Leaky Random Oracle
(Extended Abstract)

Kazuki Yoneyama[1,⋆], Satoshi Miyagawa[2,⋆⋆], and Kazuo Ohta[1]

[1] The University of Electro-Communications.
[2] NTT DoCoMo, Inc.
yoneyama@ice.uec.ac.jp

Abstract. This work focuses on vulnerability of hash functions due to sloppy usage or implementation in the real world. If our cryptographic research community succeeded in development of perfectly secure random function as random oracle, it might be broken in some sense by invalid uses. In this paper, we propose a new variant of the random oracle model in order to analyze security of cryptographic protocols under the situation of an invalid use of hash functions. Our model allows adversaries to obtain contents of the hash list of input and output pairs arbitrarily. Also, we analyze security of several prevailing protocols (FDH, OAEP, Cramer-Shoup cryptosystem, Kurosawa-Desmedt cryptosystem, NAXOS) in our model. As the result of analyses, we clarify that FDH and Cramer-Shoup cryptosystem are still secure but others are insecure in our model. This result shows the separation between our model and the standard model.

Keywords: random oracle model, standard model, hash list, provable security, leakage.

1 Introduction

Hash functions are one of most important building blocks of cryptographic protocols. Indeed, hash functions are widely used various protocols, e.g., digital signature, public-key cryptosystem, authenticated key exchange, etc.

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

In the theoretical sense, hash functions are frequently modeled as *random oracles* [1]. Random oracle is an idealized random function which is usable for

⋆ Supported by JSPS Research Fellowships for Young Scientists.
⋆⋆ This work was partially done while the author was a student at the University of Electro-Communications, Japan.

J. Baek et al. (Eds.): ProvSec 2008, LNCS 5324, pp. 226–240, 2008.

parties and adversaries in the protocol. We use random oracle model (ROM) (i.e., executed with random oracles) as a technique in order to prove security of various protocols. Mostly, proofs with ROM are easier than the model without random oracles, i.e., the standard model (SM), and can provide tight security reductions. Thus, ROM is a useful tool for the provable security.

On the other hand, Canetti et al. [2,3] showed that there are digital signature schemes and public-key cryptosystems which are secure in ROM but insecure if random oracles are instantiated by real hash functions. Thus, recently, proofs with ROM may seem to be unfavorable for practical uses. However, since to prove security of protocols in SM is generally hard, ROM still has an important role to design new protocols as the guideline for the provable security.

1.1 Motivation

Canetti et al.'s impossibility result would be avoidable if a perfectly secure hash function which has all capabilities of random oracles was developed. If so, does the protocol which is proved security in ROM keep its security if random oracles are instantiated by such perfectly secure hash functions? However, in the practical scenario, an unexpected event may occur on hash functions but does not occur on random oracles. In particular, we focus on vulnerability of hash functions due to sloppy usage or implementation in the real world.

Canetti and Krawczyk [4] formulated a security notion of authenticated key exchange. Their definition captured security under the situation where an ephemeral secret information (local randomness) is leaked. These leakages may occur in the case of that a storage or memory which save a local randomness is attacked by various types of attack or the case of that a randomness generator is corrupted. Note that, such a type of vulnerability is not caused by errors of protocol itself but caused by sloppy usages or implementations.

In this work, we apply this view to hash functions. That is, we consider the situation that pairs of inputs and outputs (contents of the hash list) of hash functions can be leaked to adversaries. These leakages may also occur by sloppy usages or implementations. For example, the hash list may remain in the memory for reuse of hash values in order to reduce computational costs or for failing to release temporary memory area, then contents of the memory may be revealed by various attacks, e.g., malicious Trojan Horse programs, Cold Boot Attacks [5]. Thus, even if we successfully developed exceedingly secure hash functions, such a leakage might be possible.

In this paper, we formulate a new model capturing the above situation in order to discuss security (or insecurity) of protocols which use hash functions as building blocks when such a leakage of the hash list occurs. In order to concentrate effects of the leakage, we suppose that hash functions are ideal as random oracles but contents of the hash list can be leaked to adversaries. By using this model, we give a new criterion of security in ROM and analyze several prevailing protocols.

1.2 Our Contribution

Our main contributions are formulating a new variant of ROM, named *leaky random oracle model* (LROM), to capture the leakage of the hash list and analyzing security and insecurity of prevailing protocols in our model.

