Research Reports on Mathematical and Computing Sciences

The Security with the Randomness Revealed
for Public-Key Encryption

Ryotaro Hayashi and Keisuke Tanaka

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

The Security with the Randomness Revealed
for Public-Key Encryption

Ryotaro Hayashi and Keisuke Tanaka

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{hayashi9, keisuke}@is.titech.ac.jp

November 10, 2006

Abstract

We consider the situation for public-key encryption that the adversary knows the randomness which was used to compute the ciphertext. In some practical scenarios, there is a possibility that the randomness is revealed. For example, the randomness used to make a ciphertext may be stored in insecure memory, or the pseudorandom generator may be corrupted. We first formalize the security notion on this situation as "the one-wayness with the randomness revealed." In addition to the formalization, we focus on two schemes, the generic chosen-ciphertext secure encryption method (GEM) and 3-round OAEP, and prove that these two schemes satisfy our security notions.

Keywords: public-key encryption, randomness, security, one-wayness.

1 Introduction

The classical security requirement of public-key encryption schemes is that it provides privacy of the encrypted data. Popular formalizations such as one-wayness, indistinguishability, or non-malleability, under either the chosen plaintext attack or the adaptive chosen ciphertext attack are directed at capturing various data-privacy requirements. The widely admitted appropriate security level for public-key encryption is the indistinguishability against the adaptive chosen ciphertext attack (IND-CCA). Up until now, many public-key encryption schemes have been proposed which are secure in the sense of IND-CCA.

In Eurocrypt 2001, Canetti and Krawczyk [4] proposed a security model of key-exchange protocols. The key-exchange protocols allow two parties who share no secret information (or share short password) to compute a session key via public communication. In the protocol, two parties usually use two kinds of secret information, that is, the long-term secret-key and the ephemeral key which is state-specific secret information used within each session. The ephemeral key contains the randomness which is used in the session, for example. The Canetti–Krawczyk model gives the adversary the power to reveal the ephemeral key (session-state reveal query) without revealing the long-term secret key. Session-state reveal queries are motivated by practical scenarios, such as if the state-specific secret information is stored in insecure memory or if the random-number generator of a party is corrupted. In both of these cases, as a result of the specific vulnerability, the adversary might be able to gain access to the ephemeral data. In these scenarios, for example, the (standard) signed Diffie–Hellman authenticated key-exchange protocol is not secure, that is, the adversary can compute the session key (See [11].). Recently, Krawczyk [10] proposed a key-exchange protocol called HMQV which is secure even if the adversary can make session-state

reveal queries. Lauter and Mityagin [11] also proposed a key-exchange protocol KEA+ which is secure against the attack with session-state reveal queries.

As we have seen above, the security for key-exchange protocols in the case that randomness is revealed was widely studied. In this paper, we consider a similar situation for public-key encryption, that is, the adversary knows the randomness which was used to compute the ciphertext. In some practical scenarios, there is a possibility that the randomness is revealed. Similar to the case of key-exchange protocols, the randomness used to make a ciphertext may be stored in insecure memory or the pseudorandom generator may be corrupted. Furthermore, the distribution of the randomness may not be uniform, and has some bias by using a pseudorandom generator. Then, we may decide the randomness used to compute the ciphertext. (As a research concerning the randomness of encryption scheme, Bosley and Dodis [3] considered the question whether privacy for the symmetric-key encryption requires true randomness.)

There are many public-key encryption schemes which are secure in the sense of IND-CPA or IND-CCA. Unfortunately, many public-key encryption schemes, which satisfy IND-CPA or IND-CCA, can be broken when the randomness is revealed. For example, consider the encryption function $E(m; r) := (g^r, m \cdot y^r)$ of the ElGamal encryption scheme with the public-key y. If the adversary knows the randomness r of the ciphertext (c_1, c_2), the adversary easily gets the plaintext by computing $m = c_2/y^r$. We can apply a similar discussion to the Cramer-Shoup encryption scheme [6] and its variants generated from the general framework proposed by Cramer and Shoup [7].

We can also see that the Paillier's encryption scheme [13] is not secure if the randomness is revealed. The encryption function of the Paillier's encryption scheme is $E(m; r) := (1 + mN)r^N \bmod N^2$ where N is the public-key. If the the adversary knows the randomness r of the ciphertext c, the adversary can get the plaintext by computing $m = (T - 1)/N$ where $T = c/r^N \bmod N^2$.

Furthermore, some schemes whose security depend on the random oracles can be broken in the situation that the randomness is revealed, even if the random oracles are used. For example, consider the Fujisaki-Okamoto conversion. The encryption function of this scheme is defined as $E_{pk}(m; r) := E_{pk}^{\mathrm{pub}}(r; H(r, m))||E_{G(r)}^{\mathrm{sym}}(m)$ where E_{pk}^{pub} is a public-key encryption scheme with a public-key pk, $E_{G(r)}^{\mathrm{sym}}$ is a symmetric-key encryption scheme with the symmetric-key $G(r)$, and G, H are hash functions (modeled as the random oracles). In this conversion, the randomness r is encrypted by the public-key encryption scheme, and the plaintext m is masked by the hash value $G(r)$ of the randomness r. We can easily see that the adversary can compute the plaintext if the randomness r is revealed. Similarly, we can easily see that REACT [12] and the scheme by Pointcheval [15] are also not secure if the randomness is revealed.

We next consider OAEP proposed by Bellare and Rogaway [2]. Let φ_{pk} be a trap-door permutation and n the length of the plaintext. Then, the encryption function of OAEP is $E(m; r) := \varphi_{\mathsf{pk}}(s||t)$ where $s = (m||0^k) \oplus G(r)$, $t = r \oplus H(s)$. Roughly speaking, Fujisaki, Okamoto, Pointcheval, and Stern [8] showed that OAEP with a trap-door permutation is secure in the sense of IND-CCA2 under the assumption that no poly-time algorithm, given $c = \varphi_{\mathsf{pk}}(s||t)$, can compute s with non-negligible probability. However, we can see that the above assumption is not sufficient to prove that OAEP with a trap-door permutation is secure when the randomness is revealed.

We consider a trap-door permutation $\hat{\varphi}_{\mathsf{pk}}(x_1||x_2||x_3) := x_1||\varphi'_{\mathsf{pk}}(x_2)||x_3$ where $|x_1| = n$, $|x_2| = k$, and φ'_{pk} is one-way. We can see that no poly-time algorithm can compute $x_1||x_2$ from $\hat{\varphi}_{\mathsf{pk}}(x_1||x_2||x_3)$ with non-negligible probability if k is sufficiently large and φ'_{pk} is one-way. That is, OAEP with $\hat{\varphi}_{\mathsf{pk}}$ satisfies IND-CCA2. However, even in this case, if the randomness is revealed, we can decrypt the plaintext m without the secret key. We can compute $G(r)$, since the randomness r is revealed, and compute the most n significant bits of the preimage of $c = \hat{\varphi}_{\mathsf{pk}}(s||t)$ where c is a ciphertext, that is, the most n significant bits of s. Thus, we can compute the plaintext as

$m = $ [the most n significant bits of $G(r)$] $\oplus$ [the most n significant bits of s]. We can apply a similar discussion to OAEP+ [16], which satisfies IND-CCA2 if the underlying trap-door permutation is one-way. That is, OAEP+ can be broken if the randomness is revealed, even if the permutation is one-way.

In this paper, we first formalize the security notion for public-key encryption in the situation that the randomness is revealed. The most popular formalization for public-key encryption is the indistinguishability (IND). However, it is easy to see that the adversary can always win the IND game when the randomness is revealed. If the adversary has a pair (m_0, m_1) of messages, a ciphertext c of either m_0 or m_1, and a randomness r used to compute c, the adversary can know which message was encrypted by checking whether $c = E(m_0; r)$ or $c = E(m_1; r)$. Thus, the indistinguishability does not fit the situation that the randomness is revealed. In this paper, we formalize the security notion as "the one-wayness with the randomness revealed." Roughly speaking, a public-key encryption scheme satisfies this security if no poly-time algorithm can decrypt c (without using the secret-key) even if the randomness used to compute c is given to the algorithm. In particular, we propose three security notions, the one-wayness with the randomness revealed against the chosen message attack (OWR-CPA), that against the plaintext checking attack (OWR-PCA), and that against the adaptive chosen ciphertext attack (OWR-CCA).

