $\mathcal{LRO}$, since $\mathcal{S}$ can know M from z by using $\mathcal{LO}$ and can know z' by querying $M||m$ to $\mathcal{RO}$.

$\mathcal{LRO}$ leaks too much information for simulating the extension attack. The important information is the value M such that $z = \mathcal{RO}(M)$. So we define $\mathcal{TRO}$ as follows.

$\mathcal{TRO}$ consists of $\mathcal{RO}$ and $\mathcal{TO}$.

- $\mathcal{TO}$ can look into $\mathcal{L}_{\mathcal{RO}}$
- On a trace query z,
  - If there exist pairs such that $(M_i, z) \in \mathcal{L}_{\mathcal{RO}}$ $(i = 1, ..., n)$, it returns $(M_1, ..., M_n)$.
  - Otherwise it returns $\perp$.

We can define $\mathcal{S}$ that can simulate the extension attack by using $\mathcal{TRO}$, since $\mathcal{S}$ can know M from z by using $\mathcal{TO}$ and can know z' by querying $M||m$ to $\mathcal{RO}$.

$\mathcal{TRO}$ leaks too much information for simulating the extension attack yet. The important information is only the value z' such that $z' = \mathcal{RO}(M||m)$. Therefore we define $\mathcal{ERO}$ as follows.

$\mathcal{ERO}$ consists of $\mathcal{RO}$ and $\mathcal{EO}$. $\mathcal{EO}$ has initially the empty list $\mathcal{L}_{\mathcal{EO}}$ and can look into $\mathcal{L}_{\mathcal{RO}}$. On a simulate query (m, z) to $\mathcal{EO}$,

- If $(m, z, z') \in \mathcal{L}_{\mathcal{EO}}$, it returns z'.
- Else if there exists only one pair $(M, z) \in \mathcal{L}_{\mathcal{RO}}$, $\mathcal{EO}$ makes the query $M||m$ to $\mathcal{RO}$, receives z', $\mathcal{L}_{\mathcal{EO}} \leftarrow (m, z, z')$ and returns z'.
- Else $\mathcal{EO}$ chooses $z' \in \{0,1\}^n$ at random, $\mathcal{L}_{\mathcal{EO}} \leftarrow (m, z, z')$ and returns z'.

We can construct $\mathcal{S}$ that can simulate the extension attack by using $\mathcal{ERO}$, since $\mathcal{S}$ can obtain z' from (m, z) where $z' = \mathcal{RO}(M||m)$ by using $\mathcal{EO}$.

3.3 Relationships among $\mathcal{LRO}$, $\mathcal{TRO}$, and $\mathcal{ERO}$ models within the Indifferentiability Framework

$\mathcal{LRO}$ leaks more information of $\mathcal{L}_{\mathcal{RO}}$ than $\mathcal{TRO}$, and $\mathcal{TRO}$ leaks more information of $\mathcal{L}_{\mathcal{RO}}$ than $\mathcal{LRO}$. Therefore, it seems reasonable to suppose that any cryptosystem secure in the $\mathcal{LRO}$ model is also secure in the $\mathcal{TRO}$ model, and any cryptosystem secure in the $\mathcal{TRO}$ model is also secure in the $\mathcal{ERO}$ model. We prove validity of these intuitions by using the indifferentiability framework.

First we will clarify the relationship between $\mathcal{TRO}$ and $\mathcal{LRO}$

Theorem 1. *$\mathcal{TRO} \sqsubset \mathcal{LRO}$ and $\mathcal{LRO} \not\sqsubset \mathcal{TRO}$.*

Proof. We construct $\mathcal{S}$ which simulates $\mathcal{TO}$ by using $\mathcal{LRO}$ as follows. On query z, $\mathcal{S}$ makes a leak query to $\mathcal{LO}$ and receives $\mathcal{L}_{\mathcal{RO}}$. If there exists pairs such that $(M_i, z) \in \mathcal{L}_{\mathcal{RO}}$ $(i = 1, ..., n)$, it returns $(M_1, ..., M_n)$. Otherwise it returns $\perp$.

It is easy to see that $|Pr[\mathcal{D}^{\mathcal{RO},\mathcal{TO}} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{LRO})} \Rightarrow 1]| = 0$, since the output from each step of $\mathcal{S}$ is equal to that of each step of $\mathcal{TO}$.

$\mathcal{LRO} \not\sqsubset \mathcal{TRO}$ is trivial, since no $\mathcal{S}$ cannot know all values in $\mathcal{L}_{\mathcal{RO}}$ by using $\mathcal{TRO}$. □

Since $\mathcal{TRO} \sqsubset \mathcal{LRO}$, any cryptosystem secure in the $\mathcal{LRO}$ model is also secure in the $\mathcal{TRO}$ model by the indifferentiability framework. Since $\mathcal{LRO} \not\sqsubset \mathcal{TRO}$, there exists some cryptosystem which is secure in the $\mathcal{TRO}$ model but insecure in the $\mathcal{LRO}$ model. For example, Yoneyama et al. proved that OAEP is insecure in the $\mathcal{LRO}$ model [18]. We will prove that OAEP is secure in the $\mathcal{TRO}$ model in section 5. Therefore, OAEP is an evidence of the separation between $\mathcal{LRO}$ and $\mathcal{TRO}$.

Next we will clarify the relationship between $\mathcal{ERO}$ and $\mathcal{TRO}$.

Theorem 2. *$\mathcal{ERO} \sqsubset \mathcal{TRO}$ and $\mathcal{TRO} \not\sqsubset \mathcal{ERO}$.*

Proof. We construct $\mathcal{S}$ which simulates $\mathcal{EO}$ by using $\mathcal{TRO}$ as follows. $\mathcal{S}$ has initially the empty list $\mathcal{L}_S$ On query (m, z), If $\exists (m, z, z') \in \mathcal{L}_S$, it returns z'. Otherwise $\mathcal{S}$ makes a query z to $\mathcal{TO}$, and receives strings X. If X consists of one value, it makes a query $X||m$ to $\mathcal{RO}$, receives z', $\mathcal{L}_S \leftarrow (m, z, z')$ and returns z'. Otherwise it chooses $z' \in \{0, 1\}^n$ at random, $\mathcal{L}_S \leftarrow (m, z, z')$ and returns z'.

It is easy to see that $|Pr[\mathcal{D}^{\mathcal{RO},\mathcal{EO}} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{TRO})} \Rightarrow 1]| = 0$, since the output from each step of $\mathcal{S}$ is equal to that of each step of $\mathcal{EO}$.

$\mathcal{TRO} \not\sqsubset \mathcal{ERO}$ is trivial, since no $\mathcal{S}$ cannot decide whether there exists (M, z) in $\mathcal{L}_{\mathcal{RO}}$ or not by using $\mathcal{TRO}$. □

Since $\mathcal{ERO} \sqsubset \mathcal{TRO}$, any cryptosystem secure in the $\mathcal{TRO}$ model is also secure in the $\mathcal{ERO}$ model by the indifferentiability framework. Since $\mathcal{TRO} \not\sqsubset \mathcal{ERO}$, there exists some cryptosystem which is secure in the $\mathcal{ERO}$ model but insecure in the $\mathcal{TRO}$ model. We will prove that RSA-KEM is secure in the $\mathcal{ERO}$ model but insecure in the $\mathcal{TRO}$ model in section 6. Therefore, RSA-KEM is an evidence of the separation between $\mathcal{TRO}$ and $\mathcal{ERO}$.

From above discussions, the following corollary is obtained.

Corollary 1. *$\mathcal{RO} \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$, and $\mathcal{LRO} \not\sqsubset \mathcal{TRO} \not\sqsubset \mathcal{ERO}$.*

4 Indifferentiability from $\mathcal{ERO}$ for MD^h

In this section we prove $MD^h \sqsubset \mathcal{ERO}$ as the main theorem.

Theorem 3. *MD^h is (t_D, t_S, q, ϵ) indifferentiable from $\mathcal{ERO}$, for any t_D, with $t_S = O(lq)$ and $\epsilon = O(l^2q^2)/2^n$, where l is the maximum length of a query made by D where t_D is run time of D, t_S is run time of S and ϵ is the advantage of D.*

We only give a rough proof based on previous work in this section. The complete proof from scratch will be described in appendix A.

4.1 Previous related result

Coron *et al.* [6] and Chang *et al.* [5] have proven that so-called *Prefix-free MD* is indifferentiable from $\mathcal{RO}$. The definition of prefix-free MD is as follows:

Prefix-free MD is MD with a prefix-free padding, where for any two messages M_1 and M_2, Pad(M_1) is not prefix of Pad(M_2).

In the indifferentiability for prefix-free MD case, there are two types of message extension properties: type 1 and type 2. But in the indifferentiability for general MD case, there are four types of message extension properties: type 1, type 2, type 3 and type 4. Namely, the difference between prefix-free MD and MD is just eliminating the message extension properties of type 3 and type 4. Hereafter we will denote the message extension properties of all types as MEP for simplicity. So for any distinguisher, whose strategy is not related to MEP of MD denoted as $\mathcal{D}_{\neg MEP}$, the advantage on distinguishing MD from $\mathcal{RO}$ should be the same as that on distinguishing prefix-free MD from $\mathcal{RO}$. As a result, the advantage of $\mathcal{D}_{\neg MEP}$ on distinguishing MD from $\mathcal{RO}$ is negligible.

4.2 Our contributions based on previous result

In this section, we will give the rough proof based on previous result that the prefix-free MD is indifferentiable from $\mathcal{RO}$ [6] [5]. First We will focus on the differences between previous result and our expected result.

1. Previous result shows that the distinguishers $\mathcal{D}_{\neg MEP}$ have negligible success probability on distinguishing MD from $\mathcal{RO}$. In our proof, the $\mathcal{RO}$ is replaced by $\mathcal{ERO}$. We have to extend the previous result to $\mathcal{ERO}$. Thanks to the indifferentiability of $\mathcal{RO}$ from $\mathcal{ERO}$ and the transitive property of indifferentiability, we can automatically get that $\mathcal{D}_{\neg MEP}$ can not succeed in distinguishing MD from $\mathcal{ERO}$.
2. Previous result does not cover the distinguishers based on MEP, which will be denoted as $\mathcal{D}_{MEP}$. In our proof, we have to prove that $\mathcal{D}_{MEP}$ have negligible success probability on distinguishing MD from $\mathcal{ERO}$. This is the essential contributions of our work.

The advantages of $\mathcal{D}_{MEP}$
We will categorize all queries into trivial and non-trivial queries. Trivial queries might be helpful for arbitrary distinguisher (that is, $\mathcal{D}_{MEP}$ and $\mathcal{D}_{\neg MEP}$) to decide (MD^h, h) or $\mathcal{ERO} = (RO, EO)$, since a trivial query can provide some relation (as a collision) between queries on the private interface and ones no the public interface.

