Efficient Universal Padding Techniques for Multiplicative Trapdoor One-way Permutation

Yuichi Komano*[1] and Kazuo Ohta[2]

[1] TOSHIBA,
Komukai Toshiba-cho 1, Saiwai-ku, Kawasaki-shi, Kanagawa, Japan
yuichi1.komano@toshiba.co.jp
[2] The University of Electro-Communications,
Chofugaoka 1-5-1, Chofu-shi, Tokyo, Japan
ota@ice.uec.ac.jp

Abstract. Coron et al. proposed the ES-based scheme PSS-ES which realizes an encryption scheme and a signature scheme with a unique padding technique and key pair. The security of PSS-ES as an encryption scheme is based on the *partial-domain one-wayness* of the encryption permutation. In this paper, we propose new ES schemes OAEP-ES, OAEP++-ES, and REACT-ES, and prove their security under the assumption of *only* the *one-wayness* of encryption permutation. OAEP-ES, OAEP++-ES, and REACT-ES suit practical implementation because they use the same padding technique for encryption and for signature, and their security proof guarantees that we can prepare one key pair to realize encryption and signature in the same way as PSS-ES. Since *one-wayness* is a weaker assumption than *partial-domain one-wayness*, the proposed schemes offer tighter security than PSS-ES. Hence, we conclude that OAEP-ES, OAEP++-ES, and REACT-ES are more effective than PSS-ES. REACT-ES is the most practical approach in terms of the tightness of security and communication efficiency.

1 Introduction

Since the invention of the RSA encryption scheme [11], there have been a lot of interest in standardization and investigations into public key cryptosystems, in particular those for encryption and signature schemes. The encryption scheme OAEP (*Optimal Asymmetric Encryption Padding*, [2]) and the signature scheme PSS (*Probabilistic Signature Scheme*, [3]) are considered to be practical because they offer the strongest security level: IND-CCA2 (*indistinguishability against adaptive chosen ciphertext attack*) and EUF-ACMA (*existentially unforgeable against adaptive chosen message attack*).

OAEP first pads and then encrypts the plaintext while PSS pads and then signs the message; for encryption (signature), the trapdoor one-way permutation is applied in the direct (inverse) direction. Coron et al. [4] proposed the ES

* Work Performed at Major in Mathematical Science, Graduate School of Engineering and Science, Waseda University.

scheme (Encryption-Signature scheme[3]) PSS-ES, which is based on the message recovery signature scheme PSS-R [3], and proved its security. For encryption and signature, PSS-ES uses a shared padding scheme and key pair; the public key and the private key are chosen adequately for encryption and signing, respectively. Hence this scheme is useful in terms of implementation. The security proofs in [4], however, have some technical mistakes. Moreover, even if these mistakes are corrected, the fact that the security of PSS-ES as an encryption scheme is based on *partial-domain one-wayness* of the encryption permutation, decreases the reduction efficiency; it must use long keys to achieve adequate security.

This paper gives the exact security of PSS-ES by correcting the problems in [4]. Moreover, this paper introduces new ES schemes, OAEP-ES and REACT-ES, that are based on OAEP+ [12] and REACT [10], respectively[4]. The proposed schemes satisfy IND-CCA2&ACMA (*Indistinguishability against adaptive chosen ciphertext attack and adaptive chosen message attack*) as an encryption scheme and EUF-CCA2&ACMA (*Existentially unforgeable against adaptive chosen ciphertext attack and adaptive chosen message attack*) as a signature scheme under the assumption of *only* the *one-wayness* of the permutation, while PSS-ES relies upon the *partial-domain one-wayness* of the encryption permutation for its security as an encryption scheme.

The rest of this paper is organized as follows. Section 2 recalls the definitions of the ES scheme and its security notations. Section 3 proposes new ES schemes, OAEP-ES and REACT-ES, and gives their security. In section 4, we point out the problems of original security proof of PSS-ES, given by Coron et al. [4], and give its exact security. In sections 5 and 6, we compare reduction efficiency of proposed schemes with the one of PSS-ES following the estimation of Nakashima and Okamoto [9] and discuss the reason why our schemes are more practical than PSS-ES. Furthermore, Appendices A and B present the security proofs of REACT-ES.

As a result, OAEP-ES, OAEP++-ES, and REACT-ES can realize secure encryption-signature scheme (ES scheme) with a unique padding technique and key pair; their reduction efficiency are much better than those of PSS-ES. Due to the high reduction efficiency of its security proof and its improved communication efficiency, REACT-ES is the most practical approach.

2 Definitions

2.1 ES scheme with Universal Padding Technique

We describe a model of the ES scheme[3] (*Encryption-Signature scheme*) and its security. Since the ES scheme realizes an encryption scheme and a signature

[3] The ES scheme differs from *signcryption* [14]; the ES scheme realizes both encryption and signature schemes with a common padding technique and key pair (*encrypt or sign*), while signcryption realizes *encrypt then sign* or *sign then encrypt* scheme.

[4] We can construct another ES scheme, OAEP++-ES, based on OAEP++ [7]. In this paper, we will omit the detail of OAEP++-ES.

scheme with a common padding technique and key pair, we introduce attack model CCA2&ACMA following [4], where adversary $\mathcal{A}$ (forger $\mathcal{F}$) can freely use both decryption oracle D and signing oracle Σ. We extend notions of security IND-CCA2 [1] and EUF-ACMA [6] to create IND-CCA2&ACMA (*Indistinguishability against adaptive chosen ciphertext attack and adaptive chosen message attack*) and EUF-CCA2&ACMA (*Existentially unforgeable against adaptive chosen ciphertext attack and adaptive chosen message attack*), respectively.

Definition 1 (ES scheme with a unique padding technique). *If μ is a padding technique, then the ES scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$ with μ is defined as follows:*

- *Key generation algorithm $\mathcal{K}$ is probabilistic algorithm which, given security parameter k, outputs the pair of public and private keys, $\mathcal{K}(1^k) = (\mathsf{pk}, \mathsf{sk})$. We regard pk as f and sk as f^{-1}, hereafter.*
- *Encryption algorithm $\mathcal{E}$ takes plaintext x and public key pk, calculates $z = \mu(x, r)$ with some random integer r, and returns ciphertext[5] $y = f(z) = \mathcal{E}_{\mathsf{pk}}(x)$. This algorithm is probabilistic[6].*
- *Decryption algorithm $\mathcal{D}$ takes ciphertext y and private key sk, calculates $z = f^{-1}(y)$ and $\mu^{-1}(z) = x||r$ (un-padding), and returns plaintext $x = \mathcal{D}_{\mathsf{sk}}(y)$ if y is a valid ciphertext. Otherwise $\mathcal{D}$ returns Reject. This algorithm is deterministic.*
- *Signing algorithm $\mathcal{S}$ takes message x and private key sk, calculates $z = \mu(x, r)$ with some random integer r, and returns signature[5] $\sigma = f^{-1}(z) = \mathcal{S}_{\mathsf{sk}}(x)$. This algorithm is probabilistic.*
- *Verification algorithm $\mathcal{V}$ takes signature σ and public key pk, calculates $z = f(\sigma)$ and $\mu^{-1}(z) = x||r$ (un-padding), and returns message $x = \mathcal{V}_{\mathsf{pk}}(\sigma)$ if σ is a valid signature. Otherwise $\mathcal{V}$ returns Reject. This algorithm is deterministic.*

We denote the ES scheme for encryption and for signature by ES(E) and ES(S), respectively (*e.g.*, OAEP-ES(E) and OAEP-ES(S) mean the OAEP-ES using in an encryption and a signature, respectively).