In addition to the formalization, we also consider the schemes which satisfy our security notions. There might be many such existing schemes, however, in this paper, we focus on two schemes. One is the generic chosen-ciphertext secure encryption method (GEM) proposed by Coron, Handschuh, Joye, Paillier, Pointcheval, and Tymen [5]. This is a conversion to transform a public-key encryption scheme which satisfies the one-wayness against the plaintext checking attack (OW-PCA) into that which satisfies IND-CCA in the random oracle model. We show that GEM satisfies OWR-CCA if the underlying scheme satisfies OWR-PCA in the random oracle model.

The other is 3-round OAEP with a trap-door permutation proposed by Phan and Pointcheval [14]. 3-round OAEP satisfies IND-CCA in the random oracle model if the underlying trap-door permutation is partial one-way. Furthermore, the scheme does not have the usual redundancy which OAEP has. We show that this scheme satisfies OWR-CCA if the trap-door permutation is partial one-way.

The organization of this paper is as follows. In Section 2, we review some definitions. In Section 3, we formalize the one-wayness with the randomness revealed. In Section 4, we review GEM and prove that GEM satisfies OWR-CCA if the underlying scheme satisfies OWR-PCA in the random oracle model. In Section 5, we review 3-round OAEP and prove that 3-round OAEP with a partial one-way trap-door permutation provides OWR-CCA in the random oracle model. We conclude in Section 6.

2 Preliminaries

In this paper, we use the following notations. If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots, x_n; r)$ is the output of A on inputs $x_1, x_2, \ldots, x_n$ and coins r. We let $y \leftarrow A(x_1, x_2, \ldots, x_n)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots, x_n; r)$. If S is a finite set then $x \xleftarrow{R} S$ is the operation of picking an element uniformly from S. If α is not an algorithm then $x \leftarrow \alpha$ is a simple assignment statement.

We say that the function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is negligible (in k) if for every constant $c > 0$ there exists an integer k' such that $\epsilon(k) < 1/k^c$ for all $k \geq k'$.

2.1 Public-Key Encryption Schemes

In this section, we briefly review the definition of public-key encryption schemes.

Definition 1. *A public-key encryption scheme* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *consists of three algorithms.*

- *The key generation algorithm $\mathcal{K}$ is a randomized algorithm that takes as input a security parameter k and returns a pair (pk, sk) of keys, a public key and a matching secret key. For given pk, a message space $\mathtt{MSPC}_\Pi(pk)$ and a randomness space $\mathtt{RSPC}_\Pi(pk)$ are uniquely determined.*

- *The encryption algorithm $\mathcal{E}$ is a randomized algorithm that takes the public key pk and a plaintext $m \in \mathtt{MSPC}_\Pi(pk)$ and returns a ciphertext c, using a random coin $r \xleftarrow{R} \mathtt{RSPC}_\Pi(pk)$.*

- *The decryption algorithm $\mathcal{D}$ is a deterministic algorithm that takes the secret key sk and a ciphertext c and returns the corresponding plaintext m or a special symbol $\perp$ to indicate that the ciphertext is invalid.*

2.2 Symmetric-Key Encryption

In this section, we review the definition of symmetric-key encryption schemes.

Definition 2. *A symmetric-key encryption scheme $\mathtt{SE} = (\mathtt{K}, \mathtt{E}, \mathtt{D})$ consists of three algorithms.*

- *The key generation algorithm $\mathtt{K}$ is a probabilistic algorithm that takes a security parameter 1^k and returns a symmetric key K. For given K, a message space $\mathtt{MSPC}_{\mathtt{SE}}(K)$ is uniquely determined.*

- *The encryption algorithm $\mathtt{E}$ is a deterministic algorithm that takes a symmetric-key K and a message $m \in \mathtt{MSPC}_{\mathtt{SE}}(K)$, and returns a ciphertext c.*

- *The decryption algorithm $\mathtt{D}$ is a deterministic algorithm that takes a symmetric key K and a ciphertext c, and returns the corresponding plaintext m.*

We next review a security notion for symmetric-key encryption, called indistinguishability. Note that in the following definition we only consider the passive adversary.

Definition 3. *Let $\mathtt{SE} = (\mathtt{K}, \mathtt{E}, \mathtt{D})$ be a symmetric-key encryption scheme. Let A be an adversary that runs in two stages, find and guess. We define the advantage of A via*

$$\mathbf{Adv}_{\mathtt{SE}, A}^{\mathsf{ind}}(k) = 2 \cdot \Pr[K \leftarrow \mathtt{K}(1^k);\ (m_0, m_1, s) \leftarrow A(\mathsf{find}, k);$$
$$b \xleftarrow{R} \{0, 1\};\ c \leftarrow \mathtt{E}_K(m_b)\ :\ A(\mathsf{guess}, c, s) = b] - 1.$$

We require that $m_0 \neq m_1$ and $m_0, m_1 \in \mathtt{MSPC}_{\mathtt{SE}}(K)$. We say that $\mathtt{SE}$ is secure in the sense of IND if $\mathbf{Adv}_{\mathtt{SE}, A}^{\mathsf{ind}}(k)$ is negligible for any polynomial-time adversary A.

2.3 Families of Trap-Door Permutations

In this section, we review the definitions of families of trap-door permutations and θ-partial one-wayness.

Definition 4 (Families of Trap-Door Permutations). *A family of trap-door permutations $\mathcal{TP} = (K, \varphi, \psi)$ is described as follows. The key generation algorithm K takes as input a security parameter 1^k and outputs a public key pk and a matching secret key (trap-door) sk. The function φ_{pk} is a permutation over $\mathtt{Dom}_{\mathcal{TP}}(\mathsf{pk})$ where $\mathtt{Dom}_{\mathcal{TP}}(\mathsf{pk})$ is uniquely determined by pk, while ψ_{sk} is the inverse permutation of φ_{pk}, that is, $\varphi_{\mathsf{pk}}^{-1} = \psi_{\mathsf{sk}}$.*

Definition 5 (θ-Partial One-Wayness). *Let $k \in \mathbb{N}$ be a security parameter, and $0 < \theta \leq 1$ a constant. Let $\mathcal{TP} = (K, \varphi, \psi)$ be a family of trap-door permutations, and A an adversary. We consider the following experiment:*

$$
\begin{aligned}
&\texttt{Experiment } \mathbf{Exp}_{\mathcal{TP},A}^{\theta\text{-pow}}(k) \\
&\quad (\mathsf{pk}, \mathsf{sk}) \leftarrow K(1^k); \ x \xleftarrow{R} \mathtt{Dom}_{\mathcal{TP}}(\mathsf{pk}); \ y \leftarrow \varphi_{\mathsf{pk}}(x) \\
&\quad x_1 \leftarrow A(\mathsf{pk}, y) \texttt{ where } |x_1| = \lceil \theta \cdot |x| \rceil \\
&\quad \texttt{if } (\varphi_{\mathsf{pk}}(x_1 \| x_2) = y \texttt{ for some } x_2) \texttt{ return 1 else return 0}
\end{aligned}
$$

Here, "$\|$" denotes concatenation. We define the advantage of the adversary via

$$
\mathbf{Adv}_{\mathcal{TP},A}^{\theta\text{-pow}}(k) = \Pr[\mathbf{Exp}_{\mathcal{TP},A}^{\theta\text{-pow}}(k) = 1]
$$

where the probability is taken over K, $x \xleftarrow{R} \mathtt{Dom}_{\mathcal{TP}}(\mathsf{pk})$, and A. We say that $\mathcal{TP}$ is θ-partial one-way if the function $\mathbf{Adv}_{\mathcal{TP},A}^{\theta\text{-pow}}(k)$ is negligible for any polynomial-time adversary A.

Note that when $\theta = 1$ the notion of θ-partial one-wayness means the standard notion of one-wayness.

3 The One-Wayness with the Randomness Revealed

In this section, we formalize the one-wayness with the randomness revealed (OWR).