Leaky Random Oracle Model. Our model (LROM) allows adversaries to obtain contents of the hash list of input and output pairs in arbitrary timings. Thus, virtually, adversaries always can observe the addition of each hash value and can know the timing of the addition. We model the ordinary query in order to obtain a hash value to (leaky) random oracle as *hash query* and the special query in order to obtain contents of hash list to leaky random oracle as *leak query*. Therefore, LROM is trivially stronger than ROM, i.e., a secure protocol in LROM is also secure in ROM.

Security Analyses of Protocols. By using LROM, we can confirm whether each cryptographic protocol is secure or not if the leakage of the hash list occurs. In this paper, we choose five prevailing protocols for analyzing security in LROM and obtain the result of analyses as follows;

- **FDH:** is secure in both ROM and LROM,
- **OAEP:** is secure in ROM but insecure in LROM,
- **Cramer-Shoup cryptosystem:** is secure in both SM and LROM,
- **Kurosawa-Desmedt cryptosystem:** is secure in SM but insecure in LROM, and
- **NAXOS:** is secure in ROM but insecure in LROM.

Separation from the standard model. Our result of analyses shows the separation between our model and the standard model because of two following observations;

- FDH is secure in LROM under the assumption of trapdoor permutation. However, Dodis et al. [6] showed that FDH is not provable in SM under the same assumption.
- Kurosawa-Desmedt cryptosystem is secure in SM under the DDH assumption, the assumption of universal hash function family and the assumption of symmetric key encryption. However, Kurosawa-Desmedt cryptosystem is insecure in LROM by instantiating hash functions by leaky random oracles under same assumptions.

Difference from randomness revealing. Also, our result shows the difference between our model and ROM under randomness revealing because of the following observation;

- NAXOS is secure in ROM under the leakage of local randomness. However, NAXOS is insecure in LROM even if there is no leakage of local randomness.

1.3 Related Works

Some studies consider modeling of weak random oracles and analyze protocols in their model. Nielsen [7] introduced the non-programmable random oracle model by restricting the simulation of random oracle as the simulator can only set answers of random oracles according to some restriction. Liskov [8] showed the way to construct weak hash functions by adding an additional oracle which can break some property of random oracles, e.g., one-wayness and collision-resistance. Pasini and Vaudenay [9] applied Liskov's model into analyses of the hash-and-sign paradigm. Unruh [10] formulated a variant of ROM by giving oracle-dependent auxiliary inputs to adversaries. In this setting, adversaries can get an auxiliary input that can contain information about the random oracle. Numayama et al. [11] relaxed Liskov's model and analyzed digital signature schemes in their model.

Therefore, previous works studied on effects into security of various protocols by vulnerability of hash functions itself. On the other hand, our LROM is different with these on the point of that LROM focuses on vulnerability of hash functions due to sloppy usages or implementations.

2 Leaky Random Oracle Model

ROM is one of techniques for provable security under idealized hash functions by using random oracles. Random oracle models truly idealized hash function which locally has the hash list of inputs and outputs. LROM is a variant of ROM which allows adversaries to obtain contents of the hash list in arbitrary timing. Thus, adversaries can correspond an input to the random oracle and an output (hash value). The definition of LROM is as follows;

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

Initialization: $\mathcal{L}_H \leftarrow \perp$
Hash query: *For a hash query* x_i *to* H*, behave as follows;*
$<$*If* $x_i \in \mathcal{L}_H$ $>$
Find y_i *corresponding to* x_i *from* $\mathcal{L}_H$ *and output* y_i *as the answer to the hash query.*
$<$*If* $x_i \notin \mathcal{L}_H$ $>$
Choose $y_i \in Y$ *randomly, add the pair* (x_i, y_i) *to* $\mathcal{L}_H$ *and output* y_i *as the answer to the hash query.*
Leak query: *For a leak query to* H*, output all contents of the hash list.*

3 Security Analysis of FDH in LROM

Full Domain Hash (FDH) [1] is secure signature scheme in ROM. In this section, we consider security of FDH in LROM.

3.1 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 $H : \{0,1\}^* \rightarrow Dom$ is a hash function.

