How to Construct Cryptosystems and Hash Functions in Weakened Random Oracle Models

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

[1] Mitsubishi Electric Corporation
[2] The University of Electro-Communications

Abstract. In this paper, we discuss how to construct secure cryptosystems and secure hash functions in weakened random oracle models.

The weakened random oracle model (WROM), which was introduced by Numayama et al. at PKC 2008, is a random oracle with several weaknesses. Though the security of cryptosystems in the random oracle model, ROM, has been discussed sufficiently, the same is not true for WROM. A few cryptosystems have been proven secure in WROM. In this paper, we will propose a new conversion that can convert *any* cryptosystem secure in ROM to a new cryptosystem that is secure in the first preimage tractable random oracle model FPT-ROM *without re-proof*. FPT-ROM is ROM without preimage resistance and so is the weakest of the WROM models. Since there are many secure cryptosystems in ROM, our conversion can yield many cryptosystems secure in FPT-ROM.

The fixed input length weakened random oracle model, FIL-WROM, introduced by Liskov at SAC 2006, reflects the known weakness of compression functions. We will propose new hash functions that are indifferentiable from RO when the underlying compression function is modeled by a two-way partially-specified preimage-tractable fixed input length random oracle model (TFILROM). TFILROM is FIL-ROM without two types of preimage resistance and is the weakest of the FIL-WROM models. The proposed hash functions are more efficient than the existing hash functions which are indifferentiable from RO when the underlying compression function is modeled by TFILROM.

Keywords: Random oracle model, variable input length weakened random oracle model, fixed input length weakened random oracle model, hash functions, indifferentiability.

1 Introduction

Most cryptographic schemes are designed by using ideal primitives such as a random oracle RO and an ideal cipher. For example, OAEP [3], PSS [4] and so on are secure in the RO model and PGV compression functions [16] are provably collision resistant compression functions in the ideal cipher model [5].

Recently, several weakened ideal models were considered [10, 15] and several cryptographic schemes were proposed that are secure in a weakened model. Liskov [10] first proposed weakened random oracle models for compression functions: fixed input length weakened random oracle model (FIL-WROM). In the models, adversaries are given sub-oracles in addition to RO. The sub-oracles return collisions, preimages and so on. He proposed the Zipper hash function which behaves like RO (indifferentiable from RO) even when an underlying compression function is modeled by FIL-WROM. Note that Hoch and Shamir [9] revised this model and proved that the Double pipe hash function and the parallel hash function are indifferentiable from RO even when an underlying compression function is modeled by FIL-WROM. Numayama et al. [15] extended this model from compression functions to hash functions creating the concept weakened random oracle model (WROM). They proposed several signature schemes secure in the WROM that are variants of FDH [2].

In this paper, we reconsider two models. For WROM, though the previous results [15] only considered variants of FDH, we propose a conversion that yields secure cryptosystems even if the underlying hash function is weakened. For FIL-WROM, we propose two hash functions that are indifferentiable from RO even if the underlying compression function is modeled FIL-WRO. Our schemes are more efficient than previous schemes.

1.1 Tools

Weakened Random Oracle Models: Let $\mathsf{FIL\text{-}RO} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be a fixed input length random oracle. Oracles in $\mathsf{FIL\text{-}WROM}$ are as follows:

- Fixed input length collision tractable oracle (FILCTO): On invoking the oracle with no input, the oracle returns values (x, x', y, y', z) uniformly chosen from a set of values $\{(x, x', y, y', z) | \mathsf{FIL\text{-}RO}(x, y) = z = \mathsf{FIL\text{-}RO}(x', y') \wedge (x, y) \neq (x', y')\}$.
- Fixed input length second preimage tractable oracle (FILSPTO): On query (x, y), it returns pair (x', y') uniformly chosen from a set $\{(x', y') | \mathsf{FIL\text{-}RO}(x, y) = \mathsf{FIL\text{-}RO}(x', y') \wedge (x, y) \neq (x', y')\}$.
- Fixed input length preimage tractable oracle (FILPTO): On query z, it returns pair (x, y) uniformly chosen from a set $\{(x, y) | \mathsf{FIL\text{-}RO}(x, y) = z\}$. If no such pair exists, it returns $\perp$.
- Bridge oracle (BrO): On query (x, z), it returns y chosen uniformly from a set $\{y | \mathsf{FIL\text{-}RO}(x, y) = z\}$. If no such pair exists, it returns $\perp$.
- Backward oracle (BO): On query (y, z), it returns x chosen uniformly from a set $\{x | \mathsf{FIL\text{-}RO}(x, y) = z\}$. If no such pair exists, it returns $\perp$.

The weakest model among WROMs is so-called two-way partially-specified preimage-tractable fixed input length random oracle model ($\mathsf{TFILROM}$), which is that an attacker is given $\mathsf{FIL\text{-}RO}$, BrO and BO. Because FILCTO, FILSPTO and FILPTO can be constructed from either BrO or BO and the relation among BrO and BO is independent.

Oracles in WROM are as follows:

- Collision tractable oracle (CO): On invoking the oracle with no input, it returns pair (M, M') chosen uniformly from a set $\{(M, M') | \mathsf{RO}(M) = \mathsf{RO}(M') \wedge M \neq M'\}$.
- Second-preimage tractable oracle (SPO): On query (M, z), it returns M' chosen uniformly from a set $\{M' | \mathsf{RO}(M) = z = \mathsf{RO}(M') \wedge M \neq M'\}$. If no such value exists, it returns $\perp$.
- First-preimage tractable oracle (FPO): On query z, it returns M from a set $\{M | \mathsf{RO}(M) = z\}$. If no such value exists, it returns $\perp$.

Note that the model wherein an attacker is given both RO and FPO is the weakest model among above models (first-preimage tractable random oracle model ($\mathsf{FPT\text{-}ROM}$)).

Indifferentiability: The indifferentiability framework generalizes the fundamental concept of the indistinguishability of two cryptosystems $\mathcal{C}(\mathcal{U})$ and $\mathcal{C}(\mathcal{V})$ where $\mathcal{C}(\mathcal{U})$ is the cryptosystem $\mathcal{C}$ that invokes the underlying primitive $\mathcal{U}$ and $\mathcal{C}(\mathcal{V})$ is the cryptosystem $\mathcal{C}$ that invokes the underlying primitive $\mathcal{V}$. $\mathcal{U}$ and $\mathcal{V}$ have two interfaces: public and private interfaces. Adversaries can access only the public interfaces and honest parties (e.g. the cryptosystem $\mathcal{C}$) can access only the private interface.

We denote the private interface of system $\mathcal{W}$ by $\mathcal{W}^{\mathsf{priv}}$ and the public interface of system $\mathcal{W}$ by $\mathcal{W}^{\mathsf{pub}}$. The definition of indifferentiability is as follows.

Definition 1. *$\mathcal{V}$ is $(t_D, t_{\mathsf{S}}, q, \epsilon)$ indifferentiable from $\mathcal{U}$, denote $\mathcal{V} \sqsubset \mathcal{U}$, if for any distinguisher D with binary output (0 or 1) there is a simulator S such that $|Pr[D^{\mathcal{V}^{\mathsf{priv}}, \mathcal{V}^{\mathsf{pub}}} \Rightarrow 1] - Pr[D^{\mathcal{U}^{\mathsf{priv}}, S(\mathcal{U}^{\mathsf{pub}})} \Rightarrow 1]| < \epsilon$. Simulator S has oracle access to $\mathcal{U}^{\mathsf{pub}}$ and runs in time at most t_{S}. Distinguisher D runs in time at most t_D and makes at most q queries. ϵ is negligible in the security parameter k.*

This definition will allow us to use construction $\mathcal{V}$ instead of $\mathcal{U}$ in *any* cryptosystem while retaining the same level of provable security (taken from the indifferentiability theory of Maurer et al. [12]). We denote the same level of provable security by $\mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$. Namely, we denote $\mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$ for the case that if $\mathcal{C}(\mathcal{U})$ is secure, then $\mathcal{C}(\mathcal{V})$ is secure. More strictly, $\mathcal{V} \sqsubset \mathcal{U} \Leftrightarrow \mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$ holds.

At Crypto 2005, Coron et al. applied this framework to hash functions [6]. Hoch and Shamir extended the framework of [6] to $\mathsf{FIL\text{-}WROM}$. Generalization of these frameworks are as follows.