Definition 6 (OWR). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $A_{\mathrm{cpa}}, A_{\mathrm{pca}}, A_{\mathrm{cca}}$ be the adversaries. The adversaries $A_{\mathrm{cpa}}, A_{\mathrm{pca}},$ and A_{cca} can access to the oracles $\mathcal{O}_{\mathrm{cpa}}, \mathcal{O}_{\mathrm{pca}},$ and $\mathcal{O}_{\mathrm{cca}}$, respectively. For atk $\in \{$cpa, pca, cca$\}$, we define the advantages of A_{atk} via*

$$
\mathbf{Adv}_{\Pi,A_{\mathrm{atk}}}^{\mathrm{owr\text{-}atk}}(k) = \Pr[(pk, sk) \leftarrow \mathcal{K}(1^k); \ m \xleftarrow{R} \mathtt{MSPC}_{\Pi}(pk); \ c \leftarrow \mathcal{E}_{pk}(m; r) \ : \ A_{\mathrm{atk}}^{\mathcal{O}_{\mathrm{atk}}}(c, r, pk) = m]
$$

where $\mathcal{O}_{\mathrm{cpa}} = \epsilon$ and $\mathcal{O}_{\mathrm{cca}} = \mathcal{D}_{sk}$. $\mathcal{O}_{\mathrm{pca}}$ is the plaintext checking oracle which takes (m, c) as input and returns 1 if m is a plaintext of c, that is, $m = \mathcal{D}_{sk}(c)$. Otherwise, it returns 0. We require that A_{cca} never queries the challenge c to $\mathcal{D}_{sk}$.

We say that Π is secure in the sense of OWR-CPA (resp. OWR-PCA, OWR-CCA) if $\mathbf{Adv}_{\Pi,A_{\mathrm{cpa}}}^{\mathrm{owr\text{-}cpa}}(k)$ (resp. $\mathbf{Adv}_{\Pi,A_{\mathrm{pca}}}^{\mathrm{owr\text{-}pca}}(k)$, $\mathbf{Adv}_{\Pi,A_{\mathrm{cca}}}^{\mathrm{owr\text{-}cca}}(k)$) is negligible for any polynomial-time adversary A_{cpa} (resp. $A_{\mathrm{pca}}, A_{\mathrm{cca}}$).

In the above definition, the adversary gets the randomness r used to compute the ciphertext c, while in the definition of the standard one-wayness, the adversary never gets the randomness r.

4 GEM and its Security

In this section, we review the generic chosen-ciphertext secure encryption method (GEM) proposed by Coron, Handschuh, Joye, Paillier, Pointcheval, and Tymen [5], and prove that GEM satisfies OWR-CCA if the underlying scheme satisfies OWR-PCA in the random oracle model.

4.1 GEM

In this section, we review GEM by Coron, Handschuh, Joye, Paillier, Pointcheval, and Tymen [5]. GEM is a generic conversion to transform a public-key encryption scheme which satisfies the one-wayness against the plaintext checking attack (OW-PCA) into that which satisfies IND-CCA in the random oracle model.

Definition 7 (GEM). *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a public-key encryption scheme where* $\mathtt{MSPC}_\Pi(pk) = \{0,1\}^{k+\ell}$. *Let* $\mathtt{SE} = (\mathtt{K}, \mathtt{E}, \mathtt{D})$ *be a length-preserving symmetric-key encryption scheme where* $\mathtt{MSPC}_{\mathtt{SE}}(K) = \{0,1\}^n$. *Let* F, G, H *be hash functions. The public-key encryption scheme* $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ *of GEM is as follows. The key generation algorithm* $\mathcal{K}'$ *is the same as* $\mathcal{K}$. *The encryption and the decryption algorithms are as follows.*

Algorithm $\mathcal{E}'_{pk}(m; (r\|u))$	Algorithm $\mathcal{D}'_{sk}(c_1, c_2)$
$r \xleftarrow{R} \{0,1\}^\ell;\ u \xleftarrow{R} \mathtt{RSPC}_\Pi(pk)$	$w \leftarrow \mathcal{D}_{sk}(c_1)$
$s \leftarrow F(m, r)$	$s\|t \leftarrow w$ where $\|s\| = k$ and $\|t\| = \ell$
$w \leftarrow s\|(r \oplus H(s))$	$K \leftarrow G(w, c_1)$
$c_1 \leftarrow \mathcal{E}_{pk}(w; u)$	$r \leftarrow t \oplus H(s)$
$K \leftarrow G(w, c_1)$	$m \leftarrow \mathtt{D}_K(c_2)$
$c_2 \leftarrow \mathtt{E}_K(m)$	if $(s = F(m, r))$ then return m
return (c_1, c_2)	else return $\perp$

4.2 OWR-CCA security of GEM

In this section, we prove that GEM satisfies OWR-CCA if the underlying scheme satisfies OWR-PCA in the random oracle model.

Theorem 1. *Suppose that there exists the adversary A that breaks the OWR-CCA security of Π', within time bound τ, after at most q_f, q_g, q_h, q_d queries to the random oracles F, G, H, and the decryption oracle, respectively, and with advantage $\mathbf{Adv}_{\Pi',A}^{\text{owr-cca}}(k) = \epsilon$. Then, for all $0 < \nu < \epsilon$, there exists*

- *an adversary C that breaks the indistinguishability of $\mathtt{SE}$ within time bound τ and advantage $\mathbf{Adv}_{\mathtt{SE},C}^{\text{ind}}(k) = \nu$, or*

- *an adversary B with access to the plaintext checking oracle $\mathcal{O}_{\text{pca}}$ (responding in time bound τ_{pca}) that breaks the OWR-PCA security of Π within time bound*

$$\tau' \leq \tau + (q_d(q_f + q_g) + q_h)(\tau_{\text{pca}} + O(1)),$$

after at most

$$q_{\text{pca}} \leq q_d(q_f + q_g)$$

queries to $\mathcal{O}_{\text{pca}}$, and with success probability

$$\mathbf{Adv}_{\Pi,B}^{\text{owr-pca}}(k) \geq \frac{\epsilon - \nu}{2} - \frac{q_f}{2^\ell} - q_d\left(\frac{1}{2^k} + q_f\left(\frac{1}{2^\ell} + \frac{1}{2^k}\right) + \nu + \frac{1}{2^n}\right).$$

Proof. The proof is similar to that of the IND-CCA security of GEM in [5].

We construct the PCA-adversary B attacking the one-wayness with the randomness revealed of Π, by using the CCA-adversary A attacking the one-wayness with the randomness revealed of Π'. The algorithm B is given pk, $c_1^* = \mathcal{E}(w^*; u^*)$, and u^*, and trying to compute w^* as follows.

1. B picks two randomnesses m^*, r^*, and sets $K^* \xleftarrow{R} \mathtt{K}(1^k)$ and $c_2^* \leftarrow \mathtt{E}_{K^*}(m^*)$.

2. B initializes three lists, called F-List, G-List, and H-List to empty, and sets $\tilde{w} \leftarrow \perp$.

3. B runs A with the public-key pk, the ciphertext (c_1^*, c_2^*), and the randomness $r^*\|u^*$, and gets $\tilde{m}$ which is the output of A. Note that B simulates A's oracles F, G, H, $\mathcal{D}'_{sk}$ as described below, and the value $\tilde{w}$ may be rewrited during the simulation of these oracles.

4. B returns $\tilde{w}$.

B simulates the random oracles F, G, H, and the decryption oracle as follows.

- When a query (m, r) is made to F, if there exists a pair $((m, r), f) \in F$-List then we respond f. Otherwise, if $m = m^*$ and there exists $(s, h) \in H$-List such that $c_1^* = \mathcal{E}_{pk}(s||(r \oplus h); u^*)$, we respond s, set $\tilde{w} \leftarrow s||(r \oplus h)$, and put (r, f) into F-List. Otherwise, we respond a random string f and put (r, f) into F-List.

- When a query (w, c_1) is made to G, if there exists a pair $((w, c_1), g) \in G$-List then we respond g. Otherwise, if $c_1 = c_1^*$ and $c_1^* = \mathcal{E}_{pk}(w; u^*)$, we respond $g = K^*$, set $\tilde{w} \leftarrow w$, and put (s, g) into G-List. Otherwise, we respond a random string g and put (s, g) into G-List.

- When a query s is made to H, if there exists a pair $(t, h) \in H$-List then we respond h. Otherwise, we respond a random string h and put (t, h) into H-List.