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

In [1], security of FDH in ROM is proved as follows;

Lemma 1 (Security of FDH in ROM [1]). *If a trapdoor permutation f is one-way, then FDH is existentially unforgeable under adaptively chosen message attacks (EUF-ACMA) where H is modeled as the random oracle.*

3.2 Security of FDH in LROM

We can also prove the security of FDH in LROM like in ROM.

Theorem 1 (Security of FDH in LROM). *If a trapdoor permutation f is one-way, then FDH satisfies EUF-ACMA.*[1]

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 SO. Let $\mathcal{L}_H$ be the local hash list of the leaky random oracle H. $\mathcal{L}_H$ consists of tuples $(x_i, H(x_i), z_i)$ $(0 \leq i \leq q_H)$ where z_i is an intermediate value.[2] 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 $y \stackrel{R}{\leftarrow} Dom$
Output: $f^{-1}(y)$
Step 0: $i^* \stackrel{R}{\leftarrow} \{0, q_H - 1\}$, $\mathcal{L}_H \leftarrow \perp$ ($\perp$ is *null string*), $i \leftarrow 0$.
Step 1: Send f to $\mathcal{F}$ as the input.
Step 2: When $\mathcal{F}$ asks a hash query x_i to H, then behave as follows:
<If $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* \neq i)$ >

[1] In this paper, we omit concrete security bounds in the proof owing to lack of space.
[2] 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.

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 $((x_i, *, *) \notin \mathcal{L}_H) \wedge (i^* = i) >$
Generate $w_i \in Dom$. 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. $i \leftarrow i + 1$.

Step 3: When $\mathcal{F}$ asks a signing query x_i to 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. $i \leftarrow i + 1$.

Step 4: When $\mathcal{F}$ asks a leak query to H, then hand all pairs of input and output $\{(x, w)\}$ to $\mathcal{F}$. Note that, do not hand intermediate value z.

Step 5: 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 Step 4, $\mathcal{I}$ 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 Step 3, $\mathsf{Abort2}$ denote the event which $\mathcal{I}$ aborts in Step 5 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 Step 3 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 Step 3 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 Step 5 occurs with $\frac{1}{q_H}$ because the event occurs only in the case of that $y = f(\sigma^*)$ holds.

Thus, we obtain

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

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 $\mathcal{F}$. □

By the same reason, we can also prove security of PFDH [12] in LROM.

4 Security Analysis of OAEP in LROM

Optimal Asymmetric Encryption Padding (OAEP) [13] is secure padding scheme for asymmetric encryptions in ROM. In this section, we consider security of OAEP in LROM.

4.1 OAEP

OAEP is based on trapdoor partial-domain one-way permutations. We omit the detailed definition of trapdoor partial-domain one-way permutations. Please refer to [14].

The description of OAEP is as follows:

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

Encryption: For input a message $m \in \{0,1\}^n$, generate randomness $r \xleftarrow{R} \{0,1\}^{k_0}$, compute $x = (m||0^{k_1}) \oplus G(r)$ and $y = r \oplus H(x)$, and output a ciphertext $c = f(z)$ for $z = x||y$ 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, and $n = k - k_0 - k_1$.

Decryption: For inputs a ciphertext c, compute $z = f^{-1}(c)$, parse z as $x||y$ and reconstruct $r = y \oplus H(x)$ where $|x| = n+k_1$ and $|y| = k_0$. If $[x \oplus G(r)]_{k_1} \stackrel{?}{=} 0^{k_1}$ holds, output $m = [x \oplus G(r)]^n$ 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, reject the decryption as an invalid ciphertext.

In [14], security of OAEP in ROM is proved as follows;

Lemma 2 (Security of OAEP in ROM [14]). *If the trapdoor permutation f is partial-domain one-way, then OAEP satisfies IND-CCA where H and G are modeled as random oracles.*

4.2 Security of OAEP in LROM

OAEP is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show OAEP does not even satisfy one-wayness under chosen-plaintext attacks (OW-CPA) in LROM.