Trivial queries: Four types of trivial queries can be considered. In prefix-free MD case, type 3 and type 4 are only considered. However, in general MD case, there are not protection property by prefix-free padding. Therefore, we have to consider type 3 and type 4 in addition to type 1 and type 2. Trivial queries are defied as follows:

Type 1: r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}$, and r_i such that $r_{i_1} = (0, IV, m_{i_1}, y_{i_1})$, $r_{i_2} = (0, y_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$ and $r_i = (1, IV, M, H)$ where $M = m_{i_1}||...||m_{i_j}$ such that $i_1 < ... < i_j < i$.

Type 2: r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}, r_s$, and r_i such that $r_{i_1} = (0, IV, m_{i_1}, y_{i_1})$, $r_{i_2} = (0, y_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$, $r_s = (1, IV, M, H)$ and $r_i = (0, y_{i_j}, m_i, y_i)$ where $M = m_{i_1}||...||m_{i_j}||m_i$ such that $i_1 < ... < i_j < i$ and $s < i$.

Type 3: r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}$, and r_i such that $r_{i_1} = (1, IV, M_{i_1}, z_{i_1})$, $r_{i_2} = (0, z_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{i_j})$ and $r_i = (1, IV, M, H)$ where $M = M_{i_1}||...||m_{i_j}$ such that $i_1 < ... < i_j < i$.

Type 4: r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}, r_s$, and r_i such that $r_{i_1} = (1, IV, M_{i_1}, z_{i_1})$, $r_{i_2} = (0, z_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{i_j})$, $r_s = (1, IV, M, H)$ and $r_i = (0, y_{i_j}, m_i, y_i)$ where $M = M_{i_1}||m_{i_1}||...||m_{i_j}||m_i$ such that $i_1 < ... < i_j < i$ and $s < i$.

Thanks to the existence of the additional oracle $\mathcal{EO}$ in $\mathcal{ERO}$, the simulator $\mathcal{S}$ can simulate MEP for the trivial queries, which guarantees the consistence between $\mathcal{ERO}$ and $\mathcal{S}$. Pick Type 3 as an example, on the query $(0, z_{i_1}, m_{i_2}, y_{i_2})$, $\mathcal{S}$ will send the (z_{i_1}, m_{i_2}) to $\mathcal{EO}$. Then $\mathcal{EO}$ will check the existence of the pre-image of z_{i_1} and get the message M_{i_1}. Then $\mathcal{EO}$ will send $M_{i_1}||m_{i_2}$ to $\mathcal{RO}$ to get output y_{i_2}. Then $\mathcal{EO}$ responds to $\mathcal{S}$ with y_{i_2}. Finally $\mathcal{S}$ responds to $\mathcal{D}_{MEP}$ with the value y_{i_2}. From the above interaction between $\mathcal{S}$ and $\mathcal{ERO}$, we can get that the value $y_{i_2} = RO(M_{i_1}||m_{i_2})$. As a result, for Type 3 of trivial queries, $\mathcal{S}$ can make H be equal to y_{i_j}. So Type 3 of trivial queries can not help $\mathcal{D}_{MEP}$. Similarly, it is convinced we can get that Type 4 of trivial queries can not help $\mathcal{D}_{MEP}$.

Non-trivial query We can show by the same proof as [5] that the probability of collision for non-trivial queries is negligible, because all the responses are randomly generated here. Therefore, advantage of $\mathcal{D}_{MEP}$ using non-trivial queries must be negligible as well as that of $\mathcal{D}_{\neg MEP}$ in [5].

To summarize, for any distinguisher $\mathcal{D}$ ($\mathcal{D}_{\neg MEP}$ and $\mathcal{D}_{MEP}$), the advantage is negligible, so $MD^h \sqsubset \mathcal{ERO}$ has been proven. For more details, refer to Appendix A.

5 Security Analysis of OAEP in $\mathcal{TRO}$ Model

Optimal Asymmetric Encryption Padding (OAEP) encryption scheme [3] is a secure padding scheme for asymmetric encryptions in the $\mathcal{RO}$ model. In this section, we consider the security of OAEP encryption scheme in the $\mathcal{TRO}$ model.

5.1 Security Notion of Asymmetric Encryption Schemes

First, we briefly review the model and the security notion of asymmetric encryption schemes.

Definition 2 (Model for Asymmetric Encryption Schemes). *An asymmetric encryption scheme consists of the following 3-tuple* (**EGen**, **Enc**, **Dec**)*:*

EGen *: a key generation algorithm which on input 1^k, where k is the security parameter, outputs a pair of keys (ek, dk). ek and dk are called encryption key and decryption key, respectively.*
Enc *: an encryption algorithm which takes as input encryption key ek and message m, outputs ciphertext c.*
Dec *: a decryption algorithm which takes as input decryption key dk and ciphertext c, output message m.*

The security of asymmetric encryption schemes is defined by several notions like one-wayness and indistinguishability. Generally, indistinguishability under chosen ciphertext attacks (IND-CCA) is recognized as the strongest security notion. Here, we recall the definition of IND-CCA as follows.

Definition 3 (IND-CCA). *An asymmetric encryption scheme is (t, ϵ)-IND-CCA if the following property holds for security parameter k; For any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $|\Pr[\ (ek, dk) \leftarrow \mathbf{EGen}(1^k);\ (m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(dk,\cdot)}(ek);\ b \xleftarrow{R} \{0,1\};\ c^* \leftarrow \mathbf{Enc}(ek, m_b);\ b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(dk,\cdot)}(ek, c^*, state);\ b' = b] - 1/2| \leq \epsilon$, where $\mathcal{DO}$ is the decryption oracle, state is state information (possibly including ek, m_0 and m_1) which $\mathcal{A}$ wants to preserve, and $\mathcal{A}$ runs in at most t steps. $\mathcal{A}$ cannot submit the ciphertext $c = c^*$ to $\mathcal{DO}$.*

5.2 OAEP

OAEP encryption scheme is based on trapdoor partial-domain one-way permutations.

Definition 4 (Trapdoor partial-domain one-way permutation). *Let $\mathcal{G}$ be a trapdoor permutation generator. We say that a trapdoor permutation f is (t, ϵ)-partial-domain one-way if*

- *for input 1^k, $\mathcal{G}$ outputs (f, f^{-1}, Dom) where Dom is a subset of $\{0,1\}^{k_0} \times \{0,1\}^{k_1}$ $(k_0 + k_1 < k)$ and f, f^{-1} are permutations on Dom which are inverses of each other,*

– *there exist a polynomial p such that f, f^{-1} and Dom are computable in time $p(k)$, and*
– *for any adversary Alg,* $\Pr[(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k);\ (x_0, x_1) \overset{R}{\leftarrow} Dom;\ Alg(f,\ Dom,\ f(x_0, x_1)) = x_0] \le \epsilon$, *where Alg runs in at most t steps.*

The description of OAEP encryption scheme is as follows:

Key generation : For input k, outputs encryption key $(ek = f)$ and decryption key $(dk = f^{-1})$ such that $(f, f^{-1}, Dom = \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}) \leftarrow \mathcal{G}(1^k)$ where $\mathcal{G}$ is a trapdoor permutation generator and $n = k - k_0 - k_1$.

Encryption : Upon input of message $m \in \{0,1\}^n$, generates randomness $r \overset{R}{\leftarrow} \{0,1\}^{k_0}$, computes $x = (m||0^{k_1}) \oplus G(r)$ and $y = r \oplus H(x)$, and outputs ciphertext $c = f(x, y)$ where " $||$ " means concatenation, $H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$ and $G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$ are hash functions.

Decryption : Upon inputs of ciphertext c, computes $z = f^{-1}(c)$, parses z as (x, y) and reconstructs $r = y \oplus H(x)$ where $|x| = n + k_1$ and $|y| = k_0$. If $[x \oplus G(r)]_{k_1} \overset{?}{=} 0^{k_1}$ holds, outputs $m = [x \oplus G(r)]^n$ as the plaintext corresponding to c where $[a]^b$ denotes the b least significant bits of a and $[a]_b$ denotes the b most significant bits of a. Otherwise, rejects the input as an invalid ciphertext.

In [8], security of OAEP encryption scheme in the $\mathcal{RO}$ model is proved as follows;

Lemma 1 (Security of OAEP encryption scheme in the $\mathcal{RO}$ model [8]). *If the trapdoor permutation f is partial-domain one-way, then OAEP encryption scheme satisfies IND-CCA where H and G are modeled as $\mathcal{RO}$s.*

5.3 Insecurity of OAEP encryption scheme in $\mathcal{LRO}$ Model

Though OAEP encryption scheme is secure in the $\mathcal{RO}$ model, it is insecure in the $\mathcal{LRO}$ model. More specifically, it was shown that OAEP encryption scheme does not even satisfy OW-CPA in the $\mathcal{LRO}$ model.

Lemma 2 (Insecurity of OAEP in $\mathcal{LRO}$ model[18]). *Even if the trapdoor permutation f is partial-domain one-way, OAEP does not satisfy OW-CPA where H and G are modeled as $\mathcal{LRO}$s.*

5.4 Security of OAEP encryption scheme in $\mathcal{TRO}$ Model

We can also prove the security of OAEP encryption scheme in the $\mathcal{TRO}$ model as well as in the $\mathcal{RO}$ model.

Theorem 4 (Security of OAEP encryption scheme in the $\mathcal{TRO}$ model). *If a trapdoor permutation f is (t', ϵ')-partial-domain one-way, then OAEP encryption scheme satisfies (t, ϵ)-IND-CCA as follows:*

$$t' = t + q_{RG} \cdot q_{RH} \cdot perm,$$
$$\epsilon' \ge \frac{1}{q_{RH}} \cdot \Big(\frac{\epsilon}{2} - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}}\Big) - \frac{q_{TH}}{2^{k_0}},$$

where H and G are modeled as the $\mathcal{TRO}$, q_{RH} is the number of hash query to the $\mathcal{RO}$ of H, q_{TH} is the number of trace queries to the $\mathcal{TO}$ of H, q_{RG} is the number of hash queries to the $\mathcal{RO}$ of G, q_{TG} is the number of trace queries to the $\mathcal{TO}$ of G, q_D is the number of queries to the decryption oracle $\mathcal{DO}$ and perm is the computational cost of f.

We only explain the sketch of the proof in this section. The full proof will be described in appendix B.

Proof (Sketch).
In the $\mathcal{TRO}$ model, we have to estimate the influence of $\mathcal{TO}$ as follows:

- By trace queries, the adversary may obtain some information about the plaintext corresponding to the challenge,
- Trace queries may be useful to obtain additional information from $\mathcal{DO}$ than the $\mathcal{RO}$ model.