- When a decryption query (c_1, c_2) is made, if $(c_1, c_2) = (c_1^*, c_2^*)$ we reject the query since it is a challenge ciphertext. Otherwise, we search $((m, r), s) \in F$-List and $(s, h) \in H$-List such that $\mathcal{O}^{\mathrm{pca}}((s||(r \oplus h)), c_1) = 1$.
  - If there exist such pairs, we compute $K = G(s||(r \oplus h), c_1)$, $m \leftarrow \mathtt{D}_K(c_2)$, and check whether $s = F(m, r)$. If the equality holds, we respond m. Otherwise we reject the query.
  - When we cannot find such pairs, if there exists $((w, c_1), K) \in G$-List such that $\mathcal{O}^{\mathrm{pca}}(w, c_1) = 1$, we set $s||t \leftarrow w$ and compute $h \leftarrow H(s)$, $m \leftarrow \mathtt{D}_K(c_2)$, and check whether $s = F(m, t \oplus h)$. If the equality holds, we respond m. Otherwise we reject the query.

Above, the simulation of the random oracles F and G are different as those in the proof of the IND-CCA security of GEM in [5]. We can simulate these oracles without the plaintext checking oracle by using the randomness used to compute the challenge ciphertext.

The rest of the proof is almost the same as that of GEM in [5]. That is, we can analyze the soundness and the success probability of B in a similar way as that in the proof of the IND-CCA security of GEM, and we have

$$\mathbf{Adv}_{\Pi, B}^{\mathrm{owr\text{-}pca}}(k) \geq \frac{\epsilon - \nu}{2} - \frac{q_f}{2^\ell} - q_d \left(\frac{1}{2^k} + q_f \left(\frac{1}{2^\ell} + \frac{1}{2^k} \right) + \nu + \frac{1}{2^n} \right)$$

where $\epsilon = \mathbf{Adv}_{\Pi', A}^{\mathrm{owr\text{-}cca}}(k)$ and $\nu = \mathbf{Adv}_{\mathsf{SE}, C}^{\mathsf{ind}}(k)$.

Finally, we estimate the total number of calls to $\mathcal{O}_{\mathrm{pca}}$ and the running time of B, which are different from those in the proof of IND-CCA security of GEM. In our reduction, we use the plaintext checking oracle only in the simulation of the decryption oracle, and in this simulation, we make at most $q_d(q_f + q_g)$ calls since in the worst case we have to call $\mathcal{O}_{\mathrm{pca}}$ for all elements in F-List and for all elements in G-List. Therefore, the total number of calls to $\mathcal{O}_{\mathrm{pca}}$ is bounded by $q_{\mathrm{pca}} \leq q_d(q_f + q_g)$. From this, we can easily see that the running time of B is bounded by $\tau' \leq \tau + (q_d(q_f + q_g) + q_h)(\tau_{\mathrm{pca}} + O(1))$. $\qquad \square$

5 3-Round OAEP and its Security

In this section, we review 3-round OAEP proposed by Phan and Pointcheval [14], and prove that 3-round OAEP with a partial one-way trap-door permutation provides OWR-CCA in the random oracle model.

5.1 3-Round OAEP

In this section, we review 3-round OAEP by Phan and Pointcheval [14]. 3-round OAEP is a novel construction in order to remove redundancy of (2-round) OAEP. Unlike OAEP, all of the ciphertexts of 3-round OAEP are valid (which means the encryption function is not only a probabilistic injection, but also a surjection), and 3-round OAEP does not have the usual redundancy $m||0^k$ which OAEP has. In [14], they proposed 3-round OAEP with *any* trap-door *bijection*. Thus, the domain and the range of the function may be different. We modify their definition to define 3-round OAEP with a trap-door *permutation*.

Definition 8 (3-round OAEP with a trap-door permutation). *3-round OAEP* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *with a family of trap-door permutations* $\mathcal{TP} = (K, \varphi, \psi)$ *is as follows. The key generation algorithm* $\mathcal{K}$ *takes a security parameters* 1^k *and* 1^ℓ, *runs the key generation algorithm* $K(1^k, 1^\ell)$ *of* $\mathcal{TP}$, *and gets a pair of public and secret keys* (pk, sk) *for* $\mathcal{TP}$, *where* $\text{Dom}_{\mathcal{TP}}(\text{pk}) = \{0,1\}^{k+\ell}$. *Then, the key generation algorithm* $\mathcal{K}$ *returns a public key* $pk = (\text{pk}, 1^k, 1^\ell)$ *and a corresponding secret key* $sk = \text{sk}$. *For any* pk, *the plaintext space* $\text{MSPC}_\Pi(pk)$ *and the randomness space* $\text{RSPC}_\Pi(pk)$ *are* $\{0,1\}^\ell$ *and* $\{0,1\}^k$, *respectively. The encryption and decryption algorithms are as follows. Note that* $F : \{0,1\}^k \to \{0,1\}^\ell$, $G : \{0,1\}^\ell \to \{0,1\}^k$, *and* $H : \{0,1\}^k \to \{0,1\}^\ell$ *are hash functions.*

```
Algorithm ℰ_pk(m;r)        Algorithm 𝒟_sk(c)
    r ←ᴿ {0,1}^k               t||u ← ψ_sk(c) where |t| = k and |u| = ℓ
    s ← m ⊕ F(r)               s ← u ⊕ H(t)
    t ← r ⊕ G(s)               r ← t ⊕ G(s)
    u ← s ⊕ H(t)               m ← s ⊕ F(r)
    c ← φ_pk(t||u)             return m
    return c
```

Phan and Pointcheval proved that 3-round OAEP with a family of trap-door permutations $\mathcal{TP}$ is secure in the sense of IND-CCA2 if $\mathcal{TP}$ is θ-partial one-way where $\theta = k/(k+\ell)$.

5.2 OWR-CCA Security of 3-Round OAEP with a Trap-Door Permutation

In this section, we prove that 3-round OAEP with a partial one-way trap-door permutation provides OWR-CCA in the random oracle model.

In the proof of IND-CCA2 by Phan and Pointcheval, they showed that the probability that the adversary asks r^* is negligible. Then, the adversary cannot get any information about the message m, since the value $F(r^*)$ cannot be guessed by the adversary. However, in our proof, the randomness r^* is revealed to the adversary. Thus, we take different strategy in the proof of the following theorem.

Theorem 2. *For any adversary* A *attacking the OWR-CCA security of 3-round OAEP* $\diamond$ *with* $\mathcal{TP}$, *and making at most* q_d *queries to the decryption oracle,* q_f *F-oracle queries,* q_g *G-oracle queries, and* q_h *H-oracle queries, there exists a* θ-*partial inverting adversary* B *for* $\mathcal{TP}$, *such that for any* k, ℓ, *and* $\theta = \frac{k}{k+\ell}$,

$$\mathbf{Adv}_{\Pi,A}^{\text{owr-cca}}(k) \leq \frac{q_d(5q_g + 3q_d + q_f + 1) + q_g(q_f + 1)}{2^\ell} + \frac{2q_d(3q_f + 2q_d)}{2^k} + 2q_h \cdot \mathbf{Adv}_{\mathcal{TP},B}^{\theta\text{-pow}}(k)$$

and the running time of B *is that of* A *plus* $q_g \cdot q_h \cdot T_\varphi + q_d \cdot T_{lu}$ *where* T_φ *denotes the time for evaluating the permutation* φ_{pk} *and* T_{lu} *denote the time for looking up in a list.*

Proof. We define a sequence Game_1, Game_2, etc., of modified attack games starting from the actual game Game_0. Each of the games operates on the same underlying probability space: the public

and secret keys of the cryptosystems, the coin tosses of the adversary A, the random oracles F, G, and H, and the message and the randomness for the challenge ciphertext.

In the following, all variables with asterisk refer to the challenge ciphertext, and all variables with no asterisk refer to the decryption queries.

Game_0. Game_0 is the same as the actual OWR-CCA game. A pair of keys (pk, sk) is generated by $\mathcal{K}(1^k, 1^\ell)$, and the CCA-adversary A takes $pk = (\mathsf{pk}, 1^k, 1^\ell)$, the challenge ciphertext c^*, and the randomness r^* where

$$m \stackrel{R}{\leftarrow} \{0,1\}^\ell, \ r^* \stackrel{R}{\leftarrow} \{0,1\}^k, \ s^* = m \oplus F(r^*), \ t^* = r^* \oplus G(s^*), \ u^* = s^* \oplus H(t^*), \ c^* = \varphi_{\mathsf{pk}}(t^* \| u^*),$$

and A outputs m'. In the above experiment, the adversary A can make access to the random oracles F, G, H, and the decryption oracle $\mathcal{D}_{sk}$. However, A cannot ask the challenge ciphertext c^* to the decryption oracle.