Definition 2 (IND-CCA2&ACMA). *Let $\mathcal{A}$ be an adversary of the encryption scheme. The attack scenario is described as follows:*

1. *$\mathcal{A}$ receives public key pk with $\mathcal{K}(1^k) = (\mathsf{pk}, \mathsf{sk})$.*
2. *$\mathcal{A}$ submits decryption queries for ciphertext y of his choice to decryption oracle D and gets corresponding plaintext x. Moreover, $\mathcal{A}$ submits signing queries for message x' of his choice to signing oracle Σ and gets corresponding signature σ.*

[5] The input of f may be a part of z, *i.e.*, we allow to regard $y = f(z_1)||z_2$ ($\sigma = f^{-1}(z_1)||z_2$) as the ciphertext (signature) for $z = z_1||z_2$.

[6] Since padding technique μ is probabilistic, encryption permutation f may be deterministic(*e.g.,* RSA).

3. *$\mathcal{A}$ generates two plaintexts x_0, x_1 of identical length, and sends them to encryption oracle E as a challenge.*
4. *E chooses $b \stackrel{R}{\leftarrow} \{0,1\}$ and returns $y^* = \mathcal{E}_{\mathsf{pk}}(x_b)$ to $\mathcal{A}$ as a target ciphertext.*
5. *$\mathcal{A}$ continues to submit decryption queries for ciphertext y of his choice to D and gets corresponding plaintext x. Moreover, $\mathcal{A}$ continues to submit signing queries for message x' of his choice to Σ and gets corresponding signature σ. In this phase, the only restriction is that $\mathcal{A}$ cannot issue a query for y^* to D.*
6. *$\mathcal{A}$ guesses b in this attack and outputs $\hat{b}$.*

The adversary's advantage is defined as $\mathsf{Adv}(\mathcal{A}) = |2\Pr[b = \hat{b}] - 1|$. We say that the encryption scheme is $(t, q_D, q_\Sigma, q_H, \epsilon)$-secure in the sense of IND-CCA2&ACMA *if an arbitrary adversary[7], whose running time is bounded by t, cannot achieve an advantage more than ϵ after making at most q_D decryption queries, q_Σ signing queries, and q_H hash queries.*

Definition 3 (EUF-CCA2&ACMA). *Let $\mathcal{F}$ be a forger of the signature scheme. The attack scenario is described as follows:*

1. *$\mathcal{F}$ receives public key pk with $\mathcal{K}(1^k) = (\mathsf{pk}, \mathsf{sk})$.*
2. *$\mathcal{F}$ submits signing queries for message x of his choice to signing oracle Σ and gets corresponding signature σ. Moreover, $\mathcal{F}$ submits decryption queries for ciphertext y' of his choice to decryption oracle D and gets corresponding plaintext x'.*
3. *$\mathcal{F}$ outputs forgery σ^* with $\mathcal{V}_{\mathsf{pk}}(\sigma^*) = x^*$ for some x^* ($x^* \neq x$ for any signing query x).*

The forger's success probability is defined as $\epsilon = \Pr[\mathcal{V}_{\mathsf{pk}}(\sigma^*) = x^*]$. We say that the signature scheme is $(t, q_D, q_\Sigma, q_H, \epsilon)$-secure in the sense of EUF-CCA2&ACMA *if an arbitrary forger[7],whose running time is bounded by t, cannot achieve a success probability more than ϵ after making at most q_D decryption queries, q_Σ signing queries, and q_H hash queries.*

Note that the security proof of the ES scheme with a unique padding technique comes in two parts, first as an encryption scheme and then as a signature scheme.

2.2 Assumption of One-way Permutation

We classify trapdoor one-way permutations according to the difficulty of inverting them as follows [5]:

[7] We restrict the adversary (forger) by upper bounding the running time and the number of decryption, signing, and hash queries. We denote that $\mathcal{A}$ ($\mathcal{F}$) breaks an encryption scheme (signature scheme) in $(t, q_D, q_\Sigma, q_H, \epsilon)$ if $\mathcal{A}$ can distinguish b ($\mathcal{F}$ can outputs a forgery) within the time bound t and the advantage (success probability) more than ϵ using, at most, q_D decryption, q_Σ signing, and q_H hash queries.

Definition 4. *Let* $f : \{0,1\}^{k_0} \times \{0,1\}^{k_1} \to \{0,1\}^{k_0} \times \{0,1\}^{k_1}$ *be a permutation. We say that*

— f *is* (τ, ϵ)*-one-way, if an arbitrary adversary whose running time is bounded by* τ *has success probability* $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ *that does not exceed* ϵ*. Here,* $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = (s,t)]$.

— f *is* (τ, ϵ)*-partial-domain one-way, if an arbitrary adversary whose running time is bounded by* τ *has success probability* $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ *that does not exceed* ϵ*. Here,* $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A}) = \Pr_{s,t}[\mathcal{A}(f(s,t)) = s]$.

Moreover, we define $\mathsf{Succ}^{\mathsf{ow}}(\tau) = \max_{\mathcal{A}} \mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ *and* $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau) = \max_{\mathcal{A}} \mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$*, for all* $\mathcal{A}$*, whose running time is bounded by* τ*.*

By the above definition, we have $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geqq \mathsf{Succ}^{\mathsf{ow}}(\tau)$ for any τ. This inequality means that *partial-domain one-wayness* is a stronger assumption than *one-wayness*.

Through this paper, we assume that permutation f is multiplicative[8]. The multiplicative property of the permutation is described below.

Definition 5. *If* f *is a function, we call it a multicative function if*

$$f(ab) = f(a)f(b)$$

for arbitrary a *and* b*.*

3 Proposal Schemes

Coron et al. [4] used PSS-R to construct PSS-ES which realizes both an encryption and a signature with a common padding technique and key pair. PSS-ES is suitable for implementation, however, its security as an encryption scheme relies on the *partial-domain one-wayness* of f. Since the *partial-domain one-wayness* is stronger assumption than the *one-wayness*, the reduction efficiency is not tight and it must use long keys to achieve adequate security.

We propose new ES schemes, OAEP-ES, OAEP++-ES, and REACT-ES, which overcome this problem, and describe their security results. Since the security proofs of OAEP-ES, OAEP++-ES are similar to that of REACT-ES, we give the proofs of REACT-ES in Appendix A and B.