To win IND game, the adversary has two strategies. One is to pose the trace query $H(x^*)$ to $\mathcal{TO}$ of H and the trace query $x^* \oplus m'_b||0^{k_1}$ to $\mathcal{TO}$ of G for a guessed bit b' where x^* is used to generate the challenge ciphertext. If $\mathcal{TO}$ of G returns $\perp$, then the adversary can know $b' \neq b$, thus, the adversary can win the game. The probability that the adversary poses the trace query $H(x^*)$ to $\mathcal{TO}$ of H is bounded by $q_{TH} \cdot 2^{-k_0}$ because r^* is randomly chosen and G is $\mathcal{RO}$. The other is to pose the hash query r^* to $\mathcal{RO}$ of G or the trace query $G(r^*)$ to $\mathcal{TO}$ of G where r^* is used to generate the challenge ciphertext. We have to estimate the probability of both case that r^* is posed to $\mathcal{RO}$ of G and the case that $G(r^*)$ is posed to $\mathcal{TO}$ of G because these events may occur separately. The probability that the adversary poses the hash query r^* to $\mathcal{RO}$ of G is bounded by $q_{RG} \cdot 2^{-k_0}$ and the probability that the adversary poses the trace query $G(r^*)$ to $\mathcal{TO}$ of G is bounded by $q_{TG} \cdot 2^{-(n+k_1)}$ because r^* is unknown and G is $\mathcal{RO}$.

Furthermore, we have to consider whether information from $\mathcal{TO}$ gives the advantage to the CCA adversary or not. Though the CCA adversary can obtain some tuples in the hash lists of $\mathcal{RO}$s by trace queries, the hash lists themselves are not updated by these queries. Thus, the number of valid ciphertexts does not increase by trace queries. Hence, even if the adversary can use $\mathcal{TO}$, it is not useful to obtain additional information from $\mathcal{DO}$.

Therefore, we can show the security of OAEP encryption scheme satisfies IND-CCA by the similar proof as that in [8].

6 Security Analysis of RSA-KEM in $\mathcal{TRO}$ and $\mathcal{ERO}$ Models

RSA-based key encapsulation mechanism (RSA-KEM) scheme [17] is secure KEM scheme in the $\mathcal{RO}$ model. In this section, we consider the security of RSA-KEM in the $\mathcal{TRO}$ and $\mathcal{ERO}$ models.

6.1 Security Notion of KEM

First, we briefly review the model and the security notion of KEM schemes.

Definition 5 (Model for KEM Schemes).
A KEM scheme consists of the following 3-tuple $(\mathbf{KEM.Gen}, \mathbf{KEM.Enc}, \mathbf{KEM.Dec})$*:*

KEM.Gen *: a key generation algorithm which on input 1^k, where k is the security parameter, outputs a pair of keys (ek, dk). ek and dk are called encryption key and decryption key respectively.*

KEM.Enc *: an encryption algorithm which takes as input encryption key ek, outputs key K and ciphertext c.*

KEM.Dec *: a decryption algorithm which takes as input decryption key dk and ciphertext c, output key K.*

The security of KEM schemes is also defined by IND-CCA. Here, we recall the definition of IND-CCA for KEM.

Definition 6 (IND-CCA for KEM). *A KEM scheme is (t, ϵ)-IND-CCA for KEM if the following property holds for security parameter k; For any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $|\Pr[\,(ek, dk) \leftarrow \mathbf{KEM.Gen}(1^k);\ (state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(dk,\cdot)}(ek);\ b \xleftarrow{R} \{0,1\};\ (K_0^*, c_0^*) \leftarrow \mathbf{KEM.Enc}(ek);\ (K_1^*, c_1^*) \leftarrow \mathbf{KEM.Enc}(ek);\ b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(dk,\cdot)}(ek, (K_b^*, c_0^*), state);\ b' = b] - 1/2| \leq \epsilon$, where $\mathcal{DO}$ is the decryption oracle, state is state information which $\mathcal{A}$ wants to preserve from $\mathcal{A}_1$ to $\mathcal{A}_2$ and $\mathcal{A}$ runs in at most t steps. $\mathcal{A}$ cannot submit the ciphertext $c = c_0^*$ to $\mathcal{DO}$.*

6.2 RSA-KEM

The security of RSA-KEM is based on the RSA assumption.

Definition 7 (RSA assumption). *Let n be an RSA modulus that is the product of two large primes (p, q) for security parameter k and e be an exponent such that $gcd(e,\ \phi(n)) = 1$. We say that RSA problem is (t, ϵ)-hard if for any adversary Alg, $\Pr[y \leftarrow \mathbb{Z}_n;\ Alg(n, e, y) = x;\ y \equiv x^e \pmod n] \leq \epsilon$, where Alg runs in at most t steps.*

The description of RSA-KEM is as follows:

Key generation : For input k, outputs encryption key $(ek = (n, e))$ and decryption key $(dk = d)$ such that n is an RSA modulus that is the product of two large primes (p, q) for security parameter k, gcd$(e,\ \phi(n)) = 1$ and $ed \equiv 1 \pmod{\phi(n)}$.

Encryption : Generates randomness $r \overset{R}{\leftarrow} \mathbb{Z}_n$, computes $c = r^e \mod n$ and $K = H(r)$, and outputs ciphertext c and key K where $H : \mathbb{Z}_n \to \{0, 1\}^k$ is a hash function.

Decryption : Upon inputs of ciphertext c, computes $r = c^d \mod n$ and outputs $K = H(r)$.

In [17], security of RSA-KEM in the $\mathcal{RO}$ model is proved as follows;

Lemma 3 (Security of RSA-KEM in the $\mathcal{RO}$ model [17]). *If RSA problem is hard, then RSA-KEM satisfies IND-CCA for KEM where H is modeled as the $\mathcal{RO}$.*

6.3 Insecurity of RSA-KEM in $\mathcal{TRO}$ Model

Though RSA-KEM is secure in the $\mathcal{RO}$ model, it is insecure in the $\mathcal{TRO}$ model. More specifically, we can show RSA-KEM does not even satisfy IND-CPA for KEM in the $\mathcal{TRO}$ model. Note that, IND-CPA means IND-CCA without $\mathcal{DO}$.

Theorem 5 (Insecurity of RSA-KEM in the $\mathcal{TRO}$ model). *Even if RSA problem is hard, RSA-KEM does not satisfy IND-CPA for KEM where H is modeled as the $\mathcal{TRO}$.*

Proof. We construct an adversary $\mathcal{A}$ which successfully plays IND-CPA game by using the $\mathcal{TRO}$ H. The construction of $\mathcal{A}$ is as follows;

Input : (n, e) as the public key

Output : b'

Step 1 : Return *state* and receive (K_b^*, c_0^*) as the challenge. Ask the trace query K_b^* to H, obtain $\{r\}$.

Step 2 : For all r in $\{r\}$, check whether $r^e \overset{?}{\equiv} c_0^* \pmod n$. If there is r^* satisfying the relation, output $b' = 0$. Otherwise, output $b' = 1$.

We estimate the success probability of $\mathcal{A}$. When the challenge ciphertext c_0^* is generated, r^* such that $K_0^* = H(r^*)$ is certainly asked to H because c_0^* is generated obeying the protocol description. Thus, $\mathcal{L}_H$ contains (r^*, c_0^*, K_0^*). If (r^*, c_0^*, K_b^*) is not in $\mathcal{L}_H$, $b = 1$. Therefore, $\mathcal{A}$ can successfully plays the IND-CPA game.

□

6.4 Security of RSA-KEM in $\mathcal{ERO}$ Model

We can also prove the security of RSA-KEM in the $\mathcal{ERO}$ model as well as in the $\mathcal{RO}$ model.

Theorem 6 (Security of RSA-KEM in the $\mathcal{ERO}$ model). *If RSA problem is (t', ϵ')-hard, then RSA-KEM satisfies (t, ϵ)-IND-CCA for KEM as follows:*

$$t' = t + (q_{RH} + q_{EH}) \cdot expo,$$
$$\epsilon' \geq \epsilon - \frac{q_D}{n},$$

where H is modeled as the $\mathcal{ERO}$, q_{RH} is the number of hash query to the $\mathcal{RO}$ of H, q_{EH} is the number of hash queries to the $\mathcal{EO}$ of H, q_D is the number of queries to the decryption oracle $\mathcal{DO}$ and expo is the computational cost of exponentiation modulo n.

We only explain the sketch of the proof in this section. The full proof will be described in appendix C.

Proof (Sketch).
Firstly, we show that the transformation of the experiment of IND-CCA for RSA-KEM from $\mathsf{Exp0}$ to $\mathsf{Exp4}$ in Appendix C. By the step of the transformation, we can show that the extension attack query (x, y) of the hash value of the randomness r^* or $r^*||x$ corresponding to the challenge ciphertext to $\mathcal{EO}$ of H only gives negligible advantage to the adversary as Lemma 6 in Appendix C. Information the adversary can obtain by the query is not useful without information of r^* itself and the adversary can succeeds if the randomness is leaked. Next, we construct a reduction from RSA assumption to the transformed experiment of IND-CCA for RSA-KEM. For the reduction part, we need to describe the simulation of $\mathcal{EO}$. However, we construct the perfect simulation of $\mathcal{EO}$. Thus, we can show that RSA-KEM is secure by the similar proof as that in [17].

7 Another Approach to Rescue Merkle-Damgård

7.1 Picking OAEP as an Example to Warm Up

Among the cryptosystems OAEP, RSA-KEM, and FDH, we found that there exists one different point in the design of OAEP comparing with the other two cryptosystems: **the bit-length of the queries to the random oracle has been fixed**. We can get that the bit-length of the queries to G and H has been fixed to be k_0 and $(n + k_1)$ respectively. When G and H are instantiated to be Merkle-Damgård hash functions, the input messages of the hash functions are also restricted to be with fixed-length. So we can get that the Merkle-Damgård hash functions under the application of OAEP are in fact **Pre-Fix-Free** Merkle-Damgård. At the same time, Pre-Fix-Free Merkle-Damgård hash functions have been proven to be indifferentiable from Random Oracle [5, 6]. Consequently, the security of OAEP will be preserved when the underlying random oracle is replaced by Merkle-Damgård hash functions. The high-level overview is described as follows.

- **Specification of OAEP** transforms Merkle-Damgård hash functions to be Pre-fix-free Merkle-Damgård hash functions.
- Pre-fix-free Merkle-Damgård hash functions have been proven indifferentiable from Random Oracle.
- OAEP has been proven secure in the Random Oracle Model.

Based on the above three points, we can get that OAEP is secure in the Merkle-Damgård hash function model. More detail discussion will be shown in the next subsection.

7.2 New Approach: Utilizing the Specification of the Cryptosystems

Suppose we are dealing with the instantiation for a cryptosystem $\mathcal{C}$ following the Random Oracle Methodology. Moreover, based on the specification of $\mathcal{C}$, we luckily can derive *some* restriction, denoted as α, on the queries to the random oracle. For example, in OAEP, α is that the input length of G and H is the fixed length. We will try rescuing Merkle-Damgård hash functions utilizing the α. We modify the indifferentiability framework as follows.

Definition 8. *$\mathcal{V}$ is indifferentiable from $\mathcal{U}$ under the restriction α, denote $\mathcal{V} \sqsubset_\alpha \mathcal{U}$, if for any distinguisher $\mathcal{D}$ which queries to $\mathcal{RO}$ under the restriction α with binary output (0 or 1) there is a simulator $\mathcal{S}$ such that the advantage $|Pr[\mathcal{D}^{\mathcal{V}^1,\mathcal{V}^2} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{U}^1,\mathcal{S}(\mathcal{U}^2)} \Rightarrow 1]|$ is negligible in the security parameter k.*

The following theorem holds.