Definition 2. *Let X be a model wherein any party is given $\mathsf{FIL\text{-}RO}$ and sub-oracles $O_1, ..., O_i$. H^f is $(t_D, t_{\mathsf{S}}, q, \epsilon)$ indifferentiable from RO when underlying compression function f is modeled by X, denote $H^f \underset{\mathsf{X}}{\sqsubset} \mathsf{RO}$, if for any distinguisher D with binary output (0 or 1) there is a simulator S such that $|Pr[D^{H^f, f, O_1, ..., O_i} \Rightarrow 1] - Pr[D^{\mathsf{RO}, \mathsf{S}(\mathsf{RO})} \Rightarrow 1]| < \epsilon$. Simulator S has oracle access to RO and runs in time at most t_{S}. Distinguisher D runs in time at most t_D and makes at most q queries. ϵ is negligible in the security parameter k.*

"f is modeled by X" means that f is FIL-RO and all parties are given oracles $f, O_1, ...,$ and O_i. S simulates $f, O_1, ..., O_i$ by using RO. When X is TFILRO, $i = 2$, O_1 is BO and O_2 is BrO. If $H^f \underset{\mathsf{X}}{\sqsubset}$ RO, then any cryptosystem secure in RO is also secure under H^f. Namely, hash functions that satisfy this property behave like RO. This property considers the structures of hash functions. Therefore, this property bridges RO and hash functions. Coron et al. showed that the Merkle-Damgård (MD) construction [7, 13] does not satisfy this property when the underlying primitive is a fixed input length random oracle (FIL-RO). Several constructions that in FIL-ROM is indifferentiable from RO have been proposed. For example, Prefix-free Merkle-Damgård (PMD) [6], Enveloped MD (EMD) [1], MD with permutation (MDP) [8] and so on are indifferentiable from RO when the underlying compression function is FIL-RO.

1.2 Our Goal

Again, this paper proposes both a conversion of cryptosystems and constructions of hash functions from the most-weakened random oracle models as follows.

1. We propose new conversion conv that converts *any* cryptosystem secure in ROM to a new cryptosystem that is secure in FPT-ROM.
2. We propose two hash constructions that are indifferentiable from RO when the underlying compression function is modeled by TFILROM.

1.3 First Goal: Constructing Secure Cryptosystems in FPT-ROM

Previous Results and Problem: Numayama et al. succeeded in constructing secure signature schemes in WROM by modifying RSA-FDH. First, they considered the attack on RSA-FDH in the model where RO and CO are available; they recognized RSA-PFDH as the patch against this attack. RSA-FDH is existentially forgeable against adaptive chosen message attack in this model as follows. (1) The forger obtains collision messages m and m' from CO. (2) He poses query m to the signing oracle and get the signature $\sigma = RSA^{-1}(\mathsf{RO}(m))$. Since the signatures of collision messages m and m' are same, signature σ is also the signature of m'. Therefore, they considered a variant of RSA-FDH that resists this attack. They picked up RSA-PFDH. Signature σ of message m for RSA-PFDH is $RSA^{-1}(\mathsf{RO}(m||r))$ where r is a random value chosen by the signer. When the above attack is applied to RSA-PFDH, CO seems to be not helpful for attacking RSA-PFDH. The reason is that the probability that $r = r_1$ holds is negligible, since collision messages $m||r_1$ and $m'||r_1'$ given by CO are chosen before making the signature. They proved that RSA-PFDH is secure in this model. They also considered variants of RSA-FDH that are secure in other models of WROMs by similar discussions. However their modifications *depend* on RSA-FDH. It is not known if their modification techniques can be applied to other cryptosystems generally.

Our Conversion: In this paper, we propose the new conversion conv; it can convert *any* cryptosystem secure in ROM to a new cryptosystem that is secure in FPT-ROM. The advantage of the new conversion is that it is unnecessary to reprove the security of the converted cryptosystem. Let $C(\mathsf{RO}(\cdot))$ be a cryptosystem. Modified cryptosystem $\mathsf{conv}(C)$ is $C(\mathsf{RO}(\cdot||c))$ such that c is an s bit constant value such that $\frac{1}{2^s}$ is negligible. Roughly speaking, since the outputs of FPO are randomly chosen, the probability that the last s bits of some output of FPO are equal to c is $\frac{1}{2^s}$. Since $\frac{1}{2^s}$ is negligible, the outputs of FPO are useless for attacking the converted cryptosystem. We propose conversion conv by using the indifferentiability with condition that was proposed by Naito et al [14].

Indifferentiability with Condition [14]: Since cryptosystems access RO while adversaries access RO, RO has public and private interfaces. Since adversaries access FPO, FPO has only the public interface. The indifferentiability with condition [14] is the extended indifferentiability framework wherein restricting the queries of D on the private interface $\mathsf{RO}^{\mathsf{priv}}$ by the following condition.

Definition 3 (Indifferentiability with Condition). FPT-RO *is* $(t_D, t_{\mathsf{S}}, q, \epsilon)$ *indifferentiable from* RO *with condition* α*, denote* FPT-RO $\sqsubset_\alpha$ RO*, if for any distinguisher D with binary output (0 or 1) such that all queries to* $\mathsf{RO}^{\mathsf{priv}}$ *are restricted by condition α there is a simulator* S *such that* $|Pr[D^{\mathsf{RO}^{\mathsf{priv}},\mathsf{RO}^{\mathsf{pub}},\mathsf{FPO}^{\mathsf{pub}}} \Rightarrow$

$1] - Pr[D^{\mathsf{RO}^{\mathsf{priv}},\mathsf{S}(\mathsf{RO}^{\mathsf{pub}})} \Rightarrow 1]| < \epsilon$. *Simulator* S *has oracle access to* $\mathsf{RO}^{\mathsf{pub}}$ *and runs in time at most* t_{S}. *Distinguisher* D *runs in time at most* t_D *and makes at most* q *queries.* ϵ *is negligible in the security parameter* k.

Note that S simulates public interfaces $\mathsf{RO}^{\mathsf{pub}}$ and $\mathsf{FPO}^{\mathsf{pub}}$ such that no D can distinguish $(\mathsf{RO}, \mathsf{S})$ from $\mathsf{FPT\text{-}RO}$. This definition will allow us to use construction $\mathsf{FPT\text{-}RO}$ instead of RO in *any* cryptosystem $\mathcal{C}_\alpha$ such that the interface between $\mathcal{C}_\alpha$ and $\mathsf{RO}^{\mathsf{priv}}$ is restricted by condition α while retaining the same level of provable security. More exactly, $\mathsf{FPT\text{-}RO} \sqsubset_\alpha \mathsf{RO} \Leftrightarrow \mathcal{C}_\alpha(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathcal{C}_\alpha(\mathsf{RO}^{\mathsf{priv}})$ holds. Note that procedures of $\mathsf{RO}^{\mathsf{priv}}$ and $\mathsf{RO}^{\mathsf{pub}}$ are the same as RO itself and the procedures of $\mathsf{FPO}^{\mathsf{pub}}$ are the same as FPO itself.

How to Use Indifferentiability with Condition: Our proposal, conversion conv, is defined by the following procedure.

1. Select condition α such that $\mathsf{FPT\text{-}RO} \sqsubset_\alpha \mathsf{RO}$ holds.
2. Check that, for any cryptosystem $\mathcal{C}$, $\mathcal{C}_\alpha(\mathsf{RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$ holds where $\mathcal{C}_\alpha$ is such that $\mathcal{C}$ is restricted by condition α with regard to accessing the private interface.

The above procedure yields condition α such that no D can distinguish $(\mathsf{RO}, \mathsf{S})$ from $\mathsf{FPT\text{-}RO}$. If we can find condition α that passes procedure 1, for any system $\mathcal{C}_\alpha$, $\mathcal{C}_\alpha(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathcal{C}_\alpha(\mathsf{RO}^{\mathsf{priv}})$ holds. If condition α passes procedure 2, we can automatically get the result that $\mathcal{C}_\alpha(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$ holds. We regard the approach of adding a condition to a cryptosystem as conversion.

How to Select Condition α: Since no S can see the input-output list of RO, the probability that S returns a response on the list is negligible. Therefore, D can distinguish $(\mathsf{RO}, \mathsf{S})$ from $\mathsf{FPT\text{-}RO}$ by the following procedure: (1) D poses query z to $\mathsf{FPO}^{\mathsf{pub}}/\mathsf{S}$ and receives M. (2) D poses query M to $\mathsf{RO}^{\mathsf{priv}}$ and receives z'. (3) If $z = z'$, D outputs 1, otherwise 0. When D interacts with $\mathsf{FPT\text{-}RO}$, D outputs 1 with probability of 1. On the other hand, when D interacts with $(\mathsf{RO}^{\mathsf{priv}}, \mathsf{S})$, D explicitly outputs 0 with overwhelming probability. Therefore, distinguisher D can distinguish RO from $\mathsf{FPT\text{-}RO}$. We select condition α such that the above attack does not work. Namely, condition α is such that in procedure 2 no D can poses query M obtained from procedure 1. Since the outputs of FPO are randomly chosen, the probability that the last s bits of some output are equal to s bit constant value c is $\frac{1}{2^s}$. Therefore, we select condition α wherein the last s bits of the inputs to $\mathsf{RO}^{\mathsf{priv}}$ are constant value c such that $\frac{1}{2^s}$ is negligible. We will prove $\mathsf{FPT\text{-}RO} \sqsubset_\alpha \mathsf{RO}$ as well as $\mathcal{C}_\alpha(\mathsf{RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$. Therefore, $\mathsf{conv}(\mathcal{C})(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$ holds.