Theorem 2 (Security of OAEP in LROM). *Even if the trapdoor permutation f is partial-domain one-way, OAEP does not satisfy OW-CPA where H and G are modeled as leaky random oracles.*

Proof. We construct an adversary $\mathcal{A}$ which successfully plays OW-CPA game by using leak queries to H and G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $(x_i, H(x_i))$ $(0 \leq i \leq q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $(r_j, G(r_j))$ $(0 \leq j \leq q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input : f
Output : m^*
Step 1 : In arbitrary timing, output $(challenge, state)$ and obtain the challenge ciphertext c^* of a plaintext m^*.
Step 2 : Given input f and c^*, ask the leak query to H and G, obtain $\mathcal{L}_H$ and $\mathcal{L}_G$, and compute m^* as follows; For each content $(x_i, H(x_i))$, $(r_j, G(r_j))$ of $\mathcal{L}_H$ and $\mathcal{L}_G$, compute $c' = f(x_i||(r_j \oplus H(x_i)))$. If find the pair $((r^*, G(r^*)), (x^*, H(x^*)))$ such that $((c' = c^*) \wedge ([x_i \oplus G(r_j)]_{k_1} = 0^{k_1}))$ holds, compute $m^* = [x^* \oplus G(r^*)]^n$.
Step 3 : Output m^* as the plaintext of c^*.

Therefore, $\mathcal{A}$ can obtain m^* corresponding to c^*.

We show the success probability of $\mathcal{A}$. When m^* is encrypted to c^*, r^* and x^* such that $x^* = (m^*||0^{k_1}) \oplus G(r^*)$ are certainly asked to G and H respectively because c^* is generated obeying the protocol description. Thus, $\mathcal{L}_H$ and $\mathcal{L}_G$ contain the pair $((r^*, G(r^*)), (x^*, H(x^*)))$ such that $((c' = c^*) \wedge ([x_i \oplus G(r_j)]_{k_1} = 0^{k_1}))$ holds, and $\mathcal{A}$ can obtain m^* without fail. Therefore, $\mathcal{A}$ successfully plays the OW-CPA game. □

By the similar procedure (i.e., same procedure as the simulation of the decryption oracle in the proof in ROM), we can also show insecurity of Fujisaki-Okamoto conversion [15] in LROM.

5 Security Analysis of Cramer-Shoup cryptosystem in LROM

Cramer-Shoup cryptosystem [16] is secure asymmetric encryption scheme in SM. In this section, we consider security of Cramer-Shoup cryptosystem in LROM.

5.1 Cramer-Shoup Cryptosystem

Cramer-Shoup cryptosystem is based on the Decisional Diffie-Hellman (DDH) assumption and universal one-way hash function family. We omit the definition of DDH assumption and universal one-way hash function. Please refer to [16].

The description of Cramer-Shoup cryptosystem is as follows:

Key generation: For input k, generate a k-bit prime q. Choose $g_1, g_2 \in \mathbb{G}$ randomly and generate a decryption key $(dk = (x_1, x_2, y_1, y_2, z) \in \mathbb{Z}_q^5)$ and public information (c, d, h) such that $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^z$. Next, choose a hash function H from a family of universal one-way hash functions and output an encryption key $ek = (g_1, g_2, c, d, h, H)$ and the decryption key dk.

Encryption: For input a message $m \in \mathbb{G}$, choose $r \in_R \mathbb{Z}_q$, compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r m$, $\alpha = H(u_1, u_2, e)$ and $v = c^r d^{r\alpha}$, and output a ciphertext $c = (u_1, u_2, e, v)$.

Decryption: For inputs a ciphertext $c = (u_1, u_2, e, v)$, compute $\alpha = H(u_1, u_2, e)$ and verify whether $u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \stackrel{?}{=} v$ holds or not by using $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$. If the verification holds, then output the message $m = \frac{e}{u_1{}^z}$ by using $z \in \mathbb{Z}_q$. Else if, reject the decryption as an invalid ciphertext $\perp$.

In [16], security of Cramer-Shoup cryptosystem in SM is proved as follows;

Lemma 3 (Security of Cramer-Shoup cryptosystem in SM [16]). *If the hash function H is chosen from a family of universal one-way hash functions and the DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA.*

5.2 Security of Cramer-Shoup Cryptosystem in LROM

We can also prove security of Cramer-Shoup cryptosystem in LROM like in SM.

Theorem 3 (Security of Cramer-Shoup cryptosystem in LROM). *If the DDH assumption of the group $\mathbb{G}$ holds, then Cramer-Shoup cryptosystem satisfies IND-CCA where H is modeled as a leaky random oracle.*

Owing to lack of space, we will give the proof of Theorem 3 in the full version. The outline of the proof is similar to the proof of Lemma 3.