Theorem 7. *$MD^h \sqsubset_\alpha \mathcal{RO} \Rightarrow$ for any cryptosystem $\mathcal{C}$ holding the restriction α, $\mathcal{C}(MD^h)$ retains the same level of provable security for $\mathcal{C}(\mathcal{RO})$.*

From above theorem, any secure cryptosystem (e.g., OAEP) in which input length of the hash function is fixed is also secure under MD^h.

The proof is detailed in Appendix E.

8 Conclusion

In this paper, we have succeeded in proving that major cryptosystems including FDH, OAEP, and RSA-KEM are secure under MD^h adopting the indifferentiability framework.

Since $MD^h \not\sqsubset \mathcal{RO}$ was shown by Coron et al. due to the extension attack, which implies there exists at least one cryptosystem $\mathcal{C}^*$ such that $\mathcal{C}^*(\mathcal{RO})$ is secure but $\mathcal{C}^*(MD^h)$ is insecure. Thus our concern is whether major cryptosystem $\mathcal{C}_0$ satisfies that $\mathcal{C}_0(\mathcal{RO})$ is secure and $\mathcal{C}_0(MD^h)$ is insecure, or $\mathcal{C}_0(\mathcal{RO})$ is secure and $\mathcal{C}_0(MD^h)$ is still secure.

In this paper, we proposed two approaches to prove the positive result. The first one is

1. Find an ideal primitive $\widetilde{\mathcal{RO}}$ and $MD^h \sqsubset \widetilde{\mathcal{RO}}$ holds, and
2. Prove that cryptosystem $\mathcal{C}_0$ is secure in the $\widetilde{\mathcal{RO}}$ model.

The second one is

1. Derive restrictions on the queries to the private interface of $\mathcal{RO}/MD^h$ according to the specification of the protocol, and
2. Analyze the indifferentiability of MD^h from $\mathcal{RO}$ under the restriction on the queries to the private interface of $\mathcal{RO}/MD^h$.

Our results following the former approach are summarised as follows:,

1. $\mathcal{RO}$ variants such as $\mathcal{LRO}$, $\mathcal{TRO}$ and $\mathcal{ERO}$ are introduced for $\widetilde{\mathcal{RO}}$. $\mathcal{RO} \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$ holds.
2. $MD^h \sqsubset \mathcal{ERO}$. Therefore $MD^h \sqsubset \mathcal{ERO} \sqsubset \mathcal{TRO} \sqsubset \mathcal{LRO}$ holds, which is an answer to claim 1.
3. OAEP is secure in the $\mathcal{TRO}$ model and RSA-KEM is secure in the $\mathcal{ERO}$ model. It is also known that FDH is secure in the $\mathcal{LRO}$ model, which is an answer to claim 2.

By combining the second and third results, we can obtain the result that FDH, OAEP, and RSA-KEM are secure under the MD^h model adopting the indifferentiability framework. We can say that the *MD construction is still alive*!

Moreover $\mathcal{LRO} \not\sqsubset \mathcal{TRO} \not\sqsubset \mathcal{ERO}$ is proven, which implies that there is some cryptosystem secure in the $\mathcal{TRO}$ model but insecure in the $\mathcal{LRO}$ model, and some cryptosystem secure in the $\mathcal{ERO}$ model but insecure in the $\mathcal{TRO}$ model. It is also proven that OAEP is separating between $\mathcal{LRO}$ and $\mathcal{TRO}$, and RSA-KEM is separating between $\mathcal{TRO}$ and $\mathcal{ERO}$, which is interesting result regardless of indifferentiability from MD^h.

Acknowledgements

Thanks to reviewers of CT-RSA 2009 who provided useful comments on a previous version of this paper.

References

1. Mihir Bellare and Thomas Ristenpart: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. ASIACRYPT 2006: 299-314.
2. Mihir Bellare and Phillip Rogaway: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM Conference on Computer and Communications Security 1993: 62-73.
3. Mihir Bellare and Phillip Rogaway: Optimal Asymmetric Encryption. EUROCRYPT 1994: 92-111.
4. Mihir Bellare and Phillip Rogaway: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. EUROCRYPT 1996: 399-416.
5. Donghoon Chang, Sangjin Lee, Mridul Nandi and Moti Yung: Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. ASIACRYPT 2006: 283-298.
6. Jean-Sebastien Coron, Yevgeniy Dodis, Cecile Malinaud and Prashant Puniya: Merkle-Damgard Revisited: How to Construct a Hash Function. CRYPTO 2005: 430-448.
7. Ivan Damgård: A Design Principle for Hash Functions. CRYPTO 1989: 416-427.
8. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval and Jacques Stern: RSA-OAEP Is Secure under the RSA Assumption. CRYPTO 2001: 260-274
9. Shoichi Hirose, Je Hong Park and Aaram Yun: A Simple Variant of the Merkle-Damgard Scheme with a Permutation. ASIACRYPT 2007: 113-129.
10. Moses Liskov: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. Selected Areas in Cryptography 2006: 358-375.
11. Ueli M. Maurer, Renato Renner, Clemens Holenstein: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. TCC 2004: 21-39.
12. Ralph C. Merkle: One Way Hash Functions and DES. CRYPTO 1989: 428-446.
13. Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta: Indifferentiability of Hash Functions Revisited: Merkle-Damgård Is Still Alive. SCIS 2009, Japan.
14. Akira Numayama, Toshiyuki Isshiki and Keisuke Tanaka: Security of Digital Signature Schemes in Weakened Random Oracle Models. Public Key Cryptography 2008: 268-287.
15. Sylvain Pasini and Serge Vaudenay: Hash-and-Sign with Weak Hashing Made Secure. ACISP 2007: 338-354.
16. Bart Preneel, Rene Govaerts and Joos Vandewalle: Hash Functions Based on Block Ciphers: A Synthetic Approach. CRYPTO 1993: 368-378
17. Victor Shoup: A Proposal for an ISO Standard for Public Key Encryption.
18. Kazuki Yoneyama, Satoshi Miyagawa and Kazuo Ohta: Leaky Random Oracle: Provsec 2008: 226-240.

A Proof of Theorem 3

We will prove Theorem 3 using the same technique as that of [5] by extending the treatment of message extension of types 3 and 4.

We assume that no query of $\mathcal{D}$ is a repeated query in order to simplify the proof.

Let R_i be a binary relation of query-response pairs of a simulator after the i-th query;
$R_i = \{(x, m, y)|(0, x, m, y) \in L_0^i\}$. Let R_i^* be a closure relation on $\{0,1\}^n \times \{0,1\}^t \times \{0,1\}^n$ such that the following relations holds:

- $R_i \subset R_i^*$
- $(a, b, c), (c, d, e) \in R_i^* \Rightarrow (a, b||d, e) \in R_i^*$
- $(a, b||d, e), (a, b, c) \in R_i^* \Rightarrow (c, d, e) \in R_i^*$

Note that no simulator $\mathcal{S}$ can simulate the extension attack by using only $\mathcal{RO}$, because no $\mathcal{S}$ can obtain $z'(= \mathcal{RO}(M||m))$ on query (z, m) where $z = \mathcal{RO}(M)$.

We will construct simulator $\mathcal{S}$ that can simulate the extension attack using the Extension Attack Simulatable Oracle ($\mathcal{EO}$) at Step 3. As a result, the distinguishing based on the extension attack will fail (see the first case of $\mathsf{Coll1}, \mathsf{Coll2}, \mathsf{Coll3}$ and $\mathsf{Coll4}$ on Lemma 4).

Simulator. Initially, $R_0 = \phi$. The simulator keeps the relation R_{i-1} and the closure relation R_{i-1}^* before i-th query. On the i-th query (x, m), the response of $\mathcal{S}$ is as follows.

1. If $\exists(IV, M, x) \in R^*_{i-1}$, then run $\mathcal{RO}(M||m)$, obtain response y and return y.
2. Else, if $x = IV$, then run $\mathcal{RO}(m)$, obtain response y and return y.
3. Else, it runs $\mathcal{EO}(m, x)$, obtains response y, and returns y.

In the worst case of the simulator's running time, the simulator executes step 3 for every query and this requires at most $O(ql)$ time.

In the following proof, we use the following fact [5]: if $Pr[\mathcal{D}^{MD^h,h} \Rightarrow 1|\neg \mathsf{E1}] = Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{ERO})} \Rightarrow 1|\neg \mathsf{E2}]$ holds, then $|Pr[\mathcal{D}^{MD^h,h} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{ERO})} \Rightarrow 1]| \leq 2 \times max\{Pr[\mathsf{E1}], Pr[\mathsf{E2}]\}$ where $\mathsf{E1}$ and $\mathsf{E2}$ are certain events defined in Second stage.

We will evaluate $|Pr[\mathcal{D}^{MD^h,h} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{ERO})} \Rightarrow 1]|$ in three stages.

– In the first stage, we will define $\mathsf{BadColl}$ where $\mathcal{D}$ may distinguish (MD^h, h) from $(\mathcal{RO}, \mathcal{S})$.
– In the second stage, we will shows that no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{RO}, \mathcal{S})$ when $\neg\mathsf{BadColl}$ occurs, that is, let $\mathsf{E1}$ be an event $\mathsf{BadColl}$ when $\mathcal{D}$ interacts with (MD^h, h), and let $\mathsf{E2}$ be an event $\mathsf{BadColl}$ when $\mathcal{D}$ interacts with $(\mathcal{RO}, \mathcal{S})$. Then $Pr[\mathcal{D}^{MD^h,h} \Rightarrow 1|\neg \mathsf{E1}] = Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{ERO})} \Rightarrow 1|\neg \mathsf{E2}]$ holds.
– Finally, we will analyze the probabilities of $Pr[\mathsf{E1}]$ and $Pr[\mathsf{E2}]$.

First Stage: We define an event $\mathsf{BadColl}$.

In order to define this event, we will define a new set $IO(i)$ by $IO(i) = \{IV\} \cup \{x|(0, x, m, y) \in L^i_0\} \cup \{y|(0, x, m, y) \in L^i_0\} \cup \{H|(1, IV, M, H) \in L^i_1\}$. Coll covers all possibilities of collisions, that is, Coll is an event where there exists $r_i = (j, a, b, c)$ such that $c \in IO(i) \cup \{a\}$. We will also define several events $\mathsf{Coll1}, \mathsf{Coll2}, \mathsf{Coll3}$ and $\mathsf{Coll4}$ corresponding to following types of trivial queries. Trivial queries satisfy the message extension property, that is, $\mathcal{D}$ knows an input-output pair of h/S without making the query to h/S during the attack procedures but with only making the query to MD^h/RO.