1.4 Second Goal: Constructing Secure Hash Functions from TFILRO

Previous Results and Problems: Liskov proposed the Zipper hash function $\mathsf{ZHF}^{g,h}$ that is indifferentiable from RO, when the underlying compression function is modeled by TFILRO. However, there is a problem in the $\mathsf{FIL\text{-}WROM}$ defined in [10]. The probability that BO returns $\perp$ is $\frac{1}{e}$. However, the author of [10] ignored the case that BO returns $\perp$. Accordingly, we revisit the indifferentiable security of the Zipper hash function in $\mathsf{TFILROM}$. We show that $\mathsf{ZHF}^{g,h} \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$ holds. This distinguishing attack uses the fact that BO returns $\perp$ with probability of $\frac{1}{e}$.

The double pipe hash function DPHF^f was proposed by Lucks [11] and Hoch and Shamir [9] proved that $\mathsf{DPHF}^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ holds. A compression function is called two times per message block in Double pipe hash function. Usually a compression function is called just one time (e.g. MD, EMD, and so on) which means that DPHF is not efficient [3]. In [9], parallel hash function $\mathsf{PHF}^{g,h}$ is proposed where $\mathsf{PHF}^{g,h} \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ holds. Since two types of compression functions are called for one block message, this compression function is not efficient.

[3] Note that DPHF was proposed for a different motivation. The grace of DPHF is that DPHF resists several types of brute force attack. Our schemes do not target these attacks, rather our goal is to propose constructions that are indifferentiable from RO when the underlying compression function is modeled by $\mathsf{TFILROM}$.

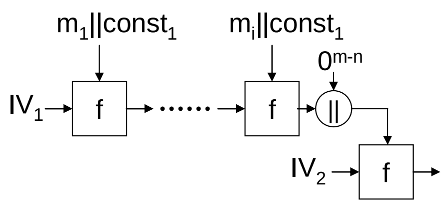

Fig. 1. $\mathsf{EMD}+^f$

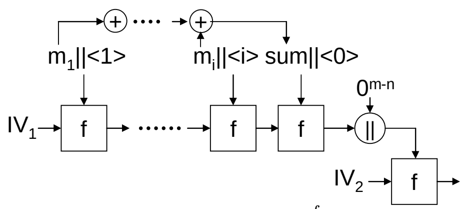

Fig. 2. $\mathsf{EMD}\times^f$

Our Goal: We propose two new constructions that are more efficient than DPHF^f and $\mathsf{PHF}^{g,h}$, and that are indifferentiable from RO when the underlying compression function is modeled by TFILRO. Namely, we propose constructions such that a compression function is called one time per block message and only one type of compression function is used.

Toward Our Goal: In order to realize our constructions, we need to overcome following three obstacles: the distinguishing attack using FIL-RO (length extension attack), the distinguishing attack using BO, and the distinguishing attack using BrO.

First, we search for candidates that resist the distinguishing attack using FIL-RO. Existing constructions such as PMD, EMD and so on resist the distinguishing attack using FIL-RO. We select these constructions as the first candidates.

Second, we choose a second round candidate from these constructions such that the distinguishing attack using BO is resisted. Before selecting a candidate, we analyze the reason why DPHF^f resists the distinguishing attack using BO. The reason for this is that this hash function has a HMAC-like structure (enveloped structure). The structure that has a HMAC-like structure among the first round candidates is the EMD construction. Therefore, we choose the EMD construction.

As a remaining work, we modify the EMD construction to resist the distinguishing attack using BrO.

First Scheme (Fig.1): In order to resist the distinguishing attack using BrO, we use the idea of the above conversion conv for the first scheme $\mathsf{EMD}+^f$. We define $\mathsf{EMD}+^f$ wherein the d bit constant value is input in each block of EMD^f. Since outputs of BrO are randomly chosen, the probability that a random value is equal to the d bit constant value is $O(\frac{1}{2^d})$. We select the constant value such that $O(\frac{1}{2^d})$ is negligible, so EMD+ resists the distinguishing attack that uses BrO. We will prove that $\mathsf{EMD}+^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ holds.

Second Scheme (Fig.2): In the second scheme, we use two techniques: the check sum operation and the block index input operation. The check sum operation is the input of value $sum = m_1 \oplus \cdots \oplus m_i$ to a compression function where $m_1, ..., m_i$ are input messages to the compression functions. The block index input operation inputs an index of a compression function to each compression function. Since the outputs of BrO are randomly chosen, the probability that messages satisfying $sum = m_1 \oplus \cdots \oplus m_i$ are found by using BrO is negligible and the probability that an output of BrO includes required index value is negligible. Therefore, the new construction $\mathsf{EMD}\times^f$, the result of combining EMD, the check sum operation and the block index input operation, resists the three attacks. We will prove that $\mathsf{EMD}\times^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ holds.

2 Preliminaries

2.1 Notation

Let $r \xleftarrow{\$} R$ represent that element r is selected from set R with uniform distribution. Let $a||b$ be the concatenated value of a and b. $\oplus$ is the bitwise exclusive-or operation, $\mathsf{lsb}_a(b)$: the least significant a bits of b, $\mathsf{msb}_a(b)$: the most significant a bits of b, 0^i is i bit value wherein each bit is 0. FPT-RO is First-preimage tractable random oracle. TFILRO is Two-way partially-specified preimage-tractable fixed input length random oracle.

2.2 Realization of FPT-RO

By using the technique of [9], we can realize FPT-RO as follows. Let L be a list of RO that is initially empty. Let $\mathcal{M}$ be a finite set and $\mathcal{Z} = \{0,1\}^n$. Let RO be a random oracle that maps from $\mathcal{M}$ to $\mathcal{Z}$. In this paper, we assume that $|\mathcal{M}| \gg |\mathcal{Z}|$, namely the probability that FPO returns $\perp$ is negligible.

- $\mathsf{RO}(M)$: If $\exists (M, z) \in L$, return z. Otherwise, $z \xleftarrow{\$} \mathcal{Z}$, $L \leftarrow (M, z)$, and return z.
- $\mathsf{FPO}(z)$: $M \xleftarrow{\$} \mathcal{M}$, $L \leftarrow (M, z)$, and return M.

Since cryptosystems (honest parties) access RO and any adversary can access RO, RO has public and private interfaces. Since any adversary can access FPO, FPO has the public interface. Note that we can also realize FPT-RO by using the technique of [15]. However, since the technique of [9] is simpler than that of [15], we use the former.

2.3 Realization of TFILRO [9]

TFILRO can be realized by using the technique of Hoch and Shamir [9]. In the following discussions, we assume $m \geq 2n$ such that the probability that BrO returns $\perp$ is less than $O(\frac{1}{2^n})$. This means that a query to BrO triggers an overwhelming number of responses. However, there are far fewer responses to query (y, z) due to the description of FIL-RO. Therefore, we realize TFILRO by considering this fact. Hoch and Shamir solve this problem by using a Poisson distribution. The number of responses of BO is defined by using a Poisson distribution and a response of BO is selected from candidates whose the number is defined by a Poison distribution. Please see Section 3.1 of [9] for more details. TFILRO can be realized as follows. Let L_f and T_f be initially empty lists.

- FIL-RO : On query (x, y),
  1. If $\exists (x, y, z) \in L_f$, return z.
  2. Generate an integer j with Poisson distribution conditioned on being non-zero, $z \xleftarrow{\$} \{0,1\}^n$, $T_f \leftarrow (j, y, z)$, $L_f \leftarrow (x, y, z)$, and Return z.
- BrO : On query (x, z),
  1. $y \xleftarrow{\$} \{0,1\}^m$.
  2. Generate an integer j with Poisson distribution conditioned on being non-zero, $T_f \leftarrow (j, y, z)$, $L_f \leftarrow (x, y, z)$, and Return y.
- BO : On query (y, z),
  1. If $\exists (a, y, z) \in T_f$, $j \leftarrow a$. Else, generate an integer j with Poisson distribution and $T_f \leftarrow (j, y, z)$.
  2. If $j = 0$, $x \leftarrow \perp$. Else choose x uniformly from the j possible answers (some may not be defined yet).
  3. If x is not defined, $x \xleftarrow{\$} \{0,1\}^n$ and $L_f \leftarrow (x, y, z)$.
  4. Return x.

2.4 Zipper Hash Function [10]

The description of Zipper hash function $\mathsf{ZHF}^{g,h}$ is as follows [10]. Let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ and $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be compression functions. On input M, the following procedure is executed.