The intuition of the reason why we can prove the security of Cramer-Shoup cryptosystem in LROM as same as SM is as follows; In the case of OAEP, we can construct the successful adversary by applying the simulation of the decryption oracle in the proof in SM. However, in the case of Cramer-Shoup cryptosystem, the simulation of the decryption oracle does not need information of the hash lists. Moreover, all inputs and outputs of hash function H are publicly known because a ciphertext contains (u_1, u_2, e) which are the inputs to the hash function. Naturally, adversaries can know the input and the output in each session. Therefore, the leak query in LROM cannot be advantage of adversaries.

6 Security Analysis of Kurosawa-Desmedt Cryptosystem in LROM

Kurosawa-Desmedt cryptosystem [17] is secure hybrid encryption scheme in SM. In this section, we consider security of Kurosawa-Desmedt cryptosystem in LROM.

6.1 Kurosawa-Desmedt Cryptosystem

Kurosawa-Desmedt cryptosystem is based on the DDH assumption, a special type of universal one-way hash functions, and a symmetric key encryption scheme which satisfies IND-CCA and ϵ-rejection secure for negligible ϵ. We omit the

detailed definition of IND-CCA and ϵ-rejection for symmetric key encryption schemes. Please refer to [17]. The description of Kurosawa-Desmedt cryptosystem is as follows:

Key generation: For input k, randomly choose two distinct generators g_1, g_2 of $\mathbb{G}$ and $(x_1, x_2, y_1, y_2) \in \mathbb{Z}_q^4$, and compute $a = g_1^{x_1} g_2^{x_2}, b = g_1^{y_1} g_2^{y_2}$. Next, choose a hash function H from a family of a special type of universal one-way hash functions, and output an encryption key $ek = (g_1, g_2, a, b, H)$ and the decryption key $dk = (x_1, x_2, y_1, y_2)$.

Encryption: For input a message $m \in \{0,1\}^n$, generate randomness $r \xleftarrow{R} \mathbb{Z}_q$, compute $u_1 = g_1^r$, $u_2 = g_2^r$, $\alpha = H(u_1, u_2)$, $v = a^r b^{r\alpha}$, $K = G(v)$ and the encryption χ of m under the key K using a symmetric key encryption scheme **SKE**, and output a ciphertext $c = (u_1, u_2, \chi)$.

Decryption: For inputs a ciphertext c, compute $\alpha = H(u_1, u_2)$, $v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$, $K = G(v)$. Next, decrypt χ under the key K using **SKE** and output the resulting decryption.

In [17], security of Kurosawa-Desmedt cryptosystem in SM is proved as follows;

Lemma 4 (Security of Kurosawa-Desmedt cryptosystem in SM [17]) *If the hash function H is chosen from a family of a special type of universal one-way hash functions, the hash function G is uniformly distributed over $\{0,1\}^k$ if v is uniformly distributed over $\mathbb{G}$,* **SKE** *satisfies IND-CCA and ϵ-rejection secure for negligible ϵ, and the DDH assumption of the group $\mathbb{G}$ holds, then Kurosawa-Desmedt cryptosystem satisfies IND-CCA.*

6.2 Security of Kurosawa-Desmedt Cryptosystem in LROM

Kurosawa-Desmedt cryptosystem is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show Kurosawa-Desmedt cryptosystem does not even satisfy OW-CPA in LROM.

Theorem 4 (Security of Kurosawa-Desmedt cryptosystem in LROM). *Even if* **SKE** *satisfies IND-CCA and ϵ-rejection secure for negligible ϵ, and the DDH assumption of the group $\mathbb{G}$ holds, Kurosawa-Desmedt cryptosystem does not satisfy OW-CPA where H and G are modeled as leaky random oracles.*

Proof. We construct an adversary $\mathcal{A}$ which successfully plays OW-CPA game by using the leak query to G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $((u_1, u_2)_i, H((u_1, u_2)_i))$ $(0 \le i \le q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $(v_j, G(v_j))$ $(0 \le j \le q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input: g_1, g_2, a, b

Output: m^*

Step 1: Ask the leak query to G, obtain $\mathcal{L}_G$. Then, immediately, output $(challenge, state)$ and obtain the challenge ciphertext $c^* = (u_1, u_2, \chi)$ of a plaintext m^*.