We denote by S_0 the event "$m' = m$" and use a similar notation S_i in each Game_i below. By definition, we have $\Pr[S_0] = \mathbf{Adv}_{\Pi,A}^{\mathrm{owr\text{-}cca}}(k)$.

Remark 1. *In the proof of IND-CCA2 by Phan and Pointcheval, they modified the game by setting r^* independently of anything else, as well as $F(r^*)$. Furthermore, they bounded the probability that the adversary asks r^* by negligible probability in the following games.*

However, we cannot use this strategy, since the adversary knows r^* and we cannot bound the probability that the adversary asks r^*. Thus, we take different strategy in the following games.

Game_1. We choose two random values $s^+ \stackrel{R}{\leftarrow} \{0,1\}^\ell$ and $g^+ \stackrel{R}{\leftarrow} \{0,1\}^k$ in advance (i.e. before the adversary A runs), and use s^+ and g^+ instead of s^* and $G(s^*)$, respectively. In Game_1, we apply the following special rules.

Note that we set s^* before choosing m but s^* and m are still identically distributed as those in Game_0.

R1: We compute the challenge ciphertext c^* by setting

$$s^* \leftarrow s^+, \ \ m \leftarrow s^* \oplus F(r^*), \ \ t^* \leftarrow r^* \oplus g^+.$$

R2: Whenever the random oracle G is queried at s^*, the answer is g^+.

Since we replace $(s^*, G(s^*), m)$ by a different, but identically distributed (by the definition of the random oracle G) random variables, we have $\Pr[S_1] = \Pr[S_0]$.

Game_2. In this game, we drop the rule **R2** from Game_1. Therefore, g^+ is just used for computing the challenge ciphertext, and if s^* is queried to G then we respond not g^+ but $G(s^*)$ by using the random oracle G.

Game_1 and Game_2 may differ if s^* is queried to G. Let AskG_2 denotes the event that, in Game_2, s^* is queried to G (except by the encryption oracle, for producing the challenge). We use an identical notation AskG_i for each Game_i below. Then, $|\Pr[S_2] - \Pr[S_1]| \le \Pr[\mathsf{AskG}_2]$.

Game_3. We now set t^* independently of anything else, as well as $H(t^*)$. We choose two random values $t^+ \stackrel{R}{\leftarrow} \{0,1\}^k$ and $h^+ \stackrel{R}{\leftarrow} \{0,1\}^\ell$ in advance (i.e. before the adversary A runs), and use t^+ instead of t^*, as well as h^+ instead of $H(t^*)$. In Game_3, we apply the following special rules. Note that we change the way to compute g^+ (but still identically distributed as that in Game_2).

R1′: We compute the challenge ciphertext c^* by setting

$$t^* \leftarrow t^+, \ \ g^+ \leftarrow t^+ \oplus r^*, \ \ u^* \leftarrow s^* \oplus h^+.$$

9

R2′: Whenever the random oracle H is queried at t^+, the answer is h^+.

Since we replace the set of elements $(t^*, H(s^*), g^+)$ by a different, but identically distributed (by the definition of the random oracle H) set of random variables, we have $\Pr[S_3] = \Pr[S_2]$ and $\Pr[\mathsf{AskG}_3] = \Pr[\mathsf{AskG}_2]$.

Game_4. In this game, we drop the rule **R2′** from Game_3. Therefore, h^+ is just used for computing the challenge ciphertext, and if t^* is queried to H then we respond not h^+ but $H(t^*)$ by using the random oracle H.

Game_3 and Game_4 may differ if t^* is queried to H. Let AskH_4 denotes the event that, in Game_4, t^* is queried to H (except by the encryption oracle, for producing the challenge). We use an identical notation AskH_i for each Game_i below. Then, $|\Pr[S_4] - \Pr[S_3]| \leq \Pr[\mathsf{AskH}_4]$ and $|\Pr[\mathsf{AskG}_4] - \Pr[\mathsf{AskG}_3]| \leq \Pr[\mathsf{AskH}_4]$.

Game_5. In this game, we randomly choose $u^+ \overset{R}{\leftarrow} \{0,1\}^\ell$ and simply set $u^* \leftarrow u^+$, instead of $s^* \oplus h^+$. Since h^+ is uniformly distributed over $\{0,1\}^\ell$ and not revealed to the adversary, the distribution of u^+ and that of $u^* = s^* \oplus h^+$ in Game_4 are identical. Therefore, $\Pr[S_5] = \Pr[S_4]$, $\Pr[\mathsf{AskG}_5] = \Pr[\mathsf{AskG}_4]$, and $\Pr[\mathsf{AskH}_5] = \Pr[\mathsf{AskH}_4]$.

The challenge ciphertext c^* in this game is computed by

$$r^* \overset{R}{\leftarrow} \{0,1\}^k, \quad s^+ \overset{R}{\leftarrow} \{0,1\}^\ell, \quad s^* \leftarrow s^+, \quad m \leftarrow s^* \oplus F(r^*)$$

and

$$t^+ \overset{R}{\leftarrow} \{0,1\}^k, \quad u^+ \overset{R}{\leftarrow} \{0,1\}^\ell, \quad t^* \leftarrow t^+, \quad u^* \leftarrow u^+, \quad c^* \leftarrow \varphi_{\mathsf{pk}}(t^* \| u^*).$$

Here, s^+ is never revealed to the adversary and the distribution of s^* is independent of the adversary's view. Therefore, $\Pr[\mathsf{AskG}_5] \leq (q_g + q_d)/2^\ell$.

Furthermore, the distribution of $m = s^* \oplus F(r^*)$ is also independent of the adversary's view (even if the adversary knows the values r^* and $F(r^*)(= m \oplus s^*)$). Thus, the best way for the adversary to guess the plaintext m of c^* is encrypting some message $\hat{m} \in \{0,1\}^\ell$ with the randomness r^* and checking whether $c^* = \mathcal{E}_{pk}(\hat{m}; r^*)$ or not. In order to check this equality, the adversary has to know the hash values $G(\hat{s})$ and $H(\hat{t})$ where $\hat{s} = F(r^*) \oplus \hat{m}$ and $\hat{t} = r^* \oplus G(\hat{s})$. Since the adversary and the decryption oracle make at most $(q_g + q_d)$ G-oracle queries and $(q_h + q_d)$ H-oracle queries, the adversary can check the above equality at most $(q_h + q_d)(q_g + q_d)$ times. Thus, the probability $\Pr[S_5]$ is bounded by $(q_g + q_d)(q_f + q_d)/2^\ell$. In the following, we modify this game in order to evaluate $\Pr[\mathsf{AskH}_5]$.

Game_6. In Game_5, when manufacturing the challenge ciphertext, we randomly choose $c^+ \overset{R}{\leftarrow} \{0,1\}^{k+\ell}$, and simply set $c^* \leftarrow c^+$, ignoring the encryption oracle altogether.

Since φ_{pk} is a permutation over $\{0,1\}^{k+\ell}$, and $t^* = t^+$ and $u^* = u^+$ are uniformly distributed over $\{0,1\}^k$ and $\{0,1\}^\ell$, respectively, the distribution of $c^* = \varphi_{\mathsf{pk}}(t^* \| u^*)$ is the same as that of c^+. Thus, we have $\Pr[\mathsf{AskH}_6] = \Pr[\mathsf{AskH}_5]$.

Game_7. In this game, we simulate the random oracles F, G, H. We use three lists, F-List, G-List, H-List for simulating the random oracles F, G, and H, respectively. They are initially set to empty lists.

- When the adversary, the decryption oracle, or the encryption oracle (which makes a challenge ciphertext) makes a query $r \in \{0,1\}^k$ to F, if there exists a pair $(r, f) \in F$-List then we respond f. Otherwise, we respond a random string $f \overset{R}{\leftarrow} \{0,1\}^\ell$ and put (r, f) into F-List.