1. Let $x_1, ..., x_i$ be m-bit strings such that $x_1||...||x_i = M$.
2. H_1 is computed as $g(x_1, IV)$, and $H_2, ..., H_i$ are computed iteratively as $H_j = g(x_j, H_{j-1})$.
3. H'_1 is computed as $h(x_i, H_i)$, and $H'_2, ..., H'_i$ are computed iteratively as $H_j = h(x_{i-j+1}, H'_{j-1})$.
4. Output H'_i.

Liskov claimed that the Zipper hash function is indifferentiable from RO when underlying compression functions g and h are modeled by TFILROM as defined in [10].

2.5 Double Pipe Hash Function [11]

Double pipe hash function DPHF^f was proposed by Lucks [11]. Hoch and Shamir [9] proved that $\mathsf{DPHF}^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$. The description of Double pipe hash function is as follows.

1. For input message M, split M into i blocks each of size $m-n$ bits, $x_1, ..., x_i$.
2. Set $r_0 = IV_1, s_0 = IV_2$ where IV_1 and IV_2 are the initialization vectors.
3. For each message block x_j compute $r_j = f(r_{j-1}, s_{j-1}||x_j)$ and $s_j = f(s_{j-1}, r_{j-1}||x_j)$ where $f : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ is a compression function.
4. Output $f(IV_3, r_i||s_i||0^{m-2n})$ where IV_3 is the initialization vector.

Note that the above description omits the message padding operation.

2.6 The Parallel Hash Function [9]

The indifferentiability of parallel hash function $\mathsf{PHF}^{g,h}$ was discussed by Hoch and Shamir [9]. They proved that $\mathsf{PHF}^{g,h} \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$. The description of $\mathsf{PHF}^{g,h}$ is as follows.

1. For input message M, split message M into i blocks each of size m bits, $x_1, ..., x_i$.
2. Set $r_0 = IV_1, s_0 = IV_2$ where IV_1 and IV_2 are the initialization vectors.
3. For each message block x_j compute $r_j = g(r_{j-1}, x_j)$ and $s_j = h(s_{j-1}, x_j)$ where $g : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ and $h : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ are compression functions.
4. Output $r_i \oplus s_i$.

Note that the above description omits the message padding operation.

2.7 EMD Construction [1]

The EMD^f construction was proposed by Bellare and Ristenpart [1]. They showed that $\mathsf{EMD}^f \underset{\mathsf{FIL\text{-}ROM}}{\sqsubset} \mathsf{RO}$ holds. The EMD^f construction is as follows.

1. For input message M, split M into $i-1$ blocks each of size m bits $x_1, ..., x_{i-1}$ and a $m-n$ bit block, x_i.
2. Set $c_0 = IV_1$ where IV_1 is the initialization vector.
3. For each message block x_j compute $c_j = f(c_{j-1}, x_j)$ where $f : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$ is a compression function.
4. Output $f(IV_2, c_{i-1}||x_i)$.

Note that the above description omits the message padding operation.

3 New Conversion for First Pre-image Tractable Random Oracle Model

In this section, by using indifferentiability with condition, we propose conversion conv that can convert any cryptosystem secure in ROM to one that is secure in $\mathsf{FPT\text{-}ROM}$.

3.1 Conversion

Our proposal can make a cryptosystem secure in $\mathsf{FPT\text{-}ROM}$ simply by modifying the hash function inputs. In following discussion, we assume that $\mathcal{M} = \{0,1\}^*$. Note that the same discussion can be applied to the case of $\mathcal{M} = \{0,1\}^i$ for some i (input length of RO is fixed.).

New Conversion: We define condition α wherein last s bits of the input to $\mathsf{RO}^{\mathsf{priv}}$ is the constant value. Therefore, new conversion conv is as follows. Let $\mathcal{C}$ be any cryptosystem.

– For any value x that is sent from cryptosystem $\mathcal{C}$ to a hash function, in modified cryptosystem $\mathsf{conv}(\mathcal{C})$ $x||c$ is sent to a hash function instead of x where c is the s bit constant value.

The following theorem holds.

Theorem 1. $\mathsf{FPT\text{-}RO} \sqsubset_{\alpha} \mathsf{RO}$ *where* α *is such that query* m *to* $\mathsf{RO}^{\mathsf{priv}}$ *is equal to* $x||c$ *where* x *is any value, for any* t_D*, with* $t_{\mathsf{S}} = O(q)$ *and* $\epsilon = O(\frac{q}{2^s})$*.*

Note that s must be selected such that if we want to select the constant value to ensure that the advantage of D is less than $O(\frac{q}{2^a})$, $s \geq a$ must hold, and if we want to select the constant value to ensure that the probability that BrO returns $\perp$ is less than $O(\frac{1}{2^b})$, $l + s - n \geq b$ must hold where l is the maximum input length of the hash functions except for c (namely $l + s$ is maximum input length).

Proof (Theorem 1). In the following proof, first we introduce a simulator that simulates the public interfaces of FPO. Let $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ be the simulator that simulates the public interface of RO and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ be the simulator that simulates $\mathsf{FPO}^{\mathsf{pub}}$. Second, we show that no D can distinguish $\mathsf{FPT\text{-}RO}$ from RO and the simulator.

S has initially empty list L_S.

– $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$: On query M,
  1. If $\exists(M, z) \in \mathsf{L}_S$, return z.
  2. Else poses query M to RO, receives z, $\mathsf{L}_S \leftarrow (M, z)$, and returns z.

– $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$: On query z,
  1. $M \xleftarrow{\$} \mathcal{M}$, $\mathsf{L}_S \leftarrow (M, z)$ and returns M.

When D poses all queries to the simulator, running time of the above simulator is maximum. Therefore, $t_{\mathsf{S}} = O(q)$.

First we show that no D can distinguish FPO from $(\mathsf{RO}^{\mathsf{priv}}, \mathsf{S})$ as follows. For query M to $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$, the simulator simply returns the output of $\mathsf{RO}(M)$. Therefore, $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ can simulate $\mathsf{RO}^{\mathsf{pub}}$.

Second we show that $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ simulates the public interface of FPO. Procedures of $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ are the same as those of FPO. Therefore, $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ can simulate $\mathsf{RO}^{\mathsf{pub}}$ in the case that consistency between several interfaces.

Finally, we show consistency between $\mathsf{RO}^{\mathsf{priv}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ and consistency between $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$.

Consistency between $\mathsf{RO}^{\mathsf{priv}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$: Above simulator $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ has the same construction as FPO. Therefore, when D poses query z to $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ where z is a response of $\mathsf{RO}^{\mathsf{priv}}$, no D can distinguish $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ from FPO.

Since all responses of $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ are randomly chosen from $\mathcal{M}$, the probability that the last s bits of some response of $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ are equal to c is $O(\frac{1}{2^s})$. Therefore, no D can make a query that is the response of $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ to $\mathsf{RO}^{\mathsf{priv}}$ without incurring the probability of $O(\frac{q}{2^s})$.

From above discussions, since D can make queries at most q times, D can distinguish RO from $\mathsf{FPT\text{-}RO}$ with probability less than $O(\frac{q}{2^s})$. Therefore, the probability that D can distinguish RO from $\mathsf{FPT\text{-}RO}$ by using the relation between $\mathsf{RO}^{\mathsf{priv}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ is at most $O(\frac{q}{2^s})$.

Consistency between $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$: Since $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ and $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$ has same list L_S, $\mathsf{S}_{\mathsf{RO}^{\mathsf{pub}}}$ is explicitly consistent with $\mathsf{S}_{\mathsf{FPO}^{\mathsf{pub}}}$.

From the above discussions, $\epsilon = O(\frac{q}{2^s})$. □

From Theorem 1, the following corollary is obtained.

Corollary 1. *For any cryptosystem* $\mathcal{C}$*,* $\mathsf{conv}(\mathcal{C})(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathsf{conv}(\mathcal{C})(\mathsf{RO}^{\mathsf{priv}})$ *holds.*

Next we show the following theorem.

Theorem 2. $\mathsf{RO}_c \sqsubset \mathsf{RO}$ *where* RO_c *is an entity such that for any query* x *to* RO_c*, it responds with* $\mathsf{RO}(x||c)$*.*

Proof. Since for new query x to RO_c the response is randomly chosen and for repeated queries the response is selected from L, the above theorem explicitly holds. □

Therefore, for any cryptosystem $\mathcal{C}$, $\mathcal{C}(\mathsf{RO}_c^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$ holds from the indifferentiability framework. Since $\mathcal{C}(\mathsf{RO}_c^{\mathsf{priv}})$ is the same as $\mathsf{conv}(\mathcal{C})(\mathsf{RO}^{\mathsf{priv}})$, the following corollary holds.