Step 2: Ask again the leak query to G, obtain $\mathcal{L}'_G$, and compare $\mathcal{L}_G$ and $\mathcal{L}'_G$. If there is a content $(v^*, G(v^*))$ in $\mathcal{L}'_G$ but $(v^*, G(v^*))$ is not in $\mathcal{L}_G$, deal with $G(v^*)$ as K^*. Then, decrypt χ under the key K^* using **SKE** and output the resulting decryption m^*.

Step 3: Output m^* as the plaintext of c^*.

Therefore, $\mathcal{A}$ can obtain m^* corresponding to c^*.

We show the success probability of $\mathcal{A}$. When m^* is encrypted to c^*, v^* such that $K^* = G(v^*)$ is certainly asked to G because c^* is generated obeying the protocol description. Thus, $\mathcal{L}_G$ contains $(v^*, G(v^*))$ such that c^* is the ciphertext of the plaintext m^*, and $\mathcal{A}$ can obtain m^* without fail by observing the hash list of G step by step. Therefore, $\mathcal{A}$ successfully plays the OW-CPA game. □

7 Security Analysis of NAXOS in LROM

NAXOS [18] is a secure authenticated key exchange scheme against the leakage of ephemeral private keys (session-specific secret information) in ROM. In this section, we consider security of NAXOS and similar schemes in LROM.

7.1 Security Notion of Authenticated Key Exchange Schemes

Security definitions of authenticated key exchange schemes are studied in many literatures. NAXOS is proven to be secure in the sense of a strong definition, called strong AKE security. Strong AKE security captures various desirable security requirements like resistance to the leakage of ephemeral private keys. We omit the detailed definition of strong AKE security. Please refer to [18]. Here, we define a very weak security notion of authenticated key exchange schemes as follows.

Definition 2 (One-way security against passive attacks). *An authenticated key exchange scheme for parties I and R is one-way secure against passive attacks if the following property holds; For any adversary $\mathcal{A}$,* $\Pr[\,(SK, transcript) \leftarrow \langle I \Leftrightarrow R\rangle;\ SK' \leftarrow \mathcal{A}(transcript);\ SK' = SK] \leq negl.$*, where $\langle I \Leftrightarrow R\rangle$ is a honest execution of the scheme outputting the transcript between I and R and the session key SK.*

Note that, this definition only captures the minimum security requirement for authenticated key exchange schemes.

7.2 NAXOS

NAXOS is based on the Gap Diffie-Hellman (GDH) assumption. We omit the detailed definition of the GDH assumption. Please refer to [19]. The description of NAXOS is as follows:

Interaction: For input k, the parties I and R pick ephemeral secret keys esk_I and esk_R at random from $\{0,1\}^k$. Then the parties exchange values $g^{H(esk_I, sk_I)}$ and $g^{H(esk_R, sk_R)}$ where sk_I and sk_R are static secret keys of I and R respectively, and $H : \{0,1\}^* \to \mathbb{Z}_q$ is a hash function.

Key derivation: The parties check if received values are in the group $\mathbb{G}$ and only compute the session keys if the check succeeds. The session key $SK \in \{0,1\}^k$ is computed as $G(g^{H(esk_R,sk_R)sk_I}, g^{H(esk_I,sk_I)sk_R}, g^{H(esk_I,sk_I)H(esk_R,sk_R)}, I, R)$ where $G : \{0,1\}^* \to \{0,1\}^k$ is a hash function.

In [18], security of NAXOS in ROM is proved as follows;

Lemma 5 (Security of NAXOS in ROM [18]). *If the GDH assumption of the group $\mathbb{G}$ holds, then NAXOS satisfies strong AKE security where H and G are modeled as random oracles.*

7.3 Security of NAXOS in LROM

NAXOS is secure in ROM but, indeed, is insecure in LROM. More specifically, we can show NAXOS does not even satisfy one-way security against passive attacks in LROM.