We introduce four types of trivial queries as follows:

– **Type 1:** r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}$, and r_i such that $r_{i_1} = (0, IV, m_{i_1}, y_{i_1})$, $r_{i_2} = (0, y_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$ and $r_i = (1, IV, M, H)$ where $M = m_{i_1}||...||m_{i_j}$ such that $i_1 < ... < i_j < i$.
– **Type 2:** r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}, r_s$, and r_i such that $r_{i_1} = (0, IV, m_{i_1}, y_{i_1})$, $r_{i_2} = (0, y_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$, $r_s = (1, IV, M, H)$ and $r_i = (0, y_{i_j}, m_i, y_i)$ where $M = m_{i_1}||...||m_{i_j}||m_i$ such that $i_1 < ... < i_j < i$ and $s < i$.
– **Type 3:** r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}$, and r_i such that $r_{i_1} = (1, IV, M_{i_1}, z_{i_1})$, $r_{i_2} = (0, z_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$ and $r_i = (1, IV, M, H)$ where $M = M_{i_1}||...||m_{i_j}$ such that $i_1 < ... < i_j < i$.
– **Type 4:** r_i is the trivial query if there are $r_{i_1}, ..., r_{i_j}, r_s$, and r_i such that $r_{i_1} = (1, IV, M_{i_1}, z_{i_1})$, $r_{i_2} = (0, z_{i_1}, m_{i_2}, y_{i_2})$, ..., $r_{i_j} = (0, y_{i_{j-1}}, m_{i_j}, y_{j_j})$, $r_s = (1, IV, M, H)$ and $r_i = (0, y_{i_j}, m_i, y_i)$ where $M = M_{i_1}||m_{i_1}||...||m_{i_j}||m_i$ such that $i_1 < ... < i_j < i$ and $s < i$.

$\mathsf{Coll1}, \mathsf{Coll2}, \mathsf{Coll3}$, and $\mathsf{Coll4}$ are events where Coll occurs by trivial queries as follows:

– $\mathsf{Coll1}$: $y_{i_j} = H$ holds if r_i is the trivial query of **type 1**.
– $\mathsf{Coll2}$: $y_i = H$ holds if r_i is the trivial query of **type 2**.
– $\mathsf{Coll3}$: $y_{i_j} = H$ holds if r_i is the trivial query of **type 3**.
– $\mathsf{Coll4}$: $y_i = H$ holds if r_i is the trivial query of **type 4**.

Type 3 and type 4 are extension attacks shown in Section 2.3. Note that if $\mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4}$ occurs then Coll occurs. $\mathsf{BadColl}$ is defined by $\mathsf{BadColl} = \mathsf{Coll} \wedge \neg\mathsf{Coll1} \wedge \neg\mathsf{Coll2} \wedge \neg\mathsf{Coll3} \wedge \neg\mathsf{Coll4}$, since $\mathcal{S}$ takes care of trivial queries.

Second Stage: We will show that no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{ERO}, \mathcal{S})$ when $\neg\mathsf{BadColl}$ occurs.

Lemma 4. *When* $\neg\mathsf{BadColl}$ $(= \neg\mathsf{Coll} \vee \mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4})$ *occurs, no* $\mathcal{D}$ *can distinguish* (MD^h, h) *from* $(\mathcal{ERO}, \mathcal{S})$.

Proof. We prove that no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{ERO}, \mathcal{S})$ when $\neg\mathsf{BadColl}$ occurs.

– $\mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4}$:
  - When $\mathcal{D}$ interacts with (MD^h, h) and sends trivial queries of type 1, type 2, type 3, or type 4, then $\mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4}$ always occurs due to MD construction.
  - When $\mathcal{D}$ interacts with $(\mathcal{RO}, \mathcal{S})$ and sends trivial queries of type 1, type 2, type 3, or type 4, then $\mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4}$ always occurs due to $\mathcal{EO}$ and $\mathcal{S}$.This is because, for all trivial queries, $\mathcal{EO}$ returns the value to $\mathcal{S}$ so that $\mathsf{Coll1}, \mathsf{Coll2}, \mathsf{Coll3}$ and $\mathsf{Coll4}$ are true by looking into the list of $\mathcal{RO}$ and making an query to $\mathcal{RO}$ (see Section 4.2).

  Therefore no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{ERO}, \mathcal{S})$ when $\mathsf{Coll1} \vee \mathsf{Coll2} \vee \mathsf{Coll3} \vee \mathsf{Coll4}$ occurs.

– $\neg\mathsf{Coll}$:

  The transcript with (MD^h, h) and that with $(\mathcal{ERO}, \mathcal{S})$ are uniformly conditioned distributed on $\{0,1\}^n \setminus IO(i)$, because of $\neg\mathsf{Coll}$. The following proof is the same as the proof of lemma 1 of [5].
  - When distinguisher $\mathcal{D}$ interacts with $(\mathcal{ERO}, \mathcal{S})$, and $\neg\mathsf{Coll}$ occurs, all responses are different, since $\neg\mathsf{Coll}$ is the event where any response is different from other responses. Therefore, all response values are fresh values for the view of $\mathcal{D}$. Namely any i-th response value is randomly chosen from $\{0,1\}^n \setminus IO(i)$, since h is a random oracle compression function and $\neg\mathsf{Coll}$ occurs. Therefore any i-th response value is randomly chosen from $\{0,1\}^n \setminus IO(i)$.
  - When distinguisher $\mathcal{D}$ interacts with (MD^h, h), and $\neg\mathsf{Coll}$ occurs, any i-th response is is randomly chosen from $\{0,1\}^n \setminus IO(i)$. First we prove it for MD^h. We prove it by proving $\forall z, z' \in \{0,1\}^n \setminus IO(i)$: $Pr[MD^h(M) = z|\neg\mathsf{Coll}] = Pr[MD^h(M) = z'|\neg\mathsf{Coll}]$ where M is i-th query. Let $X = \{h \mid h : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^n \text{ s.t. } MD^h(M) = z \wedge h \text{ satisfies } IO(i)\}$ for z. Similarly define X' for z'. Now we can define a bijection ϕ between X and X' in the following way.
    * $\phi(h)(x,m) = h(x,m)$ if $\{x, h(x,m)\} \cap \{z, z'\} = \phi$.
    * $\phi(h)(z,m) = h(z',m)$ if $h(z',m) \notin \{z,z'\}$. Similarly, $\phi(f)(z',m) = h(z,m)$ if $h(z,m) \notin \{z,z'\}$.
    * $\phi(h)(x,m) = z'$ if $x \notin \{z,z'\}$ and $h(x,m) = z$. Similarly, $\phi(h)(x,m) = z$ if $x \notin \{z,z'\}$ and $h(x,m) = z'$.
    * $\phi(h)(z,m) = z$ if $h(z',m) = z'$.
    * $\phi(h)(z,m) = z'$ if $h(z',m) = z$.
    * $\phi(h)(z',m) = z$ if $h(z,m) = z'$.
    * $\phi(h)(z',m) = z'$ if $h(z,m) = z$.

    Now it is easy to check that $\phi(h)$ is well defined and it belong to X'. Here, we mainly interchange the role of z and z' in all possible cases of input and output keeping other values the same. Thus, given $MD^h(M) = z$, we should have $MD^{\phi(h)}(M) = z'$ keeping all other equalities fixed. Now it is also easy to check that this is a bijection as we can define the inverse function similarly. Thus, $|X| = |X'|$ and hence the probabilities are equal. We can prove similarly for the distribution of h. So skip the proof of this.

  Therefore no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{ERO}, \mathcal{S})$ when $\neg\mathsf{Coll}$ occurs.

Therefore, when $\neg\mathsf{BadColl}$ occurs, no $\mathcal{D}$ can distinguish (MD^h, h) from $(\mathcal{ERO}, \mathcal{S})$. □

Final Stage: We calculate $Pr[\mathsf{E1}]$ and $Pr[\mathsf{E2}]$.

Lemma 5. $Pr[\mathsf{E1}] = O(\frac{l^2 q^2}{2^n})$ *and* $Pr[\mathsf{E2}] = O(\frac{q^2}{2^n})$.

Proof. We assume that there is no trivial query. If a trivial query exists, then we decrease the probability since the trivial query does not help for $\mathcal{D}$ to find the collision due to Lemma 4.

First we estimate $Pr[\mathsf{E1}]$ by deriving the lower bound of $Pr[\neg\mathsf{E1}]$ by using a new event E_1 such that the implication of $\mathsf{E}_1 \Rightarrow \neg\mathsf{E1}$ holds (namely $Pr[\mathsf{E1}] = 1 - P[\neg\mathsf{E1}] \leq 1 - Pr[\mathsf{E}_1]$).

We will define an event E_1 wherein no collision occurs for h, that is, for any fresh input (x, m) of h, its output value $y = h(x, m)$ satisfies the following relations: $\forall(x', m', y') \in \mathsf{List}_h$: $y \neq x$, $y \neq x'$, and $y \neq y'$. Note that the input (x, m) is from MD^h or D. Since MD^h is the iterated hash function of h, $\mathsf{E}_1 \Rightarrow \neg\mathsf{E1}$. Therefore, $Pr[\neg\mathsf{E1}] \geq Pr[\mathsf{E}_1]$ holds.

Since h is a random oracle compression function, $Pr[\mathsf{E}_1] = (\frac{2^n-1}{2^n})(\frac{2^n-3}{2^n})\cdots(\frac{2^n-2q'-1}{2^n}) \geq 1 - \frac{q'^2}{2^n}$ where q' is the total number of calls of h. When all queries are queries for MD^h and the message block length for all queries is l, $q' \leq l \times q$. Therefore, $Pr[\neg\mathsf{E1}] \geq Pr[\mathsf{E}_1] \geq 1 - \frac{l^2q^2}{2^n}$. Therefore $Pr[\mathsf{E1}] \leq 1 - Pr[\mathsf{E}_1] = O(\frac{l^2q^2}{2^n})$.

Finally we calculate $Pr[\mathsf{E2}]$. When $\mathcal{D}$ interacts with $(\mathcal{RO}, \mathcal{S})$, all outputs of $\mathcal{RO}$ and $\mathcal{S}$ are chosen at random. This is because from assumptions $\mathcal{D}$ does not make trivial query and repeated query. Therefore, $Pr[\mathsf{E2}] = O(\frac{q^2}{2^n})$.

This completes the proof of Lemma 5. □

Therefore, $|Pr[\mathcal{D}^{MD^h,h} \Rightarrow 1] - Pr[\mathcal{D}^{\mathcal{RO},\mathcal{S}(\mathcal{ERO})} \Rightarrow 1]| \leq 2 \times max\{Pr[\mathsf{E1}], Pr[\mathsf{E2}]\} = O(\frac{l^2q^2}{2^n})$. This completes the proof of Theorem 3.