Corollary 2. *For any cryptosystem* $\mathcal{C}$*,* $\mathsf{conv}(\mathcal{C})(\mathsf{RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$*.*

From corollary 1 and 2, the following corollary is obtained

Corollary 3. *For any cryptosystem* $\mathcal{C}$*,* $\mathsf{conv}(\mathcal{C})(\mathsf{FPT\text{-}RO}^{\mathsf{priv}}) \succ \mathcal{C}(\mathsf{RO}^{\mathsf{priv}})$*.*

Remark 1. There are conditions other than the above conditions. For example, an interface of an input of a hash function satisfies one of the following conditions $c||x$, $x||\mathsf{msb}_s(x)$, $x||\mathsf{lsb}_s(x)$, $\mathsf{msb}_s(x)||x$, $\mathsf{lsb}_s(x)||x$ and so on.

4 Several Constructions from Fixed Input Length Weakened Random Oracle

We show that $\mathsf{ZHF}^{g,h} \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$ holds. We propose two constructions, $\mathsf{EMD}+$ and $\mathsf{EMD}\times$, such that $\mathsf{EMD}+^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ and $\mathsf{EMD}\times^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ hold. Our constructions are based on the EMD construction. These constructions are more efficient than the existing constructions (double pipe hash function and parallel hash function).

4.1 Distinguishing Attack for Zipper Hash Function

We show $\mathsf{ZHF}^{g,h} \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$. The attack procedure of the Zipper hash function in $\mathsf{TFILROM}$ is as follows. Let q be the maximum number of queries made by D. Let $\mathsf{S_{BO}}$ be the simulator of BO of h and BO_h be BO of h. Let H be the Zipper hash function.

1. $a \leftarrow 0$.
2. While the number of queries is less than q, execute the following procedure.
   (a) $b \xleftarrow{\$} \{0,1\}$.
   (b) $m_1 \xleftarrow{\$} \{0,1\}^m$
   (c) If $b = 0$, pose query m_1 to $\mathsf{ZHF}^{g,h}/\mathsf{RO}$, receive z, pose query z to $\mathsf{BO}_h/\mathsf{S_{BO}}$, and receive x. If $x = \perp$, output 1 and stop.
   (d) Else ($b = 1$), $z \xleftarrow{\$} \{0,1\}^n$, pose query z to $\mathsf{BO}_h/\mathsf{S_{BO}}$, and receive x. If $x = \perp$, $a \leftarrow a + 1$.
3. If $a = 0$, output 1. Else, output 0.

When D interacts with $(\mathsf{ZHF}^{g,h}, \mathsf{TFILRO})$, in step 2-c $x \neq \perp$ holds with probability of 1 due to the Zipper hash function. Therefore, the above distinguisher does not output 1 in step 2-d. On random query to BO_h, BO_h returns $\perp$ with probability $\frac{1}{e}$ where e is Napier's constant. So D receives $\perp$ with probability $\frac{1}{e}$ in step 2-c. The probability that $a \neq 0$ holds in step 3 is $1 - (\frac{1}{e})^{q_d} \approx 0$ where q_d is the number of invoking steps 2-d. Therefore, D outputs 0 with probability of almost 1.

When D interacts with $(\mathsf{RO}, \mathsf{S})$, since z is randomly chosen in both step 2-c and step 2-d, no S can know the step in which z is defined. Thus S cannot correctly decide value x. Therefore, D outputs 1 with non-negligible probability when D interacts with $(\mathsf{RO}, \mathsf{S})$. Therefore, $\mathsf{ZHF}^{g,h} \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$ holds.

4.2 New Construction $\mathsf{EMD}+$

From the above attack, $\mathsf{ZHF}^{g,h} \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$ holds. So the hash function with TFILRO that is indifferentiable from RO is only the Double pipe hash function and the Parallel hash function. However, the Double pipe hash function and the Parallel hash function are not efficient. In this section, we propose new construction $\mathsf{EMD}+$ based on EMD. In order to propose $\mathsf{EMD}+$ such that $\mathsf{EMD}+ \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$ holds, we must overcome three obstacles: the distinguishing attack using $\mathsf{FIL\text{-}WRO}$, the distinguishing attack using BO (the attack in subsection 4.1), and the distinguishing attack using BrO. EMD resists the distinguishing attack using $\mathsf{FIL\text{-}WRO}$ due to [1]. Since the EMD construction has a HMAC-like structure and the Double pipe hash function resists the distinguishing attack using BO (the attack in subsection 4.1) due to its HMAC-like structure , EMD resists the distinguishing attack using BO. Therefore, we modify EMD such that modified construction $\mathsf{EMD}+$ resists the distinguishing attack using BrO. Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be a compression function.

The distinguishing attack using BrO: First we confirm the distinguishing attack using BrO for EMD. The following distinguisher can distinguish EMD^f from RO when underlying compression function f is modeled by $\mathsf{FIL\text{-}WRO}$. Let $\mathsf{S}_{\mathsf{BrO}}$ be a simulator of BrO.

1. $c \overset{\$}{\leftarrow} \{0,1\}^n$.
2. Make query $(IV_2, 0^{m-n}||c)$ to $\mathsf{BrO}/\mathsf{S}_{\mathsf{BrO}}$ and receive z.
3. Make query (IV_1, c) to $\mathsf{BrO}/\mathsf{S}_{\mathsf{BrO}}$ and receive y.
4. Make query y to $\mathsf{EMD}^f/\mathsf{RO}$ and receive z'.
5. If $z = z'$, output 0. Else output 1.

In the above procedure, D explicitly outputs 0 when D interacts with $(\mathsf{EMD}^f, \mathsf{TFILRO})$. When D interacts with $(\mathsf{RO}, \mathsf{S})$, D outputs 1 with non-negligible probability because no S can predict z' in the above step 2. Therefore, the probability that $z \neq z'$ is non-negligible and $\mathsf{EMD}^f \underset{\mathsf{TFILROM}}{\not\sqsubset} \mathsf{RO}$.

$\mathsf{EMD}+$: We apply the conversion of section 3.1 to all compression functions in order to avoid the above attack. We define $\mathsf{EMD}+^f$ as follows.

1. For input message M, compute $M' = pad(M)$ where pad is a padding function that returns a string whose length is an integer multiple of $m - n$.
2. For message M', split M' into i blocks each of size $m - n$ bits, $y_1, ..., y_i$.
3. Set $c_0 = IV_1$ where IV_1 is the initialization vector.
4. For each message block y_j such that $j \in \{1, ..., i\}$ compute $c_j = f(c_{j-1}, y_j||const_1)$ where $const_1$ is the n bit constant value.
5. Output $f(IV_2, 0^{m-n}||c_i)$ where IV_2 $(\neq IV_1)$ is the n bit constant value.

Usually a padding function that returns a string appended with the 64-bit encoding of the input length is used. Since in the above attack y is randomly chosen and the probability that $\mathsf{lsb}_n(y) = const_1$ holds is negligible, the above attack cannot be applied to $\mathsf{EMD}+$. More detail is as follows.

Theorem 3. $\mathsf{EMD}+^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$*, for any* t_D *with* $t_{\mathsf{S}} = O(lq)$ *and* $\epsilon = O(\frac{l^2q^2}{2^n})$ *where* l *is maximum block length of the query made by* D*.*

For more detail, please see proof of Theorem 3 in the Appendix A.

4.3 Efficient Construction $\mathsf{EMD}\times$

$\mathsf{EMD}+$ is more efficient than the double-pipe hash function. However, $\mathsf{EMD}+$ needs to input a n-bit constant value to each compression function. This additional input degrades the efficiency of the hash function compared to EMD. Our solution is $\mathsf{EMD}\times$ which is a more efficient construction than $\mathsf{EMD}+$.

We apply the check sum operation and the block index input operation to the EMD construction. The description of $\mathsf{EMD}\times$ is as follows.

1. For input message M, compute $M' = pad(M)$ where pad is a padding function that returns a string whose length is an integer multiple of $m - s$.
2. For message M', split M' into i blocks each of size $m - s$ bits, $y_1, ..., y_i$.
3. Set $c_0 = IV_1$ where IV_1 is the initialization vector.
4. For each message block y_j compute $c_j = f(c_{j-1}, y_j||\langle j\rangle)$ where $\langle j\rangle$ is the s-bit binary representation of the index of the compression function.
5. Compute $c = f(c_i, sum||\langle 0\rangle)$ where $sum = y_1 \oplus \cdots \oplus y_i$.
6. Output $f(IV_2, 0^{m-n}||c)$.

Usually we use a padding function that returns a string appended with the 64-bit encoding of the length. Note that $s \geq \log_2 l$ should hold where l is the maximum number of compression function calls in $\mathsf{EMD}\times$, and $s \geq \log_2 q$ should hold when the maximum total number of query of D is q. The security of indifferentiability is as follows.