[8] Though the security of ES(S) can be ensured without the multiplicative property of f (which is not used in the security proof of ES(E) at all) as in [4], the reduction is not tight. Our interest is the comparison among ES schemes discussed in Section 5 in the practical situation, where RSA scheme is adopted as (f, f^{-1}), which satisfies the multiplicative property.

3.1 Methodology

We will give new ES schemes based on several encryption schemes which have a padding technique; OAEP+, OAEP++, and REACT. The simplest method[9] of constructing an ES scheme from encryption schemes seems by replacing encryption permutation f with its inverse f^{-1}.

Unfortunately, if we construct a new signature scheme from an encryption scheme by simple replacement of permutation of f with f^{-1}, its security is not ensured. For example, it is easy for a known-message attacker to generate an existential forgery under the *one-way* permutation with a special property in the similar way of Shoup's attack.

This is a formal explanation of this situation. In the security proof of a signature, in order to invert f on an input of integer η (*i.e.*, to calculate $f^{-1}(\eta)$), we embed η into some random oracle query about message x and random integer r (*e.g.*, consider the query $r||x$ to H' in OAEP+) and simulate another random oracle about r (*e.g.*, $G(r)$ in OAEP+). In this strategy, if the random oracle value about r (*e.g.*, $G(r)$) is already defined, we abort the simulation (fail to simulate). However, when the adversary can freely choose the query r, it implies that we fail to simulate this case with a high probability.

Therefore, there might be a possibility that we could generally construct a provably secure ES scheme from an encryption scheme as follows[10]: (i) we replace the r, which is an input for random oracle G, by a hash value of x and a new r'(*e.g.*, $r = w = H'(x||r')$), and (ii) we replace x with $x||r'$.

In this paper, we create ES schemes from OAEP+ and REACT, following this methodology.

3.2 OAEP-ES

A simple ES scheme can be created using OAEP+ [12], OAEP-ES. OAEP-ES relies for its security upon *only* the *one-wayness* of the permutation, so it is more practical than PSS-ES. OAEP-ES has, however, worse reduction efficiency than OAEP++-ES and REACT-ES as we will show. A description of OAEP-ES and its security results are as follows.

OAEP-ES with hash functions $G : \{0,1\}^{k_1} \to \{0,1\}^{n+k_0}$ and $H, H' : \{0,1\}^{n+k_0} \to \{0,1\}^{k_1}$, and the common padding scheme μ_1 (Figure 1) and key pair (f, f^{-1})[11], is executed as follows:

—*Encryption and Signing*: In order to encrypt or to sign x, we choose $r \xleftarrow{R}$

[9] Coron et al. simply constructed an encryption scheme by replacing signing permutation f^{-1} of PSS-R with f and proposed PSS-ES which has the same padding technique as PSS-R.

[10] Reference [8] gives a detailed explanation of this methodology.

[11] In the general model, we assume that $f : \{0,1\}^k \to \{0,1\}^k$ is a multiplicative permutation. If the implementation uses RSA permutation: $\mathbb{Z}_n \to \mathbb{Z}_n$, we put "0" in front of the padding data to make the domain k bit integer. In this case, the model and theorems will need to be adjusted. We adopt the same discussion for PSS-ES, OAEP++-ES, and REACT-ES.

$\{0,1\}^{k_0}$, set $w = H'(x||r) \in \{0,1\}^{k_1}$, and calculate $s = (x||r) \oplus G(w)$, $t = H(s) \oplus w$, and $\mu_1(x,r) = s||t$. We then return $y = f(\mu_1(x,r))$ as the ciphertext or $\sigma = f^{-1}(\mu_1(x,r))$ as the signature, respectively.
—*Decryption and Verification*: For ciphertext y or signature σ, we recover $s||t = f^{-1}(y)$ or $s||t = f(\sigma)$ ($|s| = n + k_0$, $|t| = k_1$), respectively. Next, we calculate $w = t \oplus H(s)$, divide $x||r = s \oplus G(w)$ ($|x| = n$, $|r| = k_0$), and check whether $w = H'(x||r)$. If the check passes, we return x; otherwise Reject.

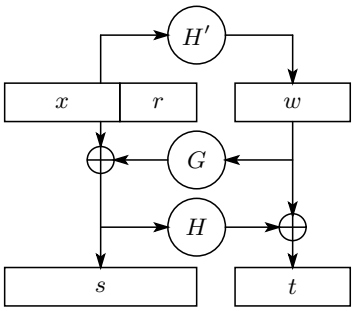

Fig. 1. Padding Techniques μ_1 for ES Schemes

The security results of OAEP-ES are as follows:

Theorem 1 (Security result of OAEP-ES(E)). *Let $\mathcal{A}$ be an adversary that breaks OAEP-ES in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of IND-CCA2&ACMA. Then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geqq \epsilon - \frac{q_{H'}+q_\Sigma}{2^{k_0}} - \frac{(q_{H'}+q_\Sigma+1)(q_G+q_{H'}+q_\Sigma)+q_D}{2^{k_1}} \\ \tau' \leqq \tau + \{(q_G + q_{H'} + q_\Sigma)(q_H + q_{H'} + q_\Sigma) + q_{H'} + q_\Sigma\}T_f \end{cases}$$

where T_f denotes the time complexity of f.

Theorem 2 (Security result of OAEP-ES(S)). *Let $\mathcal{F}$ be a forger that breaks OAEP-ES in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of EUF-CCA2&ACMA. Then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geqq \epsilon - \frac{q_{H'}q_\Sigma}{2^{k_0}} - \frac{(q_{H'}+q_\Sigma)(q_G+q_{H'}+q_\Sigma)+q_D+1}{2^{k_1}} \\ \tau' \leqq \tau + (2q_{H'} + 2q_\Sigma + 1)T_f \end{cases}$$

where T_f denotes the time complexity of f.

3.3 REACT-ES

REACT was proposed by Okamoto and Pointcheval [10]. To use REACT for encryption, we first generate random integer r and encrypt plaintext x by a symmetric encryption scheme with the hash value of r as the key. Second, we encrypt r by an asymmetric encryption scheme and send it with ciphertext of x and a check code.

Therefore, in REACT, once we encrypt r with the asymmetric encryption scheme, we can send a long plaintext using the symmetric encryption scheme with high speed (which, so-called, is KEM (Key Encapsulation Mechanism, [13])); REACT is more practical in terms of communication efficiency than OAEP, OAEP+, and OAEP++. Moreover, Nakashima and Okamoto [9] showed that REACT has tighter security than OAEP or OAEP+.