B Proof of Theorem 4

Proof. Let r^* be the random tape which is implicitly defined by c^* and m_b as follows;

$$r^* = H(x^*) \oplus y^* \text{ and } G(r^*) = x^* \oplus (m_b || 0^{k_1}).$$

Firstly, we transform the experiment of IND-CCA for OAEP with respect to the trace query to $\mathcal{TO}$ of H. Though in the initial experiment, $\mathsf{Exp0}$, an adversary $\mathcal{A}$ may pose the trace query $r^* \oplus y^*$ to $\mathcal{TO}$ of H, in the transformed experiment, $\mathsf{Exp1}$, $\mathcal{A}$ does not pose the trace query $r^* \oplus y^*$ to $\mathcal{TO}$ of H. Let $\mathsf{Succ0}$ and $\mathsf{Succ1}$ be probabilities that $\mathcal{A}$ succeeds to guess the bit b in $\mathsf{Exp0}$ and $\mathsf{Exp0}$, respectively. We consider the difference between advantages of $\mathcal{A}$ in two experiments.

In $\mathsf{Exp0}$, $\mathcal{A}$ has two scenarios: One is to pose the trace query $r^* \oplus y^*$ to $\mathcal{TO}$ of H to obtain x^* and the trace query $x^* \oplus m_{b'} || 0^{k_1}$ to $\mathcal{TO}$ of G for a guessed bit b'. If $\mathcal{TO}$ of G returns $\perp$, then the adversary can know $b' \neq b$, and finally wins the game. Note that, in this scenario, $\mathcal{A}$ does not pose the hash query r^* to $\mathcal{RO}$ of G or the trace query $G(r^*)$ to $\mathcal{TO}$ of G. The other is to pose the hash query r^* to $\mathcal{RO}$ of G or the trace query $G(r^*)$ to $\mathcal{TO}$ of G. $\mathcal{A}$ can obtain no advantage in others than the above scenarios because $G(r^*)$ is uniformly distributed in $\{0,1\}^{n+k_1}$ because G is $\mathcal{RO}$ and thus is independent from m_b.

The probability that the first scenario occurs is bounded by $q_{TH} \cdot 2^{-k_0}$ because G is $\mathcal{RO}$. On the other hand, in $\mathsf{Exp1}$, $\mathcal{A}$ has only the second scenario, i.e., to pose the hash query r^* to $\mathcal{RO}$ of G or the trace query $G(r^*)$ to $\mathcal{TO}$ of G, because $\mathsf{Exp1}$ aborts when $\mathcal{A}$ poses the trace query $r^* \oplus y^*$ to $\mathcal{TO}$ of H. Thus, the difference between advantages in $\mathsf{Exp0}$ and $\mathsf{Exp1}$ only consists of the event that $\mathcal{A}$ poses the trace query $r^* \oplus y^*$ to $\mathcal{TO}$ of H. Therefore, we obtain that $|\mathsf{Succ1} - \mathsf{Succ0}| \leq q_{TH} \cdot 2^{-k_0}$.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary which breaks IND-CCA of OAEP in the sense of $\mathsf{Exp1}$. We construct an inverter $\mathcal{I}$ which breaks the (t', ϵ')-partial-domain one-wayness of the trapdoor permutation f, i.e., for given (f, Dom, y) $\mathcal{I}$ outputs s such that $y = f(s,t)$. We assume that $\mathcal{A}$ does not repeat previous hash queries to the $\mathcal{TRO}$ H and G or previous decryption queries to the $\mathcal{DO}$. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be the local hash lists of H and G, respectively. $\mathcal{L}_H$ consists of tuples (δ_i, h_i) $(0 \leq i \leq q_{RH})$ and $\mathcal{L}_G$ consists of tuples (γ_i, g_i) $(0 \leq i \leq q_G)$. The concrete construction of $\mathcal{I}$ is as follows.

Input : (f, Dom, c^*) s.t. $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ and $c^* = f(x^*, y^*)$ for $(x^*, y^*) \overset{R}{\leftarrow} \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}$

Output : x^* s.t. $(x^*, y^*) = f^{-1}(c^*)$

Input public key : Send f to $\mathcal{A}$ as the input public key.

$\mathcal{RO}$ H simulation : When $\mathcal{A}$ poses a query δ_i to $\mathcal{RO}$ of H, then behave as follows:
For any $(\gamma_j, g_j) \in \mathcal{L}_G$, $\mathcal{I}$ computes $y = \gamma_j \oplus h_i$ and checks whether $c^* = f(\delta_i, y)$. If for some γ_j the relation holds, then $\mathcal{I}$ can obtain $\delta_i = x^*$ s.t. $(x^*, y^*) = f^{-1}(c^*)$.
<If $(\delta_i, *) \notin \mathcal{L}_H$ >
Generate $h_i \in \{0,1\}^{k_0}$. Add (δ_i, h_i) to $\mathcal{L}_H$ and return h_i to $\mathcal{A}$ as the answer.
<If $(\delta_i, *) \in \mathcal{L}_H$ >
Find h' corresponding to δ_i from $\mathcal{L}_H$ and return h' to $\mathcal{A}$ as the answer.

$\mathcal{RO}$ G simulation : When $\mathcal{A}$ poses a query γ_i to $\mathcal{RO}$ of G, then behave as follows:
For any $(\delta_j, h_j) \in \mathcal{L}_H$, $\mathcal{I}$ computes $y = \gamma_i \oplus h_j$ and checks whether $c^* = f(\delta_j, y)$. If for some γ_j the relation holds, then $\mathcal{I}$ can obtain $\delta_j = x^*$ s.t. $(x^*, y^*) = f^{-1}(c^*)$.
<If $(\gamma_i, *) \notin \mathcal{L}_G$ >
Generate $g_i \in \{0,1\}^{n+k_1}$. Add (γ_i, g_i) to $\mathcal{L}_G$ and return g_i to $\mathcal{A}$ as the answer.
<If $(\gamma_i, *) \in \mathcal{L}_G$ >
Find g' corresponding to γ_i from $\mathcal{L}_G$ and return g' to $\mathcal{A}$ as the answer.

$\mathcal{DO}$ simulation : When $\mathcal{A}$ poses a decryption query c_i to $\mathcal{DO}$, then behave as follows:
Find (δ, h) from $\mathcal{L}_H$ and (γ, g) from $\mathcal{L}_G$ such that $c_i = f(\delta, \gamma \oplus h)$ and $[g \oplus \delta]_{k_1} = 0^{k_1}$. If there is a pair satisfying the condition, then output $[g \oplus \delta]^n$ as the answer. Otherwise, return *reject* as the answer.

$\mathcal{TO}$ H simulation : When $\mathcal{A}$ poses a query h' to $\mathcal{TO}$ of H, then find h' from $\mathcal{L}_H$. If there are $\{(\delta, h')\}$ for h', then return all inputs $\{\delta\}$ to $\mathcal{A}$. Otherwise, return $\perp$.

$\mathcal{TO}$ G simulation : When $\mathcal{A}$ poses a query g' to $\mathcal{TO}$ of G, then find g' from $\mathcal{L}_G$. If there are $\{(\gamma, g')\}$ for g', then return all inputs $\{\gamma\}$ to $\mathcal{A}$. Otherwise, return $\perp$.

Challenge ciphertext : When $\mathcal{A}$ outputs (m_0, m_1), then choose random bit b and return c^* as the ciphertext of m_b.

Finalization : When $\mathcal{A}$ outputs b', if $\mathcal{I}$ obtained x^* by the simulation of $\mathcal{RO}$s, then output x^*. Otherwise, choose content δ from $\mathcal{L}_H$ and output δ.

We analyze the success probability of $\mathcal{I}$. In Finalization, if x^* has been asked by $\mathcal{A}$ to $\mathcal{RO}$ of H, $\mathcal{I}$ can obtain x^*. However, $\mathcal{A}$ can distinguish the simulation environment from the real environment in the following cases:

1. By replacing the challenge ciphertext with c^*, $\mathcal{A}$ notices that it is not valid challenge ciphertext.
2. $\mathcal{I}$ return *reject* for a decryption query in the simulation environment while the queried ciphertext is known to be valid and real $\mathcal{DO}$ returns the plaintext in the real environment.

In the former case, $\mathcal{A}$ has to pose r^* to $\mathcal{RO}$ of G or $x^* \oplus (m_b||0^{k_1})$ to $\mathcal{TO}$ of G because, if $\mathcal{A}$ does not ask either of these queries, $G(r^*)$ is uniformly distributed in $\{0,1\}^{n+k_1}$ and thus is independent from both m_0 and m_1. In the latter case, $\mathcal{I}$ fails to simulate $\mathcal{DO}$ for a ciphertext c such that $c = f(x, y)$, $x = (m||0^{k_1}) \oplus G(r)$ and $y = r \oplus H(x)$ only if either of events that r is asked to $\mathcal{RO}$ of G and x is asked to $\mathcal{RO}$ of H does not occur and c is valid.

We use the following events in order to analyze the success probability of $\mathcal{I}$:

- AskH : the event that query x^* has been asked to $\mathcal{RO}$ of H
- AskRG : the event that query r^* has been asked to $\mathcal{RO}$ of G
- AskTG : the event that query $x^* \oplus (m_b||0^{k_1})$ has been asked to $\mathcal{TO}$ of G
- RGBad : the event that query r^* has been asked to $\mathcal{RO}$ of G but the answer is different from $x^* \oplus (m_b||0^{k_1})$
- TGBad : the event that query $x^* \oplus (m_b||0^{k_1})$ has been asked to $\mathcal{TO}$ of G but the answer does not contain r^*

– DBad : the event that $\mathcal{I}$ fails to simulate $\mathcal{DO}$
– $\mathsf{Bad} = \mathsf{RGBad} \vee \mathsf{TGBad} \vee \mathsf{DBad}$

Note that RGBad implies AskRG and TGBad implies AskTG.