Theorem 4. $\mathsf{EMD}\times^f \underset{\mathsf{TFILROM}}{\sqsubset} \mathsf{RO}$, *for any* t_D *with* $t_{\mathsf{S}} = O(lq)$ *and* $\epsilon = O(\frac{l^2q^2}{2^n})$ *where* l *is the maximum block length of a query made by* D.

$\mathsf{EMD}\times$ can resist the third attack thanks to the check sum operation and the block index input operation. Roughly speaking, the reason for this is as follows. Since a block index is input in each compression function and the output of BrO is randomly chosen, the probability that the right most s bits of an output of BrO equal the target value is negligible. Moreover, since the output of BrO is randomly chosen, the probability that some message satisfying $sum = y_1 \oplus \cdots \oplus y_i$ is found by using BrO is negligible. Therefore, the probability that chain messages $y_1||\langle 1\rangle, ..., y_i||\langle i\rangle, sum||\langle 0\rangle$ are obtained by using BrO is negligible. Therefore, $\mathsf{EMD}\times$ can resist the third attack due to the check sum operation and the block index input operation. This proof is similar to the proof of theorem 3.

4.4 Comparison

We compare our schemes to the Double pipe hash function. In the following discussions, we consider the case that $n = 256$ and $m = 512$.
[EMD+ v.s. DPHF]: Length of one block message of DPHF and EMD+ is 256 bits. However, a compression function is called two times per one block message in DPHF. On the other hand, a compression function is called one time in EMD+. Therefore, EMD+ is more efficient than the Double pipe hash function.
[EMD+ v.s. PHF]: Two types of compression functions are called per one block message in PHF. EMD+ uses only one type of compression function. Therefore, EMD+ is more efficient than PHF.
[EMD× v.s. DPHF]: Usually, since the maximum input length of a hash function is less than 2^{64}, the maximum number of compression function calls in EMD× is less than 2^{64}. Therefore, length s of a block index is less than 64 bits. When $s = 64$, length of one block message of EMD× is 448 bits. A compression function is called two times per one block message in DPHF while a compression function is called one time in EMD×. Therefore, EMD× is more efficient than the Double pipe hash function.
[EMD× v.s. PHF]: Two types of compression functions are called per one block message in PHF while EMD× uses only one type of compression function. Therefore, EMD× is more efficient than PHF.
[EMD× v.s. EMD+]: One block length of EMD× is more than 448 bits. On the other hand, that of EMD+ is 256 bits. However, in EMD×, *sum* is input in the second last compression function. Therefore, when the input length is less than 512 bits, EMD+ is more efficient than EMD×. On the other hand, when the input length is more than 512 bits, EMD× is more efficient than EMD+.

4.5 Remarks

We propose hash constructions EMD+ and EMD× based on the EMD construction. When we apply the technique of either EMD+ or EMD× to other constructions such as PMD and MDP, the resulting constructions are indifferentiable from RO when the underlying compression function is modeled by FIL-RO and BrO. However, these constructions are not indifferentiable from RO when the underlying compression function is modeled by TFILRO.

References

1. Mihir Bellare and Thomas Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In *ASIACRYPT*, pages 299–314, 2006.
2. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.
4. Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.
5. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Functions Constructions from PGV. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2002.
6. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.
7. Ivan Damgård. A Design Principle for Hash Functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
8. Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007.
9. Jonathan J. Hoch and Adi Shamir. On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak. In *ICALP*, Lecture Notes in Computer Science, pages 616–630. Springer, 2008.
10. Moses Liskov. Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In *Selected Areas in Cryptography*, Lecture Notes in Computer Science, pages 358–375. Springer, 2006.
11. Stefan Lucks. A Failure-Friendly Design Principle for Hash Functions. In *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer, 2005.
12. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.
13. Ralph C. Merkle. One Way Hash Functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
14. Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta. How to Confirm Cryptosystems Security: the Original Merkle-Damgård is Still Alive! In *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*. Springer, 2009.
15. Akira Numayama, Toshiyuki Isshiki, and Keisuke Tanaka. Security of Digital Signature Schemes in Weakened Random Oracle Models. In *Public Key Cryptography*, Lecture Notes in Computer Science, pages 268–287. Springer, 2008.
16. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

A Proof of Theorem 3

In the following proof, we ignore the message padding operation so the length of any input to a hash function is a integer multiple of $m-n$, since $\mathsf{EMD+}$ with the message padding operation is just a special case of $\mathsf{EMD+}$ without the message padding operation. First ,we define simulator S as follows. S has initially empty lists L_S and T. Let f be $\mathsf{FIL\text{-}RO}$. Let S_f be a simulator that simulates f, S_{BO} a simulator that simulates BO, and $\mathsf{S}_{\mathsf{BrO}}$ a simulator that simulates BrO. We define terms "chain triples" and "semi-chain triples" as follows.

Definition 4 (Chain Triples of $\mathsf{EMD+}$). *$(x_1,y_1,z_1),...,(x_i,y_i,z_i)$ are chain triples of $\mathsf{EMD+}$ if these pairs satisfy the following: $x_1 = IV_1$, $x_t = z_{t-1}$ $(t = 2,...,i-1)$, $\mathsf{lsb}_n(y_t) = const_1$ $(t = 1,...,i-1)$, $\mathsf{lsb}_n(y_i) = z_{i-1}$, $\mathsf{msb}_{m-n}(y_i) = 0^{m-n}$ and $x_i = IV_2$.*

Chain triples of $\mathsf{EMD+}$ are input-output triples of the compression function in $\mathsf{EMD+}$.

Definition 5 (Semi-chain Triples of $\mathsf{EMD+}$). *$(x_1,y_1,z_1),...,(x_i,y_i,z_i)$ are semi-chain triples of $\mathsf{EMD+}$ if these pairs satisfy the following: $x_1 = IV_1$, $x_t = z_{t-1}$ $(t = 2,...,i)$, $\mathsf{lsb}_n(y_t) = const_1$ $(t = 1,...,i)$.*

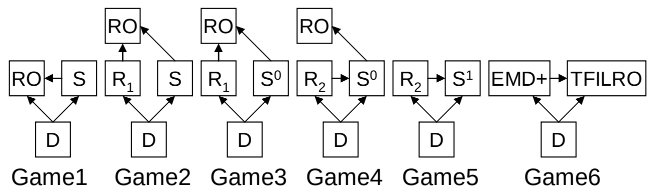

Fig. 3. Games

Semi-chain triples of EMD+ are input-output triples of a compression function in EMD+ except for the last compression function.

Simulator S.

- S_f: On query (x, y)
  1. If $\exists(x, y, z) \in \mathsf{L}_S$, return z.
  2. Generate integers a with Poisson distribution conditioned on being non-zero.
  3. $z \xleftarrow{\$} \{0,1\}^n$.
  4. $T \leftarrow (a, y, z)$.
  5. $\mathsf{L}_S \leftarrow (x, y, z)$.
  6. If $\exists(x_1, y_1, z_1), ..., (x_i, y_i, z_i) \in \mathsf{L}_S$ such that $(x_1, y_1, z_1), ..., (x_i, y_i, z_i), (x, y, z)$ are semi-chain triples,
     (a) $z' \leftarrow \mathsf{RO}(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_i))||\mathsf{msb}_{m-n}(y))$.
     (b) Generate integers b with Poisson distribution conditioned on being non-zero.
     (c) $T \leftarrow (b, 0^{m-n}||z, z')$,
     (d) $\mathsf{L}_S \leftarrow (IV_2, 0^{m^n}||z, z')$.
  7. Return z.
- S_{BO}: On query (y, z)
  1. If $\exists(j, y, z) \in T$, $a \leftarrow j$. Else generate an integer a with Poisson distribution and $T \leftarrow (a, y, z)$.
  2. Choose x uniformly from the a possible answers (some may not be defined yet).
  3. If the chosen answer x is not defined, $x \xleftarrow{\$} \{0,1\}^n$ and $\mathsf{L}_S \leftarrow (x, y, z)$.
  4. If $a = 0$, $x \leftarrow \perp$.
  5. Return x.
- $\mathsf{S}_{\mathsf{BrO}}$: On query (x, z)
  1. $y \xleftarrow{\$} \{0,1\}^m$.
  2. Generate an integer a with Poisson distribution conditioned on being non-zero.
  3. $T \leftarrow (a, y, z)$.
  4. Return y.

In above simulator, though we omit the description of how to store pairs in L_S, we can construct an efficient simulator by using the technique of [9] for storing pairs in L_S.

In the worst case of the simulator's running time, S_f executes step 4 for every query and this requires at most $O(ql)$ time where q is the maximum number of queries of D and l is maximum number of message blocks in the hash queries.