- When the adversary or the decryption oracle makes a query $s \in \{0,1\}^\ell$ to G, if there exists a pair $(s,g) \in G$-List then we respond g. Otherwise, we respond a random string $g \xleftarrow{R} \{0,1\}^k$ and put (s,g) into G-List.

- When the adversary or the decryption oracle makes a query $t \in \{0,1\}^k$ to H, if there exists a pair $(t,h) \in H$-List then we respond h. Otherwise, we respond a random string $h \xleftarrow{R} \{0,1\}^\ell$ and put (t,h) into H-List.

Since we can simulate these oracles perfectly, we have $\Pr[\mathsf{AskH}_7] = \Pr[\mathsf{AskH}_6]$.

Game_8. We now simulate the decryption oracle by using D-List, which is initially set to empty. For a decryption query c, if there exists a pair $(m,c) \in D$-List then we respond m as the plaintext. Otherwise, we simulate the decryption oracle as follows and put (m,c) into D-List:

D-1. We look up for $(t,h) \in H$-List and $(s,g) \in G$-List such that $\varphi_{\mathsf{pk}}(t, s \oplus h) = c$. If the record is found, we define $u \leftarrow s \oplus h$. Otherwise, we set $t \leftarrow \perp$ and $u \leftarrow \perp$.

D-2. If $(t,h) \in H$-List and $(s,g) \in G$-List (i.e. the first case in **D-1**), then we compute

$$r \leftarrow t \oplus g, \ f \leftarrow F(r), \ \text{and} \ m \leftarrow s \oplus f,$$

and returns m. Otherwise (i.e. the second case in **D-1**), we choose $m \xleftarrow{R} \{0,1\}^\ell$ and returns m.

In addition, we modify the simulation of the random oracle G as follows:

Whenever a pair (s,g) is added to G-List, the following process is executed: for any $(t,h) \in H$-List and any $(m,c) \in D$-List such that $c = \varphi_{\mathsf{pk}}(t\|(h \oplus s))$, we compute $r \leftarrow t \oplus g$, $f \leftarrow m \oplus s$, and add (r,f) to F-List.

The simulations of the decryption oracle and the random oracle G described above are the same as those in the proof of IND-CCA2 by Phan and Pointcheval. We prove the following lemma in Appendix A by a similar way as that in the proof of IND-CCA2 by Phan and Pointcheval.

Lemma 1.

$$\Pr[\mathsf{AskH}_7] \leq q_d(2q_g + q_d)/2^\ell + q_d(3q_f + 2q_d)/2^k + \Pr[\mathsf{AskH}_8],$$

and the running time for simulating the decryption oracle, as well as the random oracles is bounded by $q_g \cdot q_h \cdot T_\varphi + q_d \cdot T_{lu}$ where T_φ denotes the time for evaluating the permutation φ_{pk} and T_{lu} denotes the time for looking up in a list.

We now construct an algorithm B attacking θ-partial one-wayness of $\mathcal{TP}$, where $\theta = k/(k+\ell)$, by using A against Game_8.

1. B takes pk and y^* where $x^* \xleftarrow{R} \{0,1\}^{k+\ell}$ and $y^* = \varphi_{\mathsf{pk}}(x^*)$. Then, B runs A against Game_8 where $c^* \leftarrow y^*$ and $r^* \leftarrow \{0,1\}^\ell$.

2. When A terminates, B outputs $t \xleftarrow{R} \{t'|(t',h') \in H\text{-List}\}$.

Note that B simulates the random oracles and the decryption oracle for A as in Game_8 (by using F-List, G-List, H-List, and D-List). If AskH_8 occurs then there exists an element $t^* \in \{0,1\}^k$ in H-List such that $\varphi_{\mathsf{pk}}(t^*\|u') = y^*$ for some $u' \in \{0,1\}^\ell$. Therefore, we have $\Pr[\mathsf{AskH}_8] \leq q_h \cdot \mathbf{Adv}_{\mathcal{TP},B}^{\theta\text{-pow}}(k)$.

In conclusion, we have

$$\mathbf{Adv}^{\text{owr-cca}}_{\Pi,A}(k)$$
$$= \Pr[S_0] = \Pr[S_1] \leq \Pr[S_2] + \Pr[\mathsf{AskG}_2] = \Pr[S_3] + \Pr[\mathsf{AskG}_3]$$
$$\leq \Pr[S_4] + \Pr[\mathsf{AskH}_4] + \Pr[\mathsf{AskG}_4] + \Pr[\mathsf{AskH}_4]$$
$$= \Pr[S_5] + \Pr[\mathsf{AskG}_5] + 2 \cdot \Pr[\mathsf{AskH}_5]$$
$$\leq (q_g + q_d)(q_f + q_d)/2^\ell + (q_g + q_d)/2^\ell + 2 \cdot \Pr[\mathsf{AskH}_5]$$
$$= (q_g + q_d)(q_f + q_d)/2^\ell + (q_g + q_d)/2^\ell + 2 \cdot \Pr[\mathsf{AskH}_6]$$
$$= (q_g + q_d)(q_f + q_d)/2^\ell + (q_g + q_d)/2^\ell + 2 \cdot \Pr[\mathsf{AskH}_7]$$
$$\leq (q_g + q_d)(q_f + q_d)/2^\ell + (q_g + q_d)/2^\ell + 2 \cdot (q_d(2q_g + q_d)/2^\ell + q_d(3q_f + 2q_d)/2^k + \Pr[\mathsf{AskH}_8])$$
$$\leq \frac{q_d(5q_g + 3q_d + q_f + 1) + q_g(q_f + 1)}{2^\ell} + \frac{2q_d(3q_f + 2q_d)}{2^k} + 2q_h \cdot \mathbf{Adv}^{\theta\text{-pow}}_{\mathcal{TP},B}(k).$$

We now estimate the running time of B. From Lemma 1, the running time for simulating the decryption oracle, as well as the random oracles is bounded by $q_g \cdot q_h \cdot T_\varphi + q_d \cdot T_{lu}$ where T_φ denotes the time for evaluating the permutation φ_{pk} and T_{lu} denotes the time for looking up in a list. Therefore, the running time of B is bounded by that of A plus $q_g \cdot q_h \cdot T_\varphi + q_d \cdot T_{lu}$. $\qquad\square$

6 Concluding Remarks

In this paper, we have considered the security of public-key encryption in the scenario that the adversary knows the randomness which was used to compute the ciphertext. We have formalized the security notion for public-key encryption in the situation that the randomness is revealed as "the one-wayness with the randomness revealed." We have also shown that GEM satisfies OWR-CCA if the underlying scheme satisfies OWR-PCA, and 3-round OAEP with a trap-door permutation satisfies OWR-CCA if the trap-door permutation is partial one-way.

It might be interesting to consider whether other previously proposed encryption schemes are secure in the sense of OWR-CPA or OWR-CCA. In particular, OAEP or OAEP+ might satisfy OWR-CPA or OWR-CCA by modifying the assumption on the underlying trap-door permutation. For example, OAEP might satisfy OWR-CPA or OWR-CCA when the underlying trap-door permutation satisfies that no poly-time algorithm, given $c = \phi_{\mathsf{pk}}(x)$, can compute the most n significant bits of x, where n is the length of plaintexts of OAEP.

It might be also interesting to consider that other security notions for public-key encryption, the anonymity [1], for example, in the situation that the adversary knows the randomness. Furthermore, we can also consider the security notions with respect to the privacy of the randomness. For example, the adversary cannot find the randomness from the ciphertext, and the adversary cannot distinguish which randomness was used to compute the ciphertext.

References

[1] BELLARE, M., BOLDYREVA, A., DESAI, A., AND POINTCHEVAL, D. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology – ASIACRYPT 2001* (Gold Coast, Australia, December 2001), C. Boyd, Ed., vol. 2248 of *LNCS*, Springer-Verlag, pp. 566–582. Full version of this paper, available via `http://www-cse.ucsd.edu/users/mihir/`.

[2] BELLARE, M., AND ROGAWAY, P. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Advances in Cryptology – EUROCRYPT '94* (Perugia, Italy, May 1994), A. De Santis, Ed., vol. 950 of *LNCS*, Springer-Verlag, pp. 92–111.