The success probability of $\mathcal{I}$ is obtained by the probability of AskH occurring that $\epsilon' \geq \Pr[\mathsf{AskH}] \cdot {q_{RH}}^{-1}$. Thus, we evaluate $\Pr[\mathsf{AskH}]$ by splitting AskH for the event Bad as follows:

$$\Pr[\mathsf{AskH}] = \Pr[\mathsf{AskH} \wedge \mathsf{Bad}] + \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}].$$

First, we evaluate the first term. Note that, $\mathsf{Bad} = \mathsf{RGBad} \vee \mathsf{TGBad} \vee \mathsf{DBad}$, $\Pr[\mathsf{RGBad}] \leq \Pr[\mathsf{AskRG}]$ and $\Pr[\mathsf{TGBad}] \leq \Pr[\mathsf{AskTG}]$ hold.

$$\begin{aligned}\Pr[\mathsf{AskH} \wedge \mathsf{Bad}] &= \Pr[\mathsf{Bad}] - \Pr[\mathsf{Bad} \wedge \neg\mathsf{AskH}] \\ &\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{RGBad} \wedge \neg\mathsf{AskH}] - \Pr[\mathsf{TGBad} \wedge \neg\mathsf{AskH}] - \Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}] \\ &\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{AskRG} \wedge \neg\mathsf{AskH}] - \Pr[\mathsf{AskTG} \wedge \neg\mathsf{AskH}] - \Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}].\end{aligned}$$

Since $r^* = H(x^*) \oplus y^*$ is unknown for $\mathcal{A}$ by $\neg\mathsf{AskH}$ in $Exp1$, $\Pr[\mathsf{AskRG} \wedge \neg\mathsf{AskH}] \leq q_{RG} \cdot 2^{-k_0}$ holds. Because of the same reason, $\Pr[\mathsf{AskTG} \wedge \neg\mathsf{AskH}] \leq q_{TG} \cdot 2^{-(n+k_1)}$ holds. The evaluation of $\Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}]$ is the same as the evaluation in [8] because $\mathcal{TO}$ does not update the hash lists $\mathcal{L}_H$ and $\mathcal{L}_G$ with any query to $\mathcal{TO}$. Thus, $\Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}] \leq q_D(2 \cdot 2^{-k_1} + (2q_{RG} + 1) \cdot 2^{-k_0})$ holds. Therefore, we obtain

$$\Pr[\mathsf{AskH} \wedge \mathsf{Bad}] \geq \Pr[\mathsf{Bad}] - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}}.$$

Next, we evaluate the second term. This evaluation is the same as the evaluation in [8] because we can evaluate it without splitting the event Bad as well as [8]. Therefore, we obtain

$$\Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] \geq \left(\frac{\epsilon}{2} + \frac{1}{2} - \Pr[\mathsf{Bad}]\right) - \frac{\Pr[\neg\mathsf{Bad}]}{2} = \frac{\epsilon - \Pr[\mathsf{Bad}]}{2}.$$

From the first and second terms, and the fact that $\Pr[\mathsf{Bad}] \geq 0$, we obtain

$$\Pr[\mathsf{AskH}] \geq \frac{\epsilon}{2} - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}}.$$

The time complexity of $\mathcal{I}$ is equal to the complexity of the inverter in [8]. Thus, $t' = t + q_{RG} \cdot q_{RH} \cdot perm$. □

C Proof of Theorem 6

Proof. Firstly, we transform the experiment of IND-CCA for RSA-KEM to the experiment where queries to $\mathcal{DO}$ and $\mathcal{EO}$ does not give any advantage to the adversary.

Let $\mathsf{Exp0}$ be the initial experiment and $\mathsf{Succ0}$ be the probability that an adversary $\mathcal{A}$ succeeds to guess the bit b in $\mathsf{Exp0}$. $\mathcal{A}$ receives (K_b^*, c_0^*) as the challenge such that $c_0^* = r^{*e}$ for r^*.

Let $\mathsf{Exp1}$ be the same experiment as $\mathsf{Exp0}$ except when $\mathcal{A}$ queried c_0^* to $\mathcal{DO}$ before receiving c_0^* as the challenge ciphertext. $\mathsf{Exp1}$ aborts in the above case. Let $\mathsf{Succ1}$ be the probability that $\mathcal{A}$ succeeds to guess the bit b in $\mathsf{Exp1}$ and E_1 be the event that the experiment aborts. Then, the probability that the event E_1 occurs is equal or lower than q_D/n because $\mathcal{A}$ has no information about the challenge. Thus, we obtain that $|\mathsf{Succ1} - \mathsf{Succ0}| \leq q_D/n$.

Let $\mathsf{Exp2}$ be the same experiment as $\mathsf{Exp1}$ except that the challenge (K_b^*, c_0^*) is generated in the beginning of the experiment. Let $\mathsf{Succ2}$ be the probability that $\mathcal{A}$ succeeds to guess the bit b in $\mathsf{Exp2}$. Then, we trivially obtain that $|\mathsf{Succ2} - \mathsf{Succ1}| = 0$ because the challenge is determined independently from the behavior of $\mathcal{A}$.

Let $\mathsf{Exp3}$ be the same experiment as $\mathsf{Exp2}$ except that $\mathcal{EO}$ returns randomly chosen value $z' \in \{0,1\}^k$ and adds (x, K_b^*, z') to the $\mathcal{EO}$ list $\mathcal{L}_{\mathcal{EO}}$ when $\mathcal{A}$ poses query (x, K_b^*) for some x to $\mathcal{EO}$. Then, we consider a distinguisher $\mathcal{D}$ which tries to distinguish $\mathsf{Exp2}$ from $\mathsf{Exp3}$.

Lemma 6. *If the output of $\mathcal{RO}$ H is independently chosen from the input, $\mathcal{D}$ cannot distinguish* Exp2 *from* Exp3.

Proof. We show that we can construct an algorithm $\mathcal{ALG}$ which can distinguish an output of $\mathcal{RO}$ H from a random value if there exists $\mathcal{D}$ which can distinguish Exp2 from Exp3. The concrete construction of $\mathcal{ALG}$ is as follows.

Step 1 : Simulate Exp3 for $\mathcal{D}$, as the adversary, except when $\mathcal{D}$ poses query (x, K_b^*) for some x to $\mathcal{EO}$.

Step 2 : On receiving query (x, K_b^*) for some x to $\mathcal{EO}$, forward $r^*||x$ to $\mathcal{RO}$ H, receive z as the output where z is $H(r^*||x)$ or a random value $rand$, and return z to $\mathcal{D}$.

Step 3 : If $\mathcal{D}$ decides that he interacts with Exp2, decide that z is $H(r^*||x)$. Otherwise, decide that z is $rand$.

The interface of $\mathcal{D}$ is identical with Exp2 when z is $H(r^*||x)$. Also, the interface of $\mathcal{D}$ is identical with Exp3 when z is $rand$. Therefore, if $\mathcal{D}$ succeeds, then $\mathcal{ALG}$ also succeeds.

□

Thus, Exp2 and Exp3 is indistinguishable for the adversary $\mathcal{A}$. Then, we obtain that $|\mathsf{Succ3} - \mathsf{Succ2}| = 0$.

Let Exp4 be the same experiment as Exp3 except when $\mathcal{A}$ queried r^* to $\mathcal{RO}$. Exp4 aborts in the above case. Let Succ4 be the probability that $\mathcal{A}$ succeeds to guess the bit b in Exp4 and E_4 be the event that the experiment aborts by this case. Then, to evaluate the probability that the event E_4 occurs, $\Pr[E_4]$, we show that $\Pr[E_4]$ is equal or lower than the probability that RSA problem is broken as follows.

Lemma 7. *If the event E_4 occurs with the probability ϵ'' in time t'', we can construct an inverter $\mathcal{I}$ that breaks RSA problem with the probability ϵ' in time t' as follows:*

$$t' = t'' + (q_{RH} + q_{EH}) \cdot expo,$$
$$\epsilon' = \epsilon''.$$

Proof. We assume that $\mathcal{A}$ does not repeat previous hash queries to the $\mathcal{ERO}$ H or previous decryption queries to the $\mathcal{DO}$. Let $\mathcal{L}_H$ be the local hash list of H. $\mathcal{L}_H$ consists of tuples (δ_i, y_i, h_i) $(0 \leq i \leq q_{RH} + q_D + q_{EH})$. Let $\mathcal{L}_{\mathcal{EO}}$ be the local $\mathcal{EO}$ list of H. $\mathcal{L}_{\mathcal{EO}}$ consists of tuples (δ_i, h_i, z_i) $(0 \leq i \leq q_{EH})$. The concrete construction of $\mathcal{I}$ is as follows.

Input : (n, e, y^*) s.t. n is RSA modulus, e is the exponent where $\gcd(e, \phi(n)) = 1$ and $y^* \overset{R}{\leftarrow} \mathbb{Z}_n$

Output : x^* s.t. $x^* \equiv {y^*}^d \pmod n$

Input public key : Send (n, e) to $\mathcal{A}$ in Exp4 as the input public key.

$\mathcal{DO}$ simulation : When $\mathcal{A}$ poses a decryption query y_i to $\mathcal{DO}$, then behave as follows:
Find (δ_i, y_i, h_i) from $\mathcal{L}_H$ such that $y_i = \delta_i^e$. If there is a tuple (δ_i, y_i, h_i) satisfying the condition, then return h_i as the answer. Otherwise, generate $h_i \in \{0,1\}^k$, add $(\emptyset, y_i, h_i)$ to $\mathcal{L}_H$ and return h_i as the answer.

$\mathcal{RO}$ simulation : When $\mathcal{A}$ poses a query δ_i to $\mathcal{RO}$, then behave as follows:

```
<If $y_i = y^*$ s.t. $y_i = \delta_i^e \mod n$ >
    Output $\delta_i$ as $x^*$ and halt.
<If $(\delta_i, *, h_i) \in \mathcal{L}_H$ >
    Return $h_i$ to $\mathcal{A}$ as the answer.
<If $(\delta_i, *, *) \notin \mathcal{L}_H$ and $(\emptyset, y_i, h_i) \in \mathcal{L}_H$ s.t. $y_i = \delta_i^e \mod n$ >
```

Replace $(\emptyset, y_i, h_i)$ to (δ_i, y_i, h_i) in $\mathcal{L}_H$ and return h_i to $\mathcal{A}$ as the answer.
<If $(\delta_i, *, *) \notin \mathcal{L}_H$ and $(\emptyset, y, h) \notin \mathcal{L}_H$ s.t. $y = \delta_i^e \mod n$ >
Compute $y_i = \delta_i^e \mod n$, generate $h_i \in \{0,1\}^k$, add (δ_i, y_i, h_i) and return h_i to $\mathcal{A}$ as the answer.

$\mathcal{EO}$ simulation : When $\mathcal{A}$ poses an extension attack query (δ_i, h_i) to $\mathcal{EO}$, then behave as follows: Find $(\delta_i, h_i, *)$ from $\mathcal{L}_{\mathcal{EO}}$. If there is a tuple (δ_i, h_i, z_i) satisfying the condition, then return z_i. Else if there is only one tuple $(\delta', *, h_i)$ in $\mathcal{L}_H$, then generate $z_i \in \{0,1\}^k$, compute $y_i = (\delta' || \delta_i)^e \mod n$, add $(\delta' || \delta_i, y_i, z_i)$ to $\mathcal{L}_H$ and (δ_i, h_i, z_i) to $\mathcal{L}_{\mathcal{EO}}$, and return z_i. Otherwise, generate $z_i \in \{0,1\}^k$ add (δ_i, h_i, z_i) to $\mathcal{L}_{\mathcal{EO}}$, and return z_i.

Challenge ciphertext : When $\mathcal{A}$ outputs $(state)$, then compute (K^*, y') by the encryption procedure and return (K^*, y^*) as the challenge.

We determine the success probability of $\mathcal{I}$. In $\mathcal{RO}$ simulation, if $y_i = y^*$ such that $y_i = \delta_i^e \mod n$ holds, $\mathcal{I}$ successfully breaks RSA problem. This event is same as E_4 in Exp4. Also, it is clear that $\mathcal{I}$ perfectly simulates Exp4 for $\mathcal{A}$. Therefore, we obtain

$$\epsilon' = \epsilon''.$$

$\mathcal{I}$ computes at most $q_{RH} + q_{EH}$ exponentiations modulo n. Thus, we obtain

$$t' = t'' + (q_{RH} + q_{EH}) \cdot expo.$$

□

Exp3 and Exp4 are identical until E_4 occurs. Thus, $|\mathsf{Succ4} - \mathsf{Succ3}| = \epsilon'$.