We call a query to f/S_f "forward query", a query to $\mathsf{BO}/\mathsf{S}_{\mathsf{BO}}$ "backward query", and a query to $\mathsf{BrO}/\mathsf{S}_{\mathsf{BrO}}$ "bridge query".

The proof involves a hybrid argument starting in the RO scenario, and ending in the TFILRO scenario through a sequence of mutually indistinguishable hybrid games. Fig. 3 is the figure of each game.

Game 1: This is the random oracle model, where D has oracle access to RO and S. Let G1 denote the event that D outputs 1 after interacting with RO and S. Thus $Pr[\mathsf{G1}] = Pr[D^{\mathsf{RO},\mathsf{S}(\mathsf{RO})} \Rightarrow 1]$.

Game 2: In this game, we give the distinguisher oracle access to a dummy relay algorithm R_1 instead of direct oracle access to RO. R_1 is given oracle access to RO. On query M to R_1, it queries M to RO and returns $\mathsf{RO}(M)$. Let G2 denote the event that D outputs 1 in Game 2. Since the view of D remains unchanged in this game, $Pr[\mathsf{G2}] = Pr[\mathsf{G1}]$.

Game 3: In this game, we modify the simulator S. In particular, we restrict the responses of the simulator. If a response of the simulator satisfies one of following conditions, it fails explicitly instead of sending this response.

Let S^0 be the new simulator that avoids following conditions. In response to forward query (x, y), new simulator S^0_f chooses response $z \in \{0,1\}^n$ similar to S and it checks for the following conditions:

- $\mathsf{FC1}$: the case that $z = IV_1$.
- $\mathsf{FC2}$: triple (x', y', z') exists in L_S, with $(x', y') \neq (x, y)$, such that $z' = z$.
- $\mathsf{FC3}$: triple (x', y', z') exists in L_S, with $(x', y') \neq (x, y)$, such that $x' = z$.
- $\mathsf{FC4}$: triple (x', y', z') exists in L_S, such that $\mathsf{lsb}_n(y') = z$.

If response z is chosen by S^0 then S^0 checks for these conditions and explicitly fails if any of them holds.

In step 6 of S^0_f, if value z' is defined in step 6 then S^0 checks for conditions $\mathsf{FC1}-4$ and explicitly fails if any of them holds.

If backward query (y, z) is made to simulator S^0, then it chooses response $x \in \{0,1\}^n$ to this query similar to the original simulator S and checks for the following failure conditions:

- $\mathsf{BaC1}$: the case that $x = IV_1$.
- $\mathsf{BaC2}$: the case that $x = IV_2$.
- $\mathsf{BaC3}$: triple (x', y', z') exists in L_S, with $(y', z') \neq (y, z)$, such that $z' = x$.

In the case of backward queries, if response x is chosen by simulator S^0 then S^0 checks for these conditions and explicitly fails if any of them holds.

If bridge query (x, z) is made to simulator $\mathsf{S}^0_{\mathsf{BO}}$, it chooses a response $y \in \{0,1\}^m$ to this query similar to the original simulator S_{BO} and checks for the following failure conditions:

- $\mathsf{BrC1}$: the case that $\mathsf{lsb}_n(y) = const_1$.
- $\mathsf{BrC2}$: triple (x', y', z') exists in L_S such that $z' = \mathsf{lsb}_n(y)$.
- $\mathsf{BrC3}$: triple (x', y', z') exists in L_S, with $(x', z') \neq (x, z)$, such that $y' = y$.

In the case of bridge queries, if response y is chosen by simulator $\mathsf{S}^0_{\mathsf{BrO}}$ then S^0 checks for these conditions and explicitly fails if any of them holds.

If S^0 does not fail, Game 3 is identical to Game 2. We examine the probability that S^0 fails as follows.

Lemma 1. *The probability that simulator S^0 fails is at most $O(\frac{q_1^2}{2^n})$ where q_1 is maximum number of times the simulator is invoked.*

Proof. We will examine each of the twelve conditions and bound their probability.

- $\mathsf{FC1}$: These conditions are that a random value is equal to some fixed value. Therefore, the probability that S^0 fails due to $\mathsf{FC1}$ is at most $O(\frac{q_1}{2^n})$.
- $\mathsf{FC2}$: This condition is that a collision occurs between two random outputs of S^0_f. Therefore, the probability that $\mathsf{FC2}$ occurs is at most $O(\frac{q_1^2}{2^n})$.
- $\mathsf{FC3}$: Since z is randomly chosen, the probability that z collides with some value x' in L_S is at most $1-(1-\frac{1}{2^n})(1-\frac{2}{2^n})...(1-\frac{q_1-1}{2^n}) = \frac{q_1^2}{2^n} = O(\frac{q_1^2}{2^n})$. Therefore, the probability that S_0 fails due to $\mathsf{FC3}$ is at most $O(\frac{q_1^2}{2^n})$.
- $\mathsf{FC4}$: Since the output of S^0_f is randomly chosen, the probability that S^0 fails due to $\mathsf{FC4}$is at most $O(\frac{q_1^2}{2^n})$.
- $\mathsf{BaC1}$ and $\mathsf{BaC2}$: Since output of $\mathsf{S}^0_{\mathsf{BO}}$ for a backward query is randomly chosen, the probability that $\mathsf{S}^0_{\mathsf{BO}}$ fails due to $\mathsf{BaC1}$ or $\mathsf{BaC2}$ is at most $O(\frac{q_1}{2^n})$.
- $\mathsf{BaC3}$: Since response x of a backward query is randomly chosen, the probability that x collides some value z' in L_S is at most $O(\frac{q_1^2}{2^n})$. Therefore, the probability that $\mathsf{S}^0_{\mathsf{BO}}$ fails due to $\mathsf{BaC3}$ is at most $O(\frac{q_1^2}{2^n})$.
- $\mathsf{BrC1}$: Since the output of $\mathsf{S}^0_{\mathsf{BO}}$ for a bridge query is randomly chosen, the probability that S^0 fails due to $\mathsf{BrC1}$ is at most $O(\frac{q_1^2}{2^n})$.

– $\mathsf{BrC2}$: Since the output of $\mathsf{S}^0_{\mathsf{BrO}}$ for a bridge query is randomly chosen, the probability that S^0 fails due to $\mathsf{BrC2}$ is at most $O(\frac{q_1^2}{2^n})$.
– $\mathsf{BrC3}$: since the output of $\mathsf{S}^0_{\mathsf{BrO}}$ for a bridge query is randomly chosen, the probability that two random values collide is at most $O(\frac{q_1^2}{2^m})$. Therefore, the probability that S^0 fails due to $\mathsf{BrC3}$ is at most $O(\frac{q_1^2}{2^m})$.

Therefore, the probability that S^0 fails is bounded by $O(\frac{q_1^2}{2^n})$. □

Let q_h be the total number of queries to S^0 made by $\mathcal{D}$. Let $\mathsf{G3}$ denote the event that distinguisher D outputs 1 in Game 3. From Lemma 1, $|Pr[\mathsf{G3}] - Pr[\mathsf{G2}]| = O(\frac{q_h^2}{2^n})$.

Game 4: In this game, we modify the relay algorithm as follows. The underlying idea is to make the responses of the relay algorithm directly dependent on the simulator. Thus, R_2 is essentially the same as $\mathsf{EMD+}^{\mathsf{S}^0_\mathsf{f}}$.

We will show that Game 4 is identical to Game 3 unless S^0 fails. The same simulator S_0 is used in Game 3 and Game 4. However, the relay algorithm in Game 4 is different from that in Game 3. Accordingly, we demonstrate the following three facts:

1. Unless S^0 fails when interacting with (R_1, S^0), the answers given by S^0 are consistent with those given by R_1.
2. Unless S^0 fails when interacting with (R_2, S^0), the answers given by S^0 are consistent with those given by R_2.
3. Unless S^0 fails when interacting with either with (R_1, S^0) or with (R_2, S^0), the answers given by R_1 are exactly the same as those given by R_2.

Before demonstrating these facts, we present two useful properties as follows.

Lemma 2. *Any chain triples of* $\mathsf{EMD+}$ $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ *in* L_S *are stored by the ordered sequences of forward queries* $(x_1, y_1), ..., (x_{i-1}, y_{i-1})$ *unless* S^0 *fails.*

Proof. To check the contrary, assume that there is some chain triples of $\mathsf{EMD+}$ $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ in L_S such that

1. (x_t, y_t, z_t) $(t \in \{1, ..., i-1\})$ is stored in L_S by a forward query when there is $(x_{t+1}, y_{t+1}, z_{t+1})$ in L_S,
2. (x_t, y_t, z_t) $(t \in \{1, ..., i\})$ is stored by a backward query or a bridge query, or
3. (x_i, y_i, z_i) is stored in L_S by forward query (x_i, y_i).