[3] BOSLEY, C., AND DODIS, Y. Does Privacy Require True Randomness? To appear in *Theory of Cryptography Conference – TCC 2007* (Amsterdam, The Netherlands, February 2007). `http://eprint.iacr.org/2006/283`.

[4] CANETTI, R., AND KRAWCZYK, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology – EUROCRYPT 2001* (Innsbruck, Austria, May 2001), B. Pfitzmann, Ed., vol. 2045 of *LNCS*, Springer-Verlag, pp. 453–474.

[5] CORON, J.-S., HANDSCHUH, H., JOYE, M., PAILLIER, P., POINTCHEVAL, D., AND TYMEN, C. GEM: A Generic Chosen-Ciphertext Secure Encryption Method. In *Topics in Cryptology – CT-RSA 2002* (San Jose, CA, USA, February 2002), B. Preneel, Ed., vol. 2271 of *LNCS*, Springer-Verlag, pp. 263–276.

[6] CRAMER, R., AND SHOUP, V. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology – CRYPTO '98* (Santa Barbara, California, USA, August 1998), H. Krawczyk, Ed., vol. 1462 of *LNCS*, Springer-Verlag, pp. 13–25.

[7] CRAMER, R., AND SHOUP, V. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Advances in Cryptology – EUROCRYPT 2002* (Amsterdam, The Netherlands, April 2002), L. Knudsen, Ed., vol. 2332 of *LNCS*, Springer-Verlag, pp. 45–64.

[8] FUJISAKI, E., OKAMOTO, T., POINTCHEVAL, D., AND STERN, J. RSA-OAEP is Secure under the RSA Assumption. In Kilian [9], pp. 260–274.

[9] KILIAN, J., Ed. *Advances in Cryptology – CRYPTO 2001* (Santa Barbara, California, USA, August 2001), vol. 2139 of *LNCS*, Springer-Verlag.

[10] KRAWCZYK, H. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology – CRYPTO 2005* (Santa Barbara, California, USA, August 2005), V. Shoup, Ed., vol. 3621 of *LNCS*, Springer-Verlag, pp. 546–566.

[11] LAUTER, K., AND MITYAGIN, A. Security Analysis of KEA Authenticated Key Exchange Protocol. In *PKC 2006 – 9th International Workshop on Theory and Practice in Public Key Cryptography* (New York, USA, April 2006), M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds., vol. 3958 of *LNCS*, Springer-Verlag, pp. 378–394.

[12] OKAMOTO, T., AND POINTCHEVAL, D. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In *Topics in Cryptology – CT-RSA 2003* (San Francisco, CA, USA, April 2001), D. Naccache, Ed., vol. 2020 of *LNCS*, Springer-Verlag, pp. 159–175.

[13] PAILLIER, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology – EUROCRYPT '99* (Prague, Czech Republic, May 1999), J. Stern, Ed., vol. 1592 of *LNCS*, Springer-Verlag, pp. 223–238.

[14] PHAN, D. H., AND POINTCHEVAL, D. Chosen-Ciphertext Security without Redundancy. In *Advances in Cryptology – ASIACRYPT 2003* (Taipei, Taiwan, November 2003), C. S. Laih, Ed., vol. 2894 of *LNCS*, Springer-Verlag, pp. 1–18.

[15] POINTCHEVAL, D. Chosen-Ciphertext Security for Any One-Way Cryptosystem. In *PKC 2000 – 3rd International Workshop on Theory and Practice in Public Key Cryptography* (Melbourne, Victoria, Australia, January 2000), H. Imai and Y. Zheng, Eds., vol. 1751 of *LNCS*, Springer-Verlag, pp. 129–146.

[16] SHOUP, V. OAEP Reconsidered. In Kilian [9], pp. 239–259.

A Proof of Lemma 1

The following proof is similar to that of IND-CCA2 by Phan and Pointcheval [14]. We define a sequence $\mathsf{Game}_{7.1}$, $\mathsf{Game}_{7.2}$, etc., of modified attack games starting from the game Game_7.

$\mathsf{Game}_{7.1}$. In this game, we simulate the decryption oracle by using D-List which is initially set to empty.

- When the adversary makes a query c to the decryption oracle, if there exists a pair $(m, c) \in D$-List then we respond m as the plaintext. Otherwise, we compute the plaintext as follows and put (m, c) into D-List.

    D-1. We first compute $t \| u \leftarrow \psi_{\mathsf{sk}}(c)$.

    D-2. We next compute

    $$h \leftarrow H(t), \ s \leftarrow u \oplus h, \ g \leftarrow G(s), \ r \leftarrow t \oplus g, \ f \leftarrow F(r), \ m \leftarrow s \oplus f,$$

    and return m.

    Since we can simulate the decryption oracle perfectly, we have $\Pr[\mathsf{AskH}_{7.1}] = \Pr[\mathsf{AskH}_7]$.

In the following, we modify the simulation **D-2** of the decryption oracle. We consider three cases with respect to the decryption query c by the adversary. Note that we can distinguish between the following three cases after computing $t \| u \leftarrow \psi_{\mathsf{sk}}(c)$ in the simulation **D-1** of the decryption oracle.

D-2-noT. $(t, h) \notin H$-List, that is, the value t has not been queried to the oracle H.

D-2-TnoS. $(t, h) \in H$-List and $(s, g) \notin G$-List where $s = u \oplus h$, that is, the value t has been already queried to the oracle H, but the value $s = u \oplus h$ has not been queried to the oracle G.

D-2-TS. $(t, h) \in H$-List and $(s, g) \in G$-List, that is, the values t and $s = u \oplus h$ have been already queried to the oracles H and G, respectively.

In the following, we modify the above decryption process of each case.

$\mathsf{Game}_{7.2}$. In this game, we modify the simulation of the decryption oracle in the case **D-2-noT**. That is, we respond the decryption query in the case **D-2-noT** as follows:

Set $h \xleftarrow{R} \{0,1\}^\ell$, $s \leftarrow u \oplus h$, $g \xleftarrow{R} \{0,1\}^k$, $r \leftarrow t \oplus g$, $f \leftarrow F(r)$, $m \leftarrow s \oplus f$.
Add (s, g) to G-List and (t, h) to H-List, and return m.

This makes difference only if s is already in G-List. Since t has not been queried to H in the case **D-2-noT**, $h = H(t)$ is uniformly distributed. Therefore, the probability that s has already been queried to G is $(q_g + q_d)/2^\ell$. Summing up for all decryption queries, we get $|\Pr[\mathsf{AskH}_{7.2}] - \Pr[\mathsf{AskH}_{7.1}]| \le q_d(q_g + q_d)/2^\ell$.

$\mathsf{Game}_{7.3}$. In this game, we modify again the simulation of the decryption oracle in the case **D-2-noT** by not querying the oracle F:

Set $h \xleftarrow{R} \{0,1\}^\ell$, $s \leftarrow u \oplus h$, $g \xleftarrow{R} \{0,1\}^k$, $r \leftarrow t \oplus g$, $f \xleftarrow{R} \{0,1\}^\ell$, $m \leftarrow s \oplus f$.
Add (r, f) to F-List, (s, g) to G-List, and (t, h) to H-List, and return m.

This makes difference only if r is already in F-List. Since g is uniformly distributed, the probability that r has already been queried to F is $(q_f + q_d)/2^k$. Summing up for all decryption queries, we get $|\Pr[\mathsf{AskH}_{7.3}] - \Pr[\mathsf{AskH}_{7.2}]| \le q_d(q_f + q_d)/2^k$.

Game$_{7.4}$. In this game, we modify again the simulation of the decryption oracle in the case **D-2-noT** by answering as follows:

> Set $m \stackrel{R}{\leftarrow} \{0,1\}^\ell$, $h \stackrel{R}{\leftarrow} \{0,1\}^\ell$, $s \leftarrow u \oplus h$, $g \stackrel{R}{\leftarrow} \{0,1\}^k$, $r \leftarrow t \oplus g$, $f \leftarrow m \oplus s$.
> Add (r, f) to F-List, (s, g) to G-List, and (t, h) to H-List, and return m.

Here, we first choose m and define f from m, while we first choose f and define m from f in Game$_{73}$. Two games Game$_{7.4}$ and Game$_{7.3}$ are perfectly indistinguishable and we have $\Pr[\mathsf{AskH}_{7.4}] = \Pr[\mathsf{AskH}_{7.3}]$.