$\mathcal{A}$ can obtain no information about the random bit b because the key K_b^* is independent from information which $\mathcal{A}$ can obtain in Exp4. Therefore, $\mathsf{Succ4} = 1/2$. Since $\mathsf{Succ0} \leq |\mathsf{Succ1} - \mathsf{Succ0}| + |\mathsf{Succ2} - \mathsf{Succ1}| + |\mathsf{Succ3} - \mathsf{Succ2}| + |\mathsf{Succ4} - \mathsf{Succ3}| + \mathsf{Succ4}$, $\mathsf{Succ0} \leq \epsilon' + \frac{q_D}{n} + 1/2$. Hence, $\epsilon' \geq \epsilon - \frac{q_D}{n}$.

□

D Security Analysis of FDH in $\mathcal{LRO}$ Model

Full Domain Hash (FDH) [2] is secure signature scheme in the $\mathcal{RO}$ model. In this section, we recall the security of FDH in the $\mathcal{LRO}$ model.

D.1 Security Notion of Signature Schemes

First, we briefly review the model and the security notion of signature schemes.

Definition 9 (Model for Signature Schemes).
A signature scheme consists of following 3-tuple (**SGen**, **Sign**, **Ver**)*:*

SGen *: a key generation algorithm which on input 1^k, where k is the security parameter, outputs a pair of keys (vk, sk). vk and sk are called verification key and signing key respectively.*

Sign *: a signature generation algorithm which takes as input the signing key sk from a message m, outputs a signature σ.*

Ver *: a verification algorithm which takes as input the verification key vk, a message m and a signature σ, output a bit 1 or 0.*

Generally, we say that a signature scheme is secure if the scheme satisfies existentially unforgeability under adaptively chosen message attacks (EUF-ACMA). The definition of EUF-ACMA is as follows.

Definition 10 (EUF-ACMA).

A signature scheme is (t, ϵ)-EUF-ACMA if the following property holds for a security parameter k;

For any forger $\mathcal{F}$, $\Pr[(vk, sk) \leftarrow \mathbf{SGen}(1^k);\ (m, \sigma) \leftarrow \mathcal{F}^{\mathcal{SO}(sk,\cdot)}(vk);\ 1 \leftarrow \mathbf{Ver}(vk, m, \sigma)] < \epsilon$*, where $\mathcal{F}$ never posed the message m from $\mathcal{SO}$ and runs in at most t steps. ($\mathcal{F}$ can obtains signatures $\sigma_1, \ldots, \sigma_n$ of a $\mathcal{F}$'s own chosen messages $m_1, \ldots, m_n$ from a signing oracle $\mathcal{SO}$ and $\mathcal{F}$ cannot output a new pair of message m and its signature σ, where $m \notin \{m_1 \ldots, m_n\}$.)*

D.2 FDH

FDH is based on trapdoor one-way permutations. The description of FDH is as follows:

Key generation : For input k, output a signing key ($sk = f^{-1}$) and a verification key ($vk = f$) such that $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ where Dom is the domain of f and $\mathcal{G}$ is a trapdoor permutation generator.

Signature generation : For input a message $m \in \{0,1\}^*$, compute $y = H(m)$ and output a signature $\sigma = f^{-1}(y)$ where $H : \{0,1\}^* \to Dom$ is a hash function.

Signature verification : For inputs a message m and a signature σ, compute $y' = f(\sigma)$, verify $y' \overset{?}{=} H(m)$. If the verification is valid, output 1, otherwise, output 0.

In [2], security of FDH in the $\mathcal{RO}$ model is proved as follows;

Lemma 8 (Security of FDH in the $\mathcal{RO}$ model [2]). *If a trapdoor permutation f is one-way, then FDH satisfies EUF-ACMA where H is modeled as the $\mathcal{RO}$.*

D.3 Security of FDH in the $\mathcal{LRO}$ model

We can also obtain the security of FDH in the $\mathcal{LRO}$ model like in the $\mathcal{RO}$ model.

Lemma 9 (Security of FDH in the $\mathcal{LRO}$ model [18]). *If a trapdoor permutation f is one-way, then FDH satisfies EUF-ACMA where H is modeled as the $\mathcal{LRO}$.*

Proof. Let $\mathcal{F}$ be a forger which breaks EUF-ACMA of FDH. We construct an inverter $\mathcal{I}$ which breaks one-way security of the trapdoor permutation f, i.e., given (f, Dom, y) $\mathcal{I}$ outputs $f^{-1}(y)$. We suppose that $\mathcal{F}$ does not repeat the same query as previous hash queries to the leaky random oracle H or as previous signing queries to the signing oracle $\mathcal{SO}$. Let $\mathcal{L}_H$ be the local hash list of the leaky random oracle H. $\mathcal{L}_H$ consists of tuples $(x_i, H(x_i), z_i)$ $(0 \leq i \leq q_H + q_S)$ where z_i is an intermediate value.[3] The concrete construction of $\mathcal{I}$ is as follows. Note that, " $*$ " in a tuple $(x, *, *)$ means wildcard.

Input : (f, Dom, y) s.t. $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ and $y \overset{R}{\leftarrow} Dom$

Output : $f^{-1}(y)$

Initialization : $i^* \overset{R}{\leftarrow} \{0, q_H - 1\}$, $\mathcal{L}_H \leftarrow \perp$ ($\perp$ is *null string*), $i \leftarrow 0$.

Input public key : Send f to $\mathcal{F}$ as the input.

```
$\mathcal{RO}$ simulation : When $\mathcal{F}$ asks a hash query $x_i$ to $H$, then behave as follows:
    <If $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* \neq i)$ >
        Generate $z_i \in Dom$ and compute $w_i = f(z_i)$. Add $(x_i, w_i, z_i)$ to $\mathcal{L}_H$ and return $w_i$ to $\mathcal{F}$ as the answer.
    $i \leftarrow i + 1$.
    <If $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* = i)$ >
```

[3] The hash list which $\mathcal{F}$ can access has the different form (i.e., the hash list consists of $(x_i, H(x_i))$) than $\mathcal{L}_H$ because z_i is only used for the proof and does not appear in the real protocol.

Add $(x_i, y, error)$ to $\mathcal{L}_H$ and return y to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.
<If $(x_i, *, *) \in \mathcal{L}_H$ >
Find w' corresponding to x_i from $\mathcal{L}_H$ and return w' to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.

$\mathcal{SO}$ simulation : When $\mathcal{F}$ asks a signing query x_i to $\mathcal{SO}$, then behave as follows:
<If $(x_i, *, *) \in \mathcal{L}_H$ >
Find z' corresponding to x_i from $\mathcal{L}_H$. If $z' = error$, then abort. Otherwise, return z' to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.
<If $(x_i, *, *) \notin \mathcal{L}_H$ >
Generate $z_i \in Dom$ and compute $w_i = f(z_i)$. Add (x_i, w_i, z_i) to $\mathcal{L}_H$ and return z_i to $\mathcal{F}$ as the answer. $i \leftarrow i + 1$.

$\mathcal{LO}$ simulation : When $\mathcal{F}$ asks a leak query to H, then hand all pairs of input and output $\{(x, w)\}$ to $\mathcal{F}$.[4]

Finalization : When $\mathcal{F}$ outputs (x^*, σ^*), then check $y \stackrel{?}{=} f(\sigma^*)$. If $y = f(\sigma^*)$, then output σ^* as $f^{-1}(y)$. Otherwise, abort.

We show the success probability of $\mathcal{I}$.

In $\mathcal{LO}$ simulation, $\mathcal{I}$ has to return the hash list to $\mathcal{F}$ as this simulation is indistinguishable from the output of the leaky random oracle. Then, each output value w is uniformly distributing on Dom because z is uniformly chosen from Dom and f is a permutation. Thus, this simulation is perfect.

$\mathsf{Abort1}$ denote the event which $\mathcal{I}$ aborts for any query in $\mathcal{SO}$ simulation, $\mathsf{Abort2}$ denote the event which $\mathcal{I}$ aborts in Finalization and let $\mathsf{Abort} = \mathsf{Abort1} \vee \mathsf{Abort2}$. Then, we estimate the probability which $\mathcal{I}$ does not abort ($\Pr[\neg\mathsf{Abort1}]$ and $\Pr[\neg\mathsf{Abort2}]$).

By the simulation, the event which $\mathcal{I}$ aborts in $\mathcal{SO}$ simulation occurs with $\frac{1}{q_H+q_S}$ per every query to the signing oracle. Therefore, the probability that the event which $\mathcal{I}$ does not abort in $\mathcal{SO}$ simulation occurs for all queries to the signing oracle ($\Pr[\neg\mathsf{Abort1}]$) is $(1 - \frac{1}{q_H+q_S})^{q_S}$.

By the simulation, the event which $\mathcal{I}$ does not abort in Finalization when $\mathcal{I}$ does not abort in $\mathcal{SO}$ simulation occurs with $\frac{1}{q_H+q_S+1}$ because the event occurs only in the case of that $y = f(\sigma^*)$ holds.

Thus, we obtain

$$\begin{aligned}
\epsilon' &= \Pr[\mathbf{Ver}^{FDH}(x^*, \sigma^*, f) = 1 \wedge \neg\mathsf{Abort}] \\
&= \Pr[\mathbf{Ver}^{FDH}(x^*, \sigma^*, f) = 1 | \neg\mathsf{Abort}] \cdot \Pr[\neg\mathsf{Abort}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort1} \wedge \neg\mathsf{Abort2}] \\
&= \epsilon \cdot \Pr[\neg\mathsf{Abort1}] \Pr[\neg\mathsf{Abort2} | \neg\mathsf{Abort1}] \\
&= \epsilon \cdot \big(1 - \frac{1}{q_H + q_S}\big)^{q_S} \cdot \frac{1}{q_H + q_S + 1}
\end{aligned}$$

where $\mathbf{Ver}^{FDH}$ is the verification algorithm of FDH, ϵ' is the success probability of $\mathcal{I}$ and ϵ is the success probability of $\mathcal{F}$.

□

E Proof of Theorem 7

Proof. For any attacker $\mathcal{A}$ on the MD^h model, we will construct an attacker $\mathcal{A}'(= S(A))$ on the RO model, where the simulator S can guarantee that $MD^h \sqsubset_\alpha RO^S$. Based on $MD^h \sqsubset_\alpha RO^S$, we can get that the

[4] Note that, do not hand intermediate value z.

difference between the advantage of the attacker $\mathcal{A}'$ and that of $\mathcal{A}$ is negligible. If the cryptosystem $\mathcal{C}$ is secure in the RO model, the advantage of the attacker $\mathcal{A}$ is negligible. Then the advantage of the attacker $\mathcal{A}$ on MD^h model is also negligible. So the cryptosystem $\mathcal{C}$ is at least as secure in the MD^h model as in the RO model.

In fact the above proof is similar with the proof of Theorem 2.1 in [6] for the general framework.

$\square$