Given the first assumption, when $t \neq i-1$ is stored by a forward query, since $z_t = x_{t+1}$, S^0 fails due to $\mathsf{FC3}$. When (x_t, y_t, z_t) such that $t = i-1$ is stored by a forward query, since $z_{i-1} = \mathsf{lsb}_n(y_i)$, S^0 fails due to $\mathsf{FC4}$.

Given the second assumption, let j be a minimum value such that $j \in \{1, ..., i\}$ and (x_j, y_j, z_j) is stored by a bridge query or a backward query. When $j = 1$,

– in the case that (x_j, y_j, z_j) is stored by a backward query, since $x_j = IV_1$, S^0 fails due to $\mathsf{BaC1}$, or
– in the case that (x_j, y_j, z_j) is stored by a bridge query, since $\mathsf{msb}_{m-n}(y_j) = const_1$, S^0 fails due to $\mathsf{BrC1}$.

When $j \neq 1$ and $j \neq i$,

– in the case that (x_j, y_j, z_j) is stored by a backward query when $(x_{j-1}, y_{j-1}, z_{j-1})$ is in L_S, since $z_{j-1} = x_j$ holds, S^0 fails due to $\mathsf{BaC3}$,
– in the case that (x_j, y_j, z_j) is stored by a backward query when $(x_{j-1}, y_{j-1}, z_{j-1})$ is not in L_S, since $(x_{j-1}, y_{j-1}, z_{j-1})$ is stored by a forward query, S^0 fails due to $\mathsf{FC3}$, or
– in the case of (x_t, y_t, z_t) stored by a bridge query, since $\mathsf{msb}_{m-n}(y_j) = const_1$, S^0 fails due to $\mathsf{BrC1}$.

When $j = i$,

– in the case that (x_i, y_i, z_i) is stored by a backward query, since $x_i = IV_2$, S^0 fails due to $\mathsf{BaC2}$,

- in the case that (x_i, y_i, z_i) is stored by a bridge query when $(x_{i-1}, y_{i-1}, z_{i-1})$ is in L_S, S^0 fails due to $\mathsf{BrC2}$, or
- in the case that (x_i, y_i, z_i) is stored by a bridge query when $(x_{i-1}, y_{i-1}, z_{i-1})$ is not in L_S, S^0 fails due to $\mathsf{FC4}$.

Given the third assumption, since (x_i, y_i, z_i) is stored in L_S by forward query (x_i, y_i)), the following case should occur from the definition of S^0_f (step 6): (x_i, y_i, z_i) is stored in L_S before storing (x_t, y_t, z_t) for some t such that $t \le i-1$. When $t \ne i-1$, this case is the same as that of the first or second assumption. Therefore, in this case S^0 fails. When $t = i-1$ and (x_t, y_t, z_t) is stored by a forward query, namely $(x_{i-1}, y_{i-1}, z_{i-1})$ is stored after storing (x_i, y_i, z_i), S_0 fails due to $\mathsf{FC4}$. When $t = i-1$ and (x_t, y_t, z_t) is stored by a bridge query or a backward query, this case is the same as the case of assumption 2. Therefore, S^0 fails.

Therefore, this lemma is valid. □

Lemma 3. *For any chain triples of* $\mathsf{EMD+}$ $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ *in* L_S,
$z_i = \mathsf{RO}(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ *holds unless* S^0 *fails.*

Proof. From Lemma 2, $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ were stored by the ordered sequences of forward queries $(x_1, y_1), ..., (x_{i-1}, y_{i-1})$ unless S^0 fails. Therefore, z_i is defined by step 6 of S_0 from the definition of S^0_f. $z_i = \mathsf{RO}(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ holds unless one of following cases occurs.

- t exists in $\{1, ..., i-2\}$ such that $z_t = IV_1$, or
- (x_t, y_t, z_t) (t exists in $\{1, ..., i-1\}$) and $(x'_1, y'_1, z'_1), ..., (x'_j, y'_j, z'_j) \in \mathsf{L}_S$ such that $x'_1 = IV$, $z'_k = x'_{k+1}$ ($k = 1, ..., i-1$), and $z'_j = z_t$.

The first case is identical to $\mathsf{FC1}$. The second case is identical to condition $\mathsf{FC2}$.

Therefore, $z_i = \mathsf{RO}(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ unless S^0 fails. □

We prove the first point by Lemma 4: we demonstrate that, when interacting with (R_1, S^0), any chain triples of $\mathsf{EMD+}$ obtained by S_0 are consistent with an output of R_1 unless S^0 fails.

Lemma 4. *For any chain triples of* $\mathsf{EMD+}$ $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ *in* L_S,
$z_i = R_1(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ *holds unless* S^0 *fails.*

Proof. Since for any M $R_1(M) = \mathsf{RO}(M)$ holds, this lemma is explicitly valid due to Lemma 3. □

Second, we prove the first point by Lemma 5: we demonstrate that, when interacting with (R_1, S^0), any chain triples of $\mathsf{EMD+}$ obtained by S_0 are consistent with an output of R_1 unless S^0 fails.

Lemma 5. *For any chain triples of* $\mathsf{EMD+}$ $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ *in* L_S,
$z_i = R_2(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ *holds unless* S^0 *fails.*

Proof. Since R_2 is the same as $\mathsf{EMD+}^{\mathsf{S}^0_f}$ and $(x_1, y_1, z_1), ..., (x_i, y_i, z_i)$ are explicitly inner input-output triples of $\mathsf{EMD+}^{\mathsf{S}^0_f}(\mathsf{msb_{m-n}(y_1)||...||msb_{m-n}(y_{i-1})})$, $z_i = R_2(\mathsf{msb}_{m-n}(y_1)||...||\mathsf{msb}_{m-n}(y_{i-1}))$ explicitly holds unless S^0 fails. □

Third, we prove the first point by Lemma 6: we demonstrate that the answers given by R_1 are exactly the same as those given by R_2 unless S^0 fails.

Lemma 6. *For any input* M, $R_2(M) = \mathsf{RO}(M)$ *holds unless* S^0 *fails.*

Proof. This lemma is explicitly valid due to Lemma 3. □

Since $R_1(M) = \mathsf{RO}(M)$ holds for any M, answers given by R_1 are exactly the same as those given by R_2.

From Lemma 4, 5, and 6, no $\mathcal{D}$ can distinguish Game 3 from Game 4 unless S^0 fails. Let q_F be total number of queries to the relay algorithm made by $\mathcal{D}$. From 1, the probability that S^0 fails is bounded by $O(\frac{(lq_F+q_h)^2}{2^n}) = O(\frac{(lq)^2}{2^n})$. Let $\mathsf{G4}$ denote the event that distinguisher $\mathcal{D}$ outputs 1 in Game 4. Therefore, $|Pr[\mathsf{G4}] - Pr[\mathsf{G3}]| = O(\frac{(ql)^2}{2^n})$.

Game 5: In this game, we modify simulator S^0_f. In new simulator S^1 RO is removed as follows.

```
– $\mathsf{S}^1_f(x, y)$:
    1. If $\exists(x, y, z) \in \mathsf{L}_S$, return $z$.
    2. Generate integers $a$ with Poisson distribution conditioned on being non-zero.
    3. $z \xleftarrow{\$} \{0, 1\}^n$.
    4. $T \leftarrow (a, y, z)$,
    5. $\mathsf{L}_S \leftarrow (x, y, z)$.
    6. return $z$.
```

Step 6 of S^0_f is removed in S^1_f. Note that $\mathsf{S}^1_{\mathsf{BO}}$ and $\mathsf{S}^1_{\mathsf{BrO}}$ are same as $\mathsf{S}^0_{\mathsf{BO}}$ and $\mathsf{S}^0_{\mathsf{BrO}}$. Since no D can see the values defined in step 6 of S^0_f, RO outputs a random value and RO is called by just S^0_f, the view of D for S^1_f is the same as that for S^0_f. Therefore, Game 5 is identical to Game 4 unless S^0 fails. Let $\mathsf{G5}$ denote the event that distinguisher D outputs 1 in Game 5. Since the probability that S_0 fails is at most $O(\frac{(lq)^2}{2^n})$, $|Pr[\mathsf{G5}] - Pr[\mathsf{G4}]| = O(\frac{(lq)^2}{2^n})$.

Game 6. This is the final game of our argument. Here we finally replace S^1 with TFILRO. Since S^1 is the same as TFILRO, this game is identical to Game 5. Let $\mathsf{G6}$ denote the event that distinguisher D outputs 1 in Game 6. Obviously we can deduce that $Pr[\mathsf{G6}] = Pr[\mathsf{G5}]$.

Now we can complete the proof of the theorem by combining Games 1 to 6, and observing that game 1 is the same as the RO model while Game 6 is the same as the TFILRO model. Hence we can deduce that the advantage of D is at most $O(\frac{(lq)^2}{2^n})$. □