Theorem 5 (Security of NAXOS in LROM). *Even if the GDH assumption of the group $\mathbb{G}$ holds, NAXOS does not satisfy one-way security against passive attacks where H and G are modeled as leaky random oracles.*

Proof. We construct an passive adversary $\mathcal{A}$ which successfully plays one-way security game by using leak queries to H and G. Let $\mathcal{L}_H$ and $\mathcal{L}_G$ be hash lists of leaky random oracles H and G respectively. $\mathcal{L}_H$ contains tuples of $((esk, sk)_i, H((esk, sk)_i))$ $(0 \le i \le q_H - 1)$ and $\mathcal{L}_G$ contains tuples of $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ $(0 \le j \le q_G - 1)$ where q_H is the number of queries to H and q_G is the number of queries to G. The construction of $\mathcal{A}$ is as follows;

Input: $transcript^* = (g^{H(esk_I^*,sk_I^*)}$ and $g^{H(esk_R^*,sk_R^*)})$

Output: SK^*

Step 1: Ask the leak query to H and obtain $\mathcal{L}_H$. For each content $((esk, sk)_i, H((esk, sk)_i))$ of $\mathcal{L}_H$, if find the pair i_1 and i_2 such that $g^{H((esk,sk)_{i_1})} = g^{H(esk_I^*,sk_I^*)}$ and $g^{H((esk,sk)_{i_2})} = g^{H(esk_R^*,sk_R^*)}$ then compute $G_1^* = (g^{H(esk_R^*,sk_R^*)})^{sk_{i_1}}$, $G_2^* = (g^{H(esk_I^*,sk_I^*)})^{sk_{i_2}}$ and $G_3^* = g^{H((esk,sk)_{i_1})H((esk,sk)_{i_2})}$.

Step 2: Ask the leak query to G and obtain $\mathcal{L}_G$. For each content $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ of $\mathcal{L}_G$, if find j such that $G_1 = G_1^*$, $G_2 = G_2^*$ and $G_3 = G_3^*$ then deal with $G((G_1, G_2, G_3, ID_1, ID_2)_j)$ as SK^*.

Step 3: Output SK^* as the session key.

Therefore, $\mathcal{A}$ can obtain SK^* of the challenge session.

We show the success probability of $\mathcal{A}$. When SK^* is generated, $(esk, sk)_{i_1}$, $(esk, sk)_{i_2}$ and $(G_1^*, G_2^*, G_3^*, I, R)_j$ such that $g^{H((esk,sk)_{i_1})} = g^{H(esk_I^*,sk_I^*)}$, $g^{H((esk,sk)_{i_2})} = g^{H(esk_R^*,sk_R^*)}$, $G_1 = G_1^*$, $G_2 = G_2^*$ and $G_3 = G_3^*$ are certainly asked to H and G because SK^* is generated obeying the protocol description. Thus, $\mathcal{L}_G$ contains $((G_1, G_2, G_3, ID_1, ID_2)_j, G((G_1, G_2, G_3, ID_1, ID_2)_j))$ such that $SK^* = G((G_1, G_2, G_3, ID_1, ID_2)_j)$, and $\mathcal{A}$ can obtain SK^* without fail. Therefore, $\mathcal{A}$ successfully plays the one-way security game under the passive attack. □

By the similar procedure (i.e., obtaining all secret information), we can also show insecurity of CMQV [20] in LROM.

8 Discussion

8.1 Difference of Effects on Security

In LROM, though FDH and Cramer-Shoup cryptosystem can be proven security, OAEP, Kurosawa-Desmedt cryptosystem and NAXOS are insecure.

Our attack to OAEP in LROM is based on the simulation of the decryption oracle in the proof of Lemma 2. Most of asymmetric encryption schemes which are secure in ROM realize the simulation of the decryption oracle in the proof without knowledge of the decryption key by using contents of hash lists. Therefore, by the same behavior as the simulator in the proof in ROM, the adversary can decrypt the challenge ciphertext without knowledge of the decryption key because the adversary can observe contents of the hash list in LROM. Our attack to Kurosawa-Desmedt cryptosystem is simpler than one to OAEP. Since Kurosawa-Desmedt cryptosystem can be proven security in SM, the simulation of the decryption oracle does not need to use contents of hash lists. However, the hash value of G has to be secret for external entities because the hash value is used the key of symmetric key encryption part. Thus, if contents of the hash list are leaked, the plaintext of any ciphertext is easily decrypted by adversaries. Also, NAXOS falls into the similar condition as Kurosawa-Desmedt cryptosystem, i.e., all secret information of parties are leaked from the hash list.