Game$_{7.5}$. We now modify the simulation of the decryption oracle in the case **D-2-TnoS** by not calling either F or G as follows. In the case **D-2-TnoS**, the adversary asks $c = \varphi_{\mathsf{pk}}(t||u)$ such that $h = H(t)$ is known, but $s = u \oplus h$ has never been queried to G.

> Set $g \stackrel{R}{\leftarrow} \{0,1\}^k$, $r \leftarrow t \oplus g$, $f \stackrel{R}{\leftarrow} \{0,1\}^\ell$, $m \leftarrow s \oplus f$.
> Add (r, f) to F-List and (s, g) to G-List, and return m.

This makes difference only if r is already in F-List. Since g is uniformly distributed (s is not in G-List), the probability that r has already been queried to F is less than $(q_f + q_d)/2^k$. Summing up for all decryption queries, we get $|\Pr[\mathsf{AskH}_{7.5}] - \Pr[\mathsf{AskH}_{7.4}]| \leq q_d(q_f + q_d)/2^k$.

Game$_{7.6}$. We modify again the simulation of the decryption oracle in the case **D-2-TnoS** by answering as follows:

> Set $m \stackrel{R}{\leftarrow} \{0,1\}^\ell$, $g \stackrel{R}{\leftarrow} \{0,1\}^k$, $r \leftarrow t \oplus g$, $f \leftarrow m \oplus s$.
> Add (r, f) to F-List and (s, g) to G-List, and return m.

Here, we first choose m and define f from m, while we first choose f and define m from f in Game$_{7.5}$. Two games Game$_{7.6}$ and Game$_{7.5}$ are perfectly indistinguishable and we have $\Pr[\mathsf{AskH}_{7.6}] = \Pr[\mathsf{AskH}_{7.5}]$.

Game$_{7.7}$. In this game, we do not store either (s, g) in G-List or (r, f) in F-List, in the cases **D-2-noT** and **D-2-TnoS**. Instead of storing these elements, we add some process to the simulation of the random oracle G.

- In the case **D-2-noT**, the decryption oracle sets $h \stackrel{R}{\leftarrow} \{0,1\}^\ell$, $m \stackrel{R}{\leftarrow} \{0,1\}^\ell$, adds (t, h) to H-List, and returns m.
- In the case **D-2-TnoS**, the decryption oracle returns $m \stackrel{R}{\leftarrow} \{0,1\}^\ell$.
- In addition, whenever a pair (s, g) is added to G-List, the following process is executed: for any $(t, h) \in H$-List and any $(m, c) \in D$-List such that $c = \varphi_{\mathsf{pk}}(t||(h \oplus s))$, we compute $r \leftarrow t \oplus g$, $f \leftarrow m \oplus s$, and add (r, f) to F-List.

Game$_{7.7}$ and Game$_{7.6}$ are perfectly indistinguishable unless r is asked to F before s is asked to G. In fact, if r is asked after s, at the moment that s is asked, by the above simulation, we will find (t, h) and therefore (r, f) is computed and added to F-List as in the Game$_{7.6}$. Until s is asked to G, g is a uniform variable, so is r. Therefore, the probability that r has been asked to F is $q_f/2^k$. Summing up for all decryption queries, we get $|\Pr[\mathsf{AskH}_{7.7}] - \Pr[\mathsf{AskH}_{7.6}]| \leq q_d \cdot q_f/2^k$.

Game$_{7.8}$. In this game, we modify again the simulation of the decryption oracle in the case **D-2-noT**. We do not store (t, h) in H-List and just answer $m \leftarrow \{0,1\}^\ell$ in the case **D-2-noT**. In the previous game, for the query h to H, we answer randomly h, so the adversary in the two games Game$_{7.8}$ and Game$_{7.7}$ cannot distinguish the answers of a query to H. Nevertheless,

H-List has been changed and therefore, the answer for a query to F can be changed. We can see that the two games $\mathsf{Game}_{7.8}$ and $\mathsf{Game}_{7.7}$ are perfectly indistinguishable unless s is asked to G before t is asked to H. In fact, if s is asked to G after t is asked to H, at the moment that s is asked, by the above simulation, we will find (t, h) and therefore (r, f) is computed and added in F-List as in the $\mathsf{Game}_{7.7}$.

Until t is asked to H, h is a uniform variable, so is $s = u \oplus h$. Therefore, the probability that s has been asked to G is $q_g/2^\ell$. Summing up for all decryption queries, we get $|\Pr[\mathsf{AskH}_{7.8}] - \Pr[\mathsf{AskH}_{7.7}]| \le q_d \cdot q_g/2^\ell$.

$\mathsf{Game}_{7.9}$. We now complete the simulation of the decryption oracle by modifying the case **D-2-TS** and the step **D-1**. We do not ask any query to ψ_{sk} in **D-1**. Intuitively, if both t and s have been asked, we can easily find them, and can return m. Otherwise, we give a random answer as in the previous game.

- In **D-1**, for the decryption query c, we look up for $(t, h) \in H$-List and $(s, g) \in G$-List such that $\varphi_{\mathsf{pk}}(t, s \oplus h) = c$. If the record is found, we define $u \leftarrow s \oplus h$. Otherwise, we set $t \leftarrow \bot$ and $u \leftarrow \bot$.
- In the case **D-2-TS**, we compute $r \leftarrow t \oplus g$, $f \leftarrow F(r)$, and $m \leftarrow s \oplus f$, and return m.

The two games $\mathsf{Game}_{7.9}$ and $\mathsf{Game}_{7.8}$ are perfectly indistinguishable. In fact, in the first case nothing is modified, and in the second case by making $t \leftarrow \bot$ and $u \leftarrow \bot$, the answer of the decryption oracle for the query c will be a random m as in the previous game. Thus, $\Pr[\mathsf{AskH}_{7.9}] = \Pr[\mathsf{AskH}_{7.8}]$.

Since $\mathsf{Game}_{7.9}$ is identical to Game_8, we have

$$
\begin{aligned}
&\Pr[\mathsf{AskH}_7] \\
&= \Pr[\mathsf{AskH}_{7.1}] \\
&\le q_d(q_g + q_d)/2^\ell + \Pr[\mathsf{AskH}_{7.2}] \\
&\le q_d(q_g + q_d)/2^\ell + q_d(q_f + q_d)/2^k + \Pr[\mathsf{AskH}_{7.3}] \\
&= q_d(q_g + q_d)/2^\ell + q_d(q_f + q_d)/2^k + \Pr[\mathsf{AskH}_{7.4}] \\
&\le q_d(q_g + q_d)/2^\ell + 2q_d(q_f + q_d)/2^k + \Pr[\mathsf{AskH}_{7.5}] \\
&= q_d(q_g + q_d)/2^\ell + 2q_d(q_f + q_d)/2^k + \Pr[\mathsf{AskH}_{7.6}] \\
&\le q_d(q_g + q_d)/2^\ell + 2q_d(q_f + q_d)/2^k + q_d q_f/2^k + \Pr[\mathsf{AskH}_{7.7}] \\
&\le q_d(q_g + q_d)/2^\ell + 2q_d(q_f + q_d)/2^k + q_d q_f/2^k + q_d q_g/2^\ell + \Pr[\mathsf{AskH}_{7.8}] \\
&= q_d(q_g + q_d)/2^\ell + 2q_d(q_f + q_d)/2^k + q_d q_f/2^k + q_d q_g/2^\ell + \Pr[\mathsf{AskH}_{7.9}] \\
&= q_d(2q_g + q_d)/2^\ell + q_d(3q_f + 2q_d)/2^k + \Pr[\mathsf{AskH}_8].
\end{aligned}
$$

We now estimate the time for simulating the decryption oracle, as well as the random oracles. The running time for simulating these oracles is bounded by $q_g \cdot q_h \cdot T_\varphi + q_d \cdot T_{lu}$ where T_φ denotes the time for evaluating the permutation φ_{pk} and T_{lu} denotes the time for looking up in a list. We can indeed perform the simulation granted an additional list GH-List which contains all the tuples (t, h, s, g, c) where $(t, h) \in H$-List, $(s, g) \in G$-List, and $c = \varphi_{\mathsf{pk}}(t, s \oplus h)$.