REACT-ES with hash functions $G : \{0,1\}^{k_1} \to \{0,1\}^{k_3}$, $H' : \{0,1\}^{n+k_0} \to \{0,1\}^{k_1}$, and $H : \{0,1\}^{2(n+k_0+k_1)} \to \{0,1\}^{k_2}$ $(k = k_1)$, symmetric encryption scheme E^{sym}_{key}, where key length is k_3, public key f, and private key f^{-1}, is executed as follows (Figure 2):

—*Encryption and Signing*: In order to encrypt or to sign x, we choose $r \xleftarrow{R} \{0,1\}^{k_0}$, set $w = H'(x||r) \in \{0,1\}^{k_1}$, and calculate $c_2 = E^{sym}_{G(w)}(x||r)$. Next, we set $c_1 = f(w)$ for encryption or $c_1 = f^{-1}(w)$ for signing, and return $(c_1, c_2, c_3 = H(x||r, w, c_1, c_2))$ as the ciphertext or signature, respectively.

—*Decryption and Verification*: For ciphertext (c_1, c_2, c_3) or signature (c'_1, c_2, c_3), we recover $w = f^{-1}(c_1)$ or $w = f(c'_1)$, respectively. Next, we calculate $x||r$ from $E^{sym}_{G(w)}(c_2)$, and check whether both "$w = H'(x||r)$ and $c_3 = H(x||r, w, c_1, c_2)$" or both "$w = H'(x||r)$ and $c_3 = H(x||r, w, c'_1, c_2)$", respectively. If the check passes, we return x as the plaintext or the message, respectively; otherwise Reject.

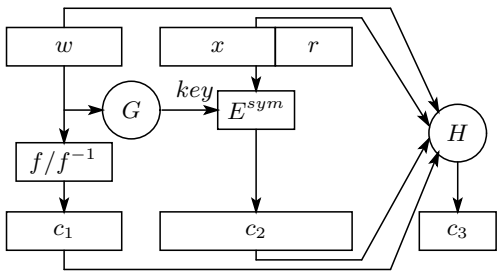

Fig. 2. REACT-ES

We use the following theorems to examine the security of REACT-ES. The proof are described in Appendix A and B, respectively.

Theorem 3 (Security result of REACT-ES(E)). *Let the symmetric encryption scheme be (τ', ν)-secure*[12]*, and let $\mathcal{A}$ be an adversary that breaks REACT-ES in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of IND-CCA2&ACMA. Then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geqq \epsilon - \nu - \frac{q_{H'}+q_\Sigma}{2^{k_0}} - \frac{q_D}{2^{k_1}} \\ \tau' \leqq \tau + (q_G + q_H + 2q_{H'} + 2q_\Sigma)T_f \end{cases}$$

where T_f denotes the time complexity of f.

[12] See the definition of security model of symmetric encryption scheme, §2.2 of [10].

Theorem 4 (Security result of REACT-ES(S)). *Let $\mathcal{F}$ be a forger that breaks REACT-ES in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of* EUF-CCA2&ACMA. *Then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geqq \epsilon - \frac{q_{H'} q_\Sigma}{2^{k_0}} - \frac{q_D+1}{2^{k_1}} \\ \tau' \leqq \tau + (2q_{H'} + 2q_\Sigma + 1)T_f \end{cases}$$

where T_f denotes the time complexity of f.

4 PSS-ES

4.1 Security of PSS-ES(E)

The security proof of PSS-ES(E) in [4] (Theorem 2 and Lemma 4) has two minor technical mistakes as follows: (i) the number of queries (about w) to G is not $q_{H'} + q_\Sigma$ (the last line in page 14 of [4]) but $q_G + q_{H'} + q_\Sigma$ because $G(w)$ may be defined by query w to G directly, (ii) this proof overlooks calculation time $(q_{H'} + q_\Sigma)T_f$ as part of the cost of querying the decryption oracle (line 10 in page 14 of [4], reading in Lemma 1's results into proof of Lemma 4). This consideration of these problems yields the following security result.

Theorem 5 (Security result of PSS-ES(E)). *Let $\mathcal{A}$ be an adversary that breaks PSS-ES(E) in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, \epsilon)$ in the sense of* IND-CCA2&ACMA. *Then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{pd-ow}}(\tau') \geqq \frac{1}{q_G+q_{H'}+q_\Sigma} \left(\epsilon - \frac{q_{H'}+q_\Sigma}{2^{k_0}} - \frac{(q_{H'}+q_\Sigma)(q_G+q_{H'}+q_\Sigma)+q_D}{2^{k_1}} \right) \\ \tau' \leqq \tau + 2(q_{H'} + q_\Sigma)T_f \end{cases}$$

where T_f denotes the time complexity of f.

4.2 Security of PSS-ES(S)

The proof of Theorem 3 in [4] has three minor technical mistakes as follows: (i) it misses the probability $\frac{q_\Sigma}{2^{k_0}}$ that appears because $\mathcal{I}$ cannot answer the signing query for the pair of message and random integers implanting η previously[13], (ii) the number of queries w to G is not $q_{H'} + q_\Sigma$ (line 18 in page 16 of [4]) but $q_G + q_{H'} + q_\Sigma$ because $G(w)$ may be defined by the query w to G directly, (iii) this proof overlooks the calculation time $(q_{H'} + q_\Sigma)T_f$ as part of the cost of querying the decryption oracle (line 9 in page 16 of [4], reading in Lemma 1's results into proof of Theorem 3). This consideration of these problems yields the following security result.

Theorem 6 (Security result of PSS-ES(S)). *If $\mathcal{F}$ is a forger that breaks PSS-ES(S) in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, \epsilon)$ in the sense of* EUF-CCA2&ACMA, *then:*

$$\begin{cases} \mathsf{Succ}^{\mathsf{ow}}(\tau') \geqq \epsilon - \frac{q_{H'} q_\Sigma}{2^{k_0}} - \frac{(q_{H'}+q_\Sigma)(q_G+q_{H'}+q_\Sigma)+q_D+1}{2^{k_1}} \\ \tau' \leqq \tau + (2q_{H'} + 2q_\Sigma + 1)T_f \end{cases}$$

where T_f denotes the time complexity of f.

[13] In our results, since η is embedded $q_{H'}$ times, the corresponding probability is $\frac{q_{H'} q_\Sigma}{2^{k_0}}$.

5 Reduction Efficiency

We evaluate the security of RSA-OAEP-ES and RSA-REACT-ES following the approach taken by Nakashima and Okamoto [9] and compare them to RSA-PSS-ES. For each scheme, we consider the usages of encryption and signature.

Reference [9] uses the recommended key size in order to confirm that no adversary has the ability to break the 1024, 2048 bits factoring problem. In estimating the key size, we use Lemma 4 of [5] to modify the security statement of RSA-PSS-ES; that is, f's *partial-domain one-wayness* is replaced by *one-wayness* of RSA permutation paying the cost of running time and decreasing the success probability.

Throughout this evaluation, we assume that breaking the RSA problem is equivalent to solving the factoring problem, and that k_0 and k_1 are enough large so that factors that suppress the reduction efficiency can be ignored. The complexity of the factoring problem is measured by applying a number field sieve. Table 1 shows the recommended key size that achieves the same complexity as the 1024, 2048 bits factoring problem.

Table 1. Recommended key size

Scheme		1024bit	2048bit
PSS-ES	Encryption	6221	12452
	Signature	1363	2596
OAEP-ES	Encryption	5252	10838
	Signature	1363	2596
REACT-ES	Encryption	1363	2596
	Signature	1363	2596

As in Table 1, OAEP-ES has better reduction efficiency than PSS-ES because the security of PSS-ES(E) is based on *partial-domain one-wayness*. Therefore, compared to PSS-ES, OAEP-ES can decrease the key size by more than 950 bits for the 1024 bits factoring problem and by more than 1600 bits for 2048 bits factoring problem.

Moreover, as in Table 1, REACT-ES offers much better reduction efficiency than PSS-ES and OAEP-ES, and the key size does not increase comparing with the number of bits in the factoring problem[14]. This is because the running time of the permutation inverter of REACT-ES is of the order of $q_{H'}$ while that of OAEP-ES is of order of $q_G q_H$. This means that the key length of REACT-ES is shorter than that of OAEP-ES.

[14] OAEP++-ES has the same reduction efficiency as REACT-ES, *i.e.*, the recommended key size for OAEP++-ES is the same as the one for REACT-ES.

6 Discussion

REACT-ES is superior to OAEP-ES in terms of the running time of the permutation inverter, as shown by Theorem 3 (moreover, since PSS-ES owes its security to *partial-domain one-wayness*, its reduction efficiency is not good).

More precisely, when inverting the permutation for PSS-ES and OAEP-ES, the inverter should locate the preimage using the product of two hash functions' input/output lists[15] (G-List and H-List). The inverter of REACT-ES, however, locates the preimage using the sum of two lists (H-List and G-List).Accordingly, the running time of the above theorem on REACT-ES is less than those on PSS-ES and OAEP-ES.

Therefore, as described in Section 5, the recommended key sizes that provide the same complexity as the 1024, 2048 bits factoring problem are, for OAEP++-ES and REACT-ES, much shorter than those of PSS-ES and OAEP-ES, and are about the same as the bit size of the factoring problem.

With regard to communication efficiency, the length of plaintext or message, in PSS-ES, OAEP-ES, and OAEP++-ES, is restricted to the key size. REACT-ES, however, allows us to encrypt (sign) arbitrary length of plaintext (message) by using symmetric encryption; it follows that REACT-ES is the most practical technique giving the high reduction efficiency of its security proof and its improved communication efficiency.

7 Conclusion

This paper first gave the general methodology to construct an ES scheme from an encryption scheme with a padding technique and proposed new ES schemes, OAEP-ES, OAEP++-ES, and REACT-ES, which use a unique padding technique and key pair to realize encryption and signature. It also proved that these two usages of proposed schemes satisfy IND-CCA2&ACMA and EUF-CCA2&ACMA, respectively. These schemes are suitable for implementation because they need only one padding technique and key pair.

Moreover, OAEP++-ES and REACT-ES offer much better reduction efficiency than PSS-ES and OAEP-ES. Using the evaluation of [9], the difficulty of breaking OAEP++-ES and REACT-ES is almost equal to that of the key size factoring problem. Hence, we conclude that OAEP++-ES and REACT-ES are more efficient than PSS-ES or OAEP-ES. Furthermore, from the view of the communication efficiency, REACT-ES allows us to encrypt (sign) a plaintext (message) arbitrary length through the use of symmetric encryption; we can conclude that REACT-ES is the most practical candidate due to the tightness of its security and its improved communication efficiency.

This paper also corrected the original mistakes made in proving the security of PSS-ES.

[15] For PSS-ES, when replacing the *partial-domain one-wayness* to the *one-wayness* as in Lemma 4 of [5], we ought to run the adversary twice and get two input/output lists (two G-Lists).

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, pages 26–45. Springer, 1998. Lecture Notes in Computer Science No. 1462.
2. M. Bellare and P. Rogaway. Optimal asymetric encryption — how to encrypt with RSA. In A.D. Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Berlin, Heidelberg, New York, 1995. Springer-Verlag.
3. M. Bellare and P. Rogaway. The exact security of digital signatures –how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, Heidelberg, New York, 1996. Springer-Verlag.
4. J. S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal padding schemes for RSA. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2422 of *Lecture Notes in Computer Science*, pages 226–241, Berlin, Heidelberg, New York, 2002. Springer-Verlag.
5. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is chosen-ciphertext secure under the RSA assumption. *Journal of Cryptology*, 2002.
6. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme against adaptive chosen message attack. *Journal of Computing (Society for Industrial and Applied Mathematics)*, 17(2):281–308, 1988.
7. K. Kobara and H. Imai. OAEP++ : A very simple way to apply OAEP to deterministic OW-CPA primitives. 2002. Available at `http://eprint.iacr.org/2002/130/`.
8. Y. Komano and K. Ohta. OAEP-ES – Methodology of universal padding technique. *manuscript*, 2003.
9. T. Nakashima and T. Okamoto. Key size evaluation of provably secure RSA-based encryption schemes. *SCIS 2002, The 2002 Symposium on Cryptography and Information Security*, 2002.
10. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Encryptosystem Tranceform. In D. Naccache, editor, *CT – RSA '2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175, Berlin, Heidelberg, New York, 2001. Springer-Verlag.
11. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
12. V. Shoup. OAEP reconsidered. In J. Kilian, editor, *Advances in Cryptology — CRYPTO'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259, Berlin, Heidelberg, New York, 2001. Springer-Verlag.
13. V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). In *manuscript*, 2001. `http://shoup.net/papers/`.
14. Y. Zheng. Degital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In *Advances in Cryptology — CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179, Berlin, Heidelberg, New York, 1997. Springer-Verlag.

A Proof of Theorem 3

We follow the definition of symmetric encryption scheme and its security model from [10]. In the security proof of Theorem 3, assume that the symmetric encryption scheme is (τ', ν)-secure.

A.1 Construction of Inverter $\mathcal{I}$

We give the construction of inverter $\mathcal{I}$ that breaks the *one-wayness* of f about c^+, by using adversary $\mathcal{A}$ that breaks REACT-ES(E) in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of IND-CCA2&ACMA, as follows: we input public key f to $\mathcal{A}$, answer the queries that $\mathcal{A}$ asks to the random oracles, to the decryption oracle, and to the signing oracle in the following way, and receive challenge (x_0, x_1). We then choose $b \stackrel{R}{\leftarrow} \{0,1\}$, $r^+ \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, and $\mathsf{k} \stackrel{R}{\leftarrow} \{0,1\}^{k_3}$ and put $c_2^+ = E_{\mathsf{k}}^{sym}(x_b||r^+)$. Moreover, we answer the queries that $\mathcal{A}$ asks in the following way, and finally, receive $\hat{b}$ (or stop $\mathcal{A}$ after its running time τ is over).

In simulating random oracles G, H', and H, we construct input/output lists, G-List, H′-List, and H-List, respectively. In G-List, we preserve pair $(w, G(w))$ of query w and answer $G(w)$. In H′-List, we keep seven-tuple $(x||r, H'(x||r), z, c_1, c_2, c_3, c_3')$ of query $x||r$, answer $H'(x||r)$, guarantee z, c_2, c_3' for signing queries, and pledge c_1, c_2, c_3 for decryption queries. In H-List, we preserve sextuplet $(w, x||r, c_1, c_2, H(w, x||r, c_1, c_2))$ of query $w, x||r, c_1, c_2$ and answer $H(w, x||r, c_1, c_2)$.

Answering the random oracle queries to G, H', and H: For new query w to G, we choose a random integer from $\{0,1\}^{k_3}$, put it to $G(w)$, answer to $\mathcal{A}$, and add $(w, G(w))$ to G-List. If w has already been queried to G, we locate $(w, G(w)) \in$ G-List and answer $G(w)$.

For new query $(w, x||r, c_1, c_2)$ to H, we choose random integer c_3 from $\{0,1\}^{k_2}$, put it to $H(w, x||r, c_1, c_2)$, answer to $\mathcal{A}$, and add $(w, x||r, c_1, c_2, c_3)$ to H-List. Moreover, we simulate $G(w)$ in the above way[16]. If $(w, x||r, c_1, c_2)$ has already been queried to H, we locate $(w, x||r, c_1, c_2, c_3) \in$ H-List and answer c_3.

For new query $x||r$ to H', we get $z \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$, set $f(z) = w$, and calculate $c_1 = f(w)$. Next, we simulate $G(w)$ in the same way described above, calculate $c_2 = E_{G(w)}^{sym}(x||r)$. Finally, we put $c_3 = H(w, x||r, c_1, c_2)$ and $c_3' = H(w, x||r, z, c_2)$ by simulating H in the same way described above, answer w as $H'(x||r)$ to $\mathcal{A}$, and add $(x||r, w, z, c_1, c_2, c_3, c_3')$ to H′-List. If $x||r$ has already been queried to H', we locate $(x||r, w, *, *, *, *, *) \in$ H′-List and answer w.

Answering the decryption queries to D: In order for decryption query $y = (c_1, c_2, c_3)$ to be valid ciphertext, $(x||r, *, *, c_1, c_2, c_3, *)$ must be contained in H′-List. In this case, we can answer with the corresponding plaintext x. Otherwise, we answer Reject since the probability of $H'(x||r) = w$ is negligible.

Answering the signing queries to Σ: For signing query x to Σ, we get $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and check whether $(x||r, *, z, *, c_2, *, c_3')$ is in H′-List. If so, we answer

[16] We simulate G because we want to collect the information of input/output on w in G-List; this makes the estimation of the success probability of the permutation inversion easy.

$\sigma = (z, c_2, c_3')$ to $\mathcal{A}$ as a signature. Otherwise, we choose $z \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$, set $f(z) = w$, and calculate $c_1 = f(w)$. Next, we simulate $G(w)$ in the same way described above, calculate $c_2 = E^{sym}_{G(w)}(x||r)$. Finally, we put $c_3 = H(w, x||r, c_1, c_2)$ and $c_3' = H(w, x||r, z, c_2)$ by simulating H in the same way described above, add $(x||r, w, z, c_1, c_2, c_3, c_3')$ to $\mathsf{H'\text{-}List}$, and answer $\sigma = (z, c_2, c_3')$ as a signature to $\mathcal{A}$.

A.2 Analysis

Let $y^+ = (c^+, c_2^+, c_3^+)$ be a target ciphertext that we answer to $\mathcal{A}$ deviating the protocol, and w^+, r^+, and x^+ be corresponding elements. In order to analyze the success probability of $\mathcal{I}$, we use following notations: AskG and $\mathsf{AskH'}$ are events for which $(w^+, *) \in \mathsf{G\text{-}List}$, and $(*||r^+, *, *, *, *, *, *) \in \mathsf{H'\text{-}List}$, respectively, and moreover, let EBad be an event[17] that $\mathsf{AskH'} \wedge [H'(x_i||r^+) \neq w^+$ for $i = 0, 1]$, let DBad be an event that we fail to simulate in D, and let $\mathsf{Bad} = \mathsf{EBad} \vee \mathsf{DBad}$[18]. Our aim in setting these notations is to estimate the probability of AskG. At first, we divide this event as follows:

$$\Pr[\mathsf{AskG}] = \Pr[\mathsf{AskG} \wedge \mathsf{Bad}] + \Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}]. \tag{1}$$

With regard to $\Pr[\mathsf{AskG} \wedge \mathsf{Bad}]$ in equation (1), from the definition of Bad, we have

$$\begin{aligned}\Pr[\mathsf{AskG} \wedge \mathsf{Bad}] &= \Pr[\mathsf{Bad}] - \Pr[\neg\mathsf{AskG} \wedge \mathsf{Bad}] \\ &\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{EBad}|\neg\mathsf{AskG}] - \Pr[\mathsf{DBad}|\neg\mathsf{AskG}]. \end{aligned}\tag{2}$$

We can estimate $\Pr[\mathsf{EBad}|\neg\mathsf{AskG}]$ in inequality (2) because, by the definition of EBad, we have $\Pr[\mathsf{EBad}|\neg\mathsf{AskG}] \leqq \Pr[\mathsf{AskH'}|\neg\mathsf{AskG}]$. Here, $\Pr[\mathsf{AskH'}|\neg\mathsf{AskG}] \leqq \frac{q_{H'}+q_\Sigma}{2^{k_0}}$, because if $\neg\mathsf{AskG}$ occurs, $G(w^+)$ and r^+ are random integers for $\mathcal{A}$ and it is only by accident that $*||r^+$ is queried to H'.

Moreover, $\Pr[\mathsf{DBad}|\neg\mathsf{AskG}]$ in inequality (2) is less than $\frac{q_D}{2^{k_1}}$. Note that in answering to decryption query (c_1, c_2, c_3), we search $\mathsf{H'\text{-}List}$ for corresponding plaintext x, therefore we fail to simulate the decryption oracle if $\mathcal{A}$ does not query H' about $x||r$ and ciphertext (decryption query) y output by $\mathcal{A}$ is valid. However, if $\mathcal{A}$ does not query H' about $x||r$, $H'(x||r)$ is uniformly distributed in $\{0,1\}^{k_1}$, and then, it is only by accident (with probability $\frac{1}{2^{k_1}}$) that $w = f^{-1}(c_1)$ equals $H'(x||r)$.

Hence, we can evaluate $\Pr[\mathsf{AskG} \wedge \mathsf{Bad}]$ in equation (1) by

$$\Pr[\mathsf{AskG} \wedge \mathsf{Bad}] \geqq \Pr[\mathsf{Bad}] - \frac{q_{H'} + q_\Sigma}{2^{k_0}} - \frac{q_D}{2^{k_1}}. \tag{3}$$

With regard to the second term of equation (1), it is meaningful to consider the advantage of $\mathcal{A}$ because of the condition $\neg\mathsf{Bad}$. We can do this by evaluating

[17] In this event, $\mathcal{A}$ may notice that we answer y^+ as a target ciphertext deviating the protocol.

[18] Note that we never fail to simulate the answer to the signing query, described in section A.1, and do not need to consider event $\Sigma\mathsf{Bad}$.

$\Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}]$ as follows:

$$\begin{aligned}\Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}] &\geq \Pr[\mathcal{A} = b \wedge \mathsf{AskG} \wedge \neg\mathsf{Bad}] \\ &= \Pr[\mathcal{A} = b \wedge \neg\mathsf{Bad}] - \Pr[\mathcal{A} = b \wedge \neg\mathsf{AskG} \wedge \neg\mathsf{Bad}]. \quad (4)\end{aligned}$$

In inequality (4), both

$$\Pr[\mathcal{A} = b \wedge \neg\mathsf{Bad}] \geqq \Pr[\mathcal{A} = b] - \Pr[\mathsf{Bad}] = (\frac{\epsilon}{2} + \frac{1}{2}) - \Pr[\mathsf{Bad}]$$

and[19]

$$\begin{aligned}\Pr[\mathcal{A} = b \wedge \neg\mathsf{AskG} \wedge \neg\mathsf{Bad}] &= \Pr[\mathcal{A} = b | \neg\mathsf{AskG} \wedge \neg\mathsf{Bad}] \Pr[\neg\mathsf{AskG} \wedge \neg\mathsf{Bad}] \\ &= (\frac{\nu}{2} + \frac{1}{2})(1 - \Pr[\mathsf{Bad}] - \Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}]) \\ &\leqq \frac{1}{2}(1 - \Pr[\mathsf{Bad}] - \Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}]) + \frac{\nu}{2} \cdot 1.\end{aligned}$$

hold[20]. Therefore, by substituting above two inequalities into (4),

$$\Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}] \geqq \frac{\epsilon - \nu - \Pr[\mathsf{Bad}] + \Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}]}{2}$$

holds and this inequality leads to

$$\Pr[\mathsf{AskG} \wedge \neg\mathsf{Bad}] \geqq \epsilon - \nu - \Pr[\mathsf{Bad}]. \quad (5)$$

Hence, the considerations of equation (1) and inequalities (3) and (5) conclude the proof of Theorem 3.

The running time τ' of $\mathcal{I}$ is the sum of the following terms: (i) the running time τ of $\mathcal{A}$ because we run $\mathcal{A}$ once, (ii) in order to find corresponding pair from $\mathsf{G\text{-}List}$ to c^+, we compute f at most $q_G + q_{H'} + q_H + q_\Sigma$ times, *i.e.*, $(q_G + q_{H'} + q_H + q_\Sigma)T_f$, (iii) in order to be able to simulate D and Σ, we calculate both $f(z)$ and $f(w)$ in simulation of H' and Σ[21] $q_{H'} + q_\Sigma$ times, *i.e.*, $(q_{H'} + q_\Sigma)T_f$. Hence, $\tau' \leqq \tau + (q_G + q_H + 2q_{H'} + 2q_\Sigma)T_f$ holds.

[19] Note that in our simulation, if $\mathcal{A}$ notices the deviation (*i.e.*, if event Bad occurs), it does not run for some pairs of random coins of $\mathcal{A}$ and $\mathcal{I}$. Therefore, $\Pr[\mathcal{A} = b]$ in this inequality is taken over the random coins of $\mathcal{A}$ and $\mathcal{I}$ in which $\mathcal{A}$ does not notice the deviation. Though the probabilistic space is restricted and smaller than the entire probabilistic space, the probability of the event that $\mathcal{A} = b$ is equal to the one taken over the entire probabilistic space, from the definition of the random oracle model; $\Pr[\mathcal{A} = b] = \frac{\epsilon}{2} + \frac{1}{2}$.

[20] Note that the probability that $\mathcal{A} = b$ holds under the condition of $\neg\mathsf{AskG}$ and $\neg\mathsf{Bad}$ is equal to the probability that $\mathcal{A}$ can distinguish b from x_0, x_1 and c_2^+, without secret key k; $\Pr[\mathcal{A} = b | \neg\mathsf{AskG} \wedge \neg\mathsf{Bad}] = \frac{\nu}{2} + \frac{1}{2}$. This is because from $\neg\mathsf{Bad}$, $\mathcal{A}$ cannot notice the deviation and performs the same way as in the real run. Moreover, from $\neg\mathsf{AskG}$, $\mathcal{A}$ cannot know $\mathsf{k} = G(w^+)$.

[21] This seems to require the calculation of f $2(q_{H'} + q_\Sigma)$ times, but $q_{H'} + q_\Sigma$ calculations are sufficient. Indeed, when we add an element including w to $\mathsf{G\text{-}List}$ or $\mathsf{H\text{-}List}$, we check whether $f(w) = c^+$ holds. This action plays the role of preparing for the simulation of D and is already counted in (ii). Therefore, we consider only the preparation for the signing oracle queries in (iii).

B Proof of Theorem 4

B.1 Construction of Inverter $\mathcal{I}$

We give the construction of inverter $\mathcal{I}$ that breaks the *one-wayness* of f about η, by using forger $\mathcal{F}$ that breaks REACT-ES(S) in $(\tau, q_D, q_\Sigma, q_G, q_{H'}, q_H, \epsilon)$ in the sense of EUF-CCA2&ACMA as follows: we input public key f to $\mathcal{F}$, answer the queries that $\mathcal{F}$ asks to the random oracles, to the decryption oracle, and to the signing oracle in the same way in section A.1, except those to H' and Σ (described below). Finally, we receive forgery $\sigma^* = (c_1^*, c_2^*, c_3^*)$ (or stop $\mathcal{F}$ after its running time τ is over.)

In simulating random oracles G, H', and H, we construct input/output lists, G-List, H′-List, and H-List, respectively. G-List holds $(w, G(w))$, the pairing of query w and answer $G(w)$. H′-List holds $(b, x||r, H'(x||r), z, c_1, c_2, c_3.c_3')$, the bit $b = 0/1$, query $x||r$, answer $H'(x||r)$, guarantee z, c_2, c_3' for signing queries, and pledge c_1, c_2, c_3 for decryption queries. H-List holds $(w, x||r, c_1, c_2, H(w, x||r, c_1, c_2))$, the pairing of query $w, x||r, c_1, c_2$ and answer $H(w, x||r, c_1, c_2)$.

Answering the random oracle queries to H'

For new query $x||r$ to H', we get $z \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$, set $f(z)\eta = w$, and calculate $c_1 = f(w)$. Next, we simulate $G(w)$ in the same way as in section A.1, calculate $c_2 = E^{sym}_{G(w)}(x||r)$. Finally, we put $c_3 = H(w, x||r, c_1, c_2)$ and $c_3' = H(w, x||r, z, c_2)$ by simulating in the same way as in section A.1, answer w as $H'(x||r)$ to $\mathcal{F}$, and add $(1, x||r, w, z, c_1, c_2, c_3, c_3')$ to H′-List. If $x||r$ has already been queried to H', we locate $(*, x||r, w, *, *, *, *, *) \in$ H′-List and answer w.

Answering the signing queries to Σ: For signing query x to Σ, we get $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ and check whether $(0, x||r, *, z, *, c_2, *, c_3')$ is in H′-List. If so, we answer $\sigma = (z, c_2, c_3')$ to $\mathcal{F}$ as a signature. Moreover, if $(1, x||r, *, *, *, *, *, *)$ is in H′-List, we abort. Otherwise, we choose $z \stackrel{R}{\leftarrow} \{0,1\}^{k_1}$, put $f(z) = w$, and calculate $c_1 = f(w)$. Next, we simulate $G(w)$ in the same way as in section A.1, calculate $c_2 = E^{sym}_{G(w)}(x||r)$. Finally, we put $c_3 = H(w, x||r, c_1, c_2)$ and $c_3' = H(w, x||r, z, c_2)$ by simulating in the same way as in section A.1, add $(0, x||r, w, z, c_1, c_2, c_3, c_3')$ to H′-List, and answer $\sigma = (z, c_2, c_3')$ as a signature to $\mathcal{F}$.

B.2 Analysis

Let $\sigma^* = (c_1^*, c_2^*, c_3^*)$ be a forgery output by $\mathcal{F}$; w^*, r^*, and x^* are the corresponding elements. In order to analyze the success probability of $\mathcal{I}$, let DBad be the same event as in A.2, ΣBad an event that $\mathcal{I}$ fails to simulate in Σ, and Bad = DBad $\vee$ ΣBad. Moreover, let S be an event that $\mathcal{V}_{\mathsf{pk}}(\sigma^*) = x^*$, and let AskH′ be one that $\mathcal{F}$ queries directly H' about $x^*||r^*$.

At first, we consider

$$1 = \Pr[\mathsf{Bad}] + \Pr[\neg\mathsf{Bad}]. \tag{6}$$

With regard to $\Pr[\mathsf{Bad}] \leqq \Pr[\mathsf{DBad}] + \Pr[\Sigma\mathsf{Bad}]$ in equation (6), we have

$$\Pr[\mathsf{Bad}] \leqq \frac{q_{H'} q_\Sigma}{2^{k_0}} + \frac{q_D}{2^{k_1}}. \tag{7}$$

In fact, $\Pr[\mathsf{DBad}]$ is evaluated in the same way as in section A.2. On the other hand, $\Pr[\Sigma\mathsf{Bad}]$ is bounded by $q_\Sigma(\frac{q_{H'}}{2^{k_0}})$. Note that in simulating the answer signing query x, we first choose random integer r and z for the candidate of the signature, and simulate H' about $x||r$. In this phase, $\Sigma\mathsf{Bad}$ occurs if $x||r$ has already queried to H' by $\mathcal{F}$ directly, because we can not calculate $f^{-1}(\eta)$. For a signing query, the probability that $x||r$ is queried to H' is bounded by $\frac{q_{H'}}{2^{k_0}}$[22] because of randomness of r, and then, we can estimate $\Pr[\Sigma\mathsf{Bad}]$ by $q_\Sigma(\frac{q_{H'}}{2^{k_0}})$.

With regard to $\Pr[\neg\mathsf{Bad}]$ in equation (6), we divide event $\neg\mathsf{Bad}$ by S and have

$$\Pr[\neg\mathsf{Bad}] = \Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] + \Pr[\neg\mathsf{S} \wedge \neg\mathsf{Bad}]. \tag{8}$$

In this equation (8),

$$\Pr[\neg\mathsf{S} \wedge \neg\mathsf{Bad}] \leqq \Pr[\neg\mathsf{S}|\neg\mathsf{Bad}] = 1 - \Pr[\mathsf{S}|\neg\mathsf{Bad}] = 1 - \epsilon \tag{9}$$

holds[23].

Next, we estimate $\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}]$ in equation (8) by dividing event $\mathsf{S} \wedge \neg\mathsf{Bad}$ by AskH':

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

In this equality, $\Pr[\mathsf{S} \wedge \neg\mathsf{Bad} \wedge \neg\mathsf{AskH}']$ is bounded by $\frac{1}{2^{k_1}}$ because it is an incident that $H'(x^*||r^*) = w^*$ if $(1, x^*||r^*, *, *, *, *, *, *) \notin \mathsf{H}'\text{-List}$ holds. On the other hand, we have $\Pr[\mathsf{S} \wedge \neg\mathsf{Bad} \wedge \mathsf{AskH}'] \leqq \Pr[\mathsf{S} \wedge \mathsf{AskH}'] \leqq \mathsf{Succ}^{\mathsf{ow}}(\tau')$ because if both $(1, x^*||r^*, *, z^*, *, *, *, *) \in \mathsf{H}'\text{-List}$ and S hold, then we can compute $\frac{c_1^*}{z^*} = \frac{f^{-1}(f(z^*)\eta)}{z^*} = f^{-1}(\eta)$ from the multiplicative property of f. Therefore, we have

$$\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] \leqq \mathsf{Succ}^{\mathsf{ow}}(\tau') + \frac{1}{2^{k_1}}. \tag{10}$$

By substituting inequalities (9) and (10) into equation (8), we have

$$\Pr[\neg\mathsf{Bad}] \leqq \mathsf{Succ}^{\mathsf{ow}}(\tau') + \frac{1}{2^{k_1}} + 1 - \epsilon. \tag{11}$$

Finally, if we substitute inequalities (7) and (11) into equation (6), we can conclude the proof of Theorem 4.

The running time τ' of $\mathcal{I}$ is the sum of the following terms: (i) the running time τ of $\mathcal{F}$ because we run $\mathcal{F}$ once, (ii) in the simulation of Σ, we have to prepare the answer for queries to D and Σ, *i.e.*, $2q_\Sigma T_f$, (iii) in the simulation of H', we have to prepare the answer for queries to D and to implant η, *i.e.*, $2q_{H'}T_f$, (iv) we have to find z^* corresponding to c_1^* by computing $f(c_1^*)$ once, *i.e.*, T_f. Hence, $\tau' \leqq \tau + (2q_{H'} + 2q_\Sigma + 1)T_f$ holds.

[22] Note that for signing query x, if we choose random integer r such that $x||r$ is queried to H' through the past signing query, we can reply this query by locating corresponding signature from $\mathsf{H}'\text{-List}$. Therefore, we only consider the case that $x||r$ has already queried to H' by $\mathcal{F}$ directly, in the estimation of $\Sigma\mathsf{Bad}$.

[23] Note that the success probability of $\mathcal{F}$ under the condition that $\mathcal{F}$ does not notice the simulation is equal to the one in real run and this leads $\Pr[\mathsf{S}|\neg\mathsf{Bad}] = \epsilon$.