On the other hand, FDH is different with these protocols in the following points; Firstly, though we have to simulate the signing oracle without knowledge of the signing key in the security proof of FDH, the simulation does not need to use contents of the hash list. Secondly, in the signing procedure the input of the hash function and the corresponding hash value are not secret information because the input is the message to be signed. Therefore, if contents of the hash list is available to adversaries, it does not become any advantage of adversaries. Thus, we can prove security of FDH in LROM. The case of Cramer-Shoup cryptosystem is also similar to the case of FDH. Since Cramer-Shoup cryptosystem can be proven security in SM, we can simulate the decryption oracle without contents of the hash list. Moreover, in the encryption procedure the inputs of the hash function and the corresponding hash value are not secret information because the inputs are contained in the ciphertext. Thus, we can prove security of Cramer-Shoup cryptosystem in LROM.

Hence, in order to prove security of a protocol in LROM, it is important that we can realize all the simulation without contents of the hash list, and the input of the hash function and the corresponding hash value are not secret information.

8.2 Relation between the Standard Model

From the modeling, the proof of a protocol in LROM implies the proof of the protocol in ROM trivially. Moreover, LROM is independent from SM. Our result of analyses shows the separation between LROM and SM because of two

following observations; Firstly, FDH is secure in LROM under the assumption of trapdoor permutation. However, Dodis et al. [6] showed that FDH is not provable in SM under the same assumption. Thus, we obtain that the proof of a protocol in LROM does not implies the proof of the protocol in SM.

Next, Kurosawa-Desmedt cryptosystem is secure in SM under the DDH assumption, the assumption of universal hash function family and the assumption of symmetric key encryption. However, Kurosawa-Desmedt cryptosystem is insecure in LROM by instantiating hash functions by leaky random oracles under same assumptions. Thus, we obtain that the proof of a protocol in SM does not implies the proof of the protocol in LROM.

Therefore, LROM is independent from SM. We have to check whether a new protocol is secure in LROM even if the protocol is known to be secure in SM.

8.3 Relation between Randomness Revealing

At first sight, it may seem that the leakage of contents of the hash list is corresponding to randomness revealing because it often happen that the inputs of hash functions contains local randomness. Thus, it may seem that LROM is same as ROM under randomness revealing. Indeed, these are different. Our result of analyses shows the difference between LROM and ROM under randomness revealing because of the following observation;

NAXOS is secure in ROM under the leakage of local randomness. However, NAXOS is insecure in LROM even if there is no leakage of local randomness. Thus, we obtain that the proof of a protocol in ROM under randomness revealing does not implies the proof of the protocol in LROM.

Therefore, we have to check whether a new protocol is secure or not in LROM even if the protocol is known to be secure in ROM under randomness revealing.

9 Further Works

A remaining problem of future works is more detailed analyses of protocols under the leakage. For example, though OAEP is insecure if both random oracles H and G are instantiated by leaky random oracles, OAEP may be secure if either of two random oracles is only instantiated by the leaky random oracle. Indeed, Boldyreva and Fischlin [21] showed that OAEP is secure if either of two random oracles is instantiated by the real hash function and the other remain as the random oracle.

References

1. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security 1993, pp. 62–73 (1993)
2. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited (Preliminary Version). In: STOC 1998, pp. 131–140 (1998)
3. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. J. ACM 51(4), 557–594 (2004)

4. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
5. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest We Remember: Cold Boot Attacks on Encryption Keys. In: 17th USENIX Security Symposium, pp. 45–60 (2008)
6. Dodis, Y., Oliveira, R., Pietrzak, K.: On the Generic Insecurity of the Full Domain Hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
7. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
8. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
9. Pasini, S., Vaudenay, S.: Hash-and-Sign with Weak Hashing Made Secure. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 338–354. Springer, Heidelberg (2007)
10. Unruh, D.: Random Oracles and Auxiliary Input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007)
11. Numayama, A., Isshiki, T., Tanaka, K.: Security of Digital Signature Schemes in Weakened Random Oracle Models. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 268–287. Springer, Heidelberg (2008)
12. Coron, J.S.: Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
13. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
14. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP Is Secure under the RSA Assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 260–274. Springer, Heidelberg (2001)
15. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
16. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
17. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
18. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Provsec 2007, pp. 1–16 (2007)
19. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
20. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. In: Des. Codes Cryptography, vol. 46(3), pp. 329–342 (2008)
21. Boldyreva, A., Fischlin, M.: Analysis of Random Oracle Instantiation Scenarios for OAEP and Other Practical Schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005)