Security of Digital Signature Schemes in Weakened Random Oracle Models

Akira Numayama[1], Toshiyuki Isshiki[1,2], and Keisuke Tanaka[1]

[1] Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{numayama.a.aa@m,keisuke@is}.titech.ac.jp
[2] NEC Corporation, 1753 Shimonumabe Nakahara-ku Kawasaki, Kanagawa 211-8666, Japan
t-issiki@bx.jp.nec.com

Abstract. We formalize the notion of several weakened random oracle models in order to capture which property of a hash function is crucial to prove the security of a cryptographic scheme. In particular, we focus on augmenting the random oracle with additional oracles that respectively return collisions, second-preimages, and first-preimages. We study the security of the full domain hash signature scheme, as well as three variants thereof in the weakened random oracle models, leading to a separation result.

Keywords: random oracle model, digital signature, collision, preimage.

1 Introduction

Background. When analyzing the security of cryptographic schemes, we often idealize hash functions as truly random functions called random oracles. A number of schemes were proposed and proved secure in the random oracle model (ROM) [1,2,3,4,5].

When it comes to implementations of the cryptographic schemes, we have to replace the random oracles by cryptographic hash functions. This replacement might make the cryptographic schemes insecure.

An important thing is that one should carefully observe the properties of the ROM, which are necessary for proving the security of the schemes, and replace the random oracles with some *suitable* hash functions. For example the security of the *hash-and-sign* type signature schemes, which are secure in the ROM, relies on the collision resistance property of the ROM. If one can obtain two distinct m, m' such that $h(\mathsf{m}) = h(\mathsf{m}')$ and the signature $\sigma = \mathsf{Sig}(h(\mathsf{m}))$, then (m', σ) is a valid forgery. Therefore, this case requires that a hash function is collision resistant.

Recent progress [6,7] on the attacks against cryptographic hash functions such as SHA-1 and MD5, raises the question on the assumption that hash functions are collision resistant. Therefore, it is interesting to know whether the collision resistance property of the ROM is necessary for proving the security of the schemes. More generally, it is worth classifying the schemes by the properties of the ROM that their security essentially rely on.

Previous works. Recent works [8,9,10,11] introduced variants of the random oracle model, where some properties of the ROM are weakened. If one can prove that a cryptographic scheme is secure in the ROM but not in a weakened random oracle model, then the security of the scheme essentially relies on the difference between these models.

Unruh [8] proposed a random oracle model where *oracle-dependent* auxiliary inputs are allowed. In this setting, the adversary of some cryptographic protocol obtains an auxiliary input that can contain information about the random oracle (e.g. collisions). He showed that the RSA-OAEP encryption scheme [2] is secure in the random oracle model even in the presence of *oracle-dependent* auxiliary inputs.

Nielsen [9] proposed the *non-programmable* random oracle model where the random oracle is not *programmable*. In this model, one cannot set the value that the random oracle answers to some appropriate value. The author showed that a non-interactive non-committing encryption scheme exists in the ROM (assuming trapdoor permutations exists), but not in the *non-programmable* random oracle model.

Liskov [10] proposed the models of weak hash functions where there exist the random oracle and the additional oracles that break some properties of the ROM. He listed several such oracles that provide, for example, collisions. He also proposed a general construction of a hash function from weak hash functions. Pasini and Vaudenay [11] applied Liskov's idea to the security analysis of digital signature schemes. They considered the security of *hash-then-sign* type signature schemes in the random oracle model with an additional oracle that returns first-preimages. In the security analysis of signature schemes in their model, the reduction algorithm simulates both the random oracle and the additional oracle.

Our contributions. By using Liskov's idea, we propose the following three models: the *collision tractable* random oracle model (CT-ROM), the *second-preimage tractable* random oracle model (SPT-ROM), and the *first-preimage tractable* random oracle model (FPT-ROM). The CT-ROM (resp. SPT-ROM, FPT-ROM) consists of the random oracle and the collision (resp. second-preimage, first-preimage) oracle that returns collisions (resp. second-preimages, first-preimages).

Our models are a bit different from those of Liskov with respect to: first, in our model, the collision oracle may not provide a collision even if there are collisions, while in the Liskov model it always provides a collision; second, in our model, the second-preimage (resp. first-preimage) oracle provides $\perp$ if there is no second-preimage (resp. first-preimage). Liskov only considered compression functions, where there are some collisions and preimages with high probability. When taking into account expanding functions, the Liskov model turns out to be too strong.

Notice here that it can be shown that the security with respect to the random oracle model with *oracle-dependent* auxiliary input implies the security with respect to the CT-ROM, since the *oracle-dependent* auxiliary input can contain a sufficiently long list of collisions. For the security with respect to the SPT-ROM and the FPT-ROM, the proof technique employed in [8] cannot be applied to our models. This is because the random oracle model with *oracle-dependent* auxiliary input does not capture the attack models with adaptive queries.

In almost all the proofs employing the random oracles, the reduction algorithms simulate the random oracles with embedding the target problem instances. We give new oracle simulation methods that are applicable for our models. These methods are useful to simulate both the random oracle and the additional oracles when analyzing the security of cryptographic schemes.

In our models, we consider the security of two RSA-based signature schemes: RSA-FDH [3] and RSA-PFDH [12], which are simple and popular. In particular, we focus on the existential unforgeability under the adaptive chosen message attack [13], and show the following statements.

1. RSA-FDH is *not* secure in the CT-ROM.
2. RSA-PFDH is secure in the CT-ROM, but *not* secure in the SPT-ROM.

Moreover, we slightly modify RSA-PFDH to obtain two variants which we call RSA-PFDH^{+} and RSA-PFDH$^{\oplus}$. We consider their security and show the following statements.

3. RSA-PFDH^{+} is secure in the SPT-ROM, but *not* secure in the FPT-ROM.
4. RSA-PFDH$^{\oplus}$ is secure in the FPT-ROM.

We summarize the security of the four schemes in Table 1.

Table 1. Security of four schemes

scheme\model	ROM	CT-ROM	SPT-ROM	FPT-ROM
RSA-FDH	secure	insecure		
RSA-PFDH	secure		insecure	
RSA-PFDH^{+}	secure			insecure
RSA-PFDH$^{\oplus}$	secure			

In conclusion, we show the relations among our models. Let $\mathbf{S}$ be a security notion and $\mathbf{M}_1, \mathbf{M}_2$ models. Let $\mathbf{S}/\mathbf{M}_1 \Rightarrow \mathbf{S}/\mathbf{M}_2$ and $\mathbf{S}/\mathbf{M}_1 \nRightarrow \mathbf{S}/\mathbf{M}_2$ be as follows.

– $\mathbf{S}/\mathbf{M}_1 \Rightarrow \mathbf{S}/\mathbf{M}_2$: for any signature scheme Σ if Σ meets a security notion $\mathbf{S}$ in the model $\mathbf{M}_1$, then Σ also meets $\mathbf{S}$ in the model $\mathbf{M}_2$.
– $\mathbf{S}/\mathbf{M}_1 \nRightarrow \mathbf{S}/\mathbf{M}_2$: there exists a signature scheme Σ such that Σ meets a security notion $\mathbf{S}$ in the model $\mathbf{M}_1$ while Σ doesn't meet $\mathbf{S}$ in the model $\mathbf{M}_2$.

It is clear from the definitions of the models that the following relations hold for any security notion $\mathbf{S}$ (see Section 3).

$$\mathbf{S}/\mathsf{ROM} \Leftarrow \mathbf{S}/\mathsf{CT\text{-}ROM} \Leftarrow \mathbf{S}/\mathsf{SPT\text{-}ROM} \Leftarrow \mathbf{S}/\mathsf{FPT\text{-}ROM}$$

From Table 1, under the RSA assumption we can show the separations for the security notion $\mathbf{S}'$: the existential unforgeability under the adaptive chosen message attack, that is, the following relations hold.

$$\mathbf{S}'/\mathsf{ROM} \nRightarrow \mathbf{S}'/\mathsf{CT\text{-}ROM} \nRightarrow \mathbf{S}'/\mathsf{SPT\text{-}ROM} \nRightarrow \mathbf{S}'/\mathsf{FPT\text{-}ROM}$$

Organization. In Section 2, we give some notation. Our models are presented in Section 3. We discuss the security of the schemes in Section 4. Finally, in Section 5, we make a few remarks on our models and schemes.

2 Preliminaries

2.1 Notation

If $\mathcal{D}$ is a distribution, $x \leftarrow \mathcal{D}$ denote that x is sampled according to $\mathcal{D}$, and let $f_{\mathcal{D}}(x)$ be the probability mass function of distribution $\mathcal{D}$. Let $\mathrm{B}(N, p)$ be the binomial distribution with N trials and success probability p.

Let S be a finite set. Let $s \leftarrow S$ denote that s is sampled from the uniform distribution on S. $\#S$ denotes the number of elements in S.

If $\mathcal{A}$ is a probabilistic machine and x is an input, let $\mathcal{A}(x)$ denote the output distribution of $\mathcal{A}$ on input x.

Let ϕ be a boolean function. Let $\Pr_s[d \leftarrow \mathcal{D} : \phi(s, d)]$ be the probability that $\phi(s, d)$ is true after sampling $s \leftarrow S$ and $d \leftarrow \mathcal{D}$.

Let "$\|$" denote concatenation, and $w_1 \parallel w_2 \overset{p}{\leftarrow} w$ that string w is parsed as w_1 and w_2.

Finally, for a table $\mathbb{T} = \{(x, y)\}$, we define $\mathbb{T}(y) = \{(\tilde{x}, \tilde{y}) \in \mathrm{T} \mid y = \tilde{y}\}$.

2.2 Digital Signature Schemes

We review a model of digital signature schemes.

Syntax. A digital signature scheme over message space $\mathcal{M}$ is defined by the following three algorithms.

- The key generation algorithm Gen. On input 1^k, where k is the security parameter, the algorithm produces a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.
- The signing algorithm Sig. Given a secret key sk and a message $\mathsf{m} \in \mathcal{M}$, the algorithm produces a signature σ on the message m.
- The verification algorithm Ver. Given a public key pk, a message m, and a signature σ, the algorithm outputs a bit τ. If $\tau = 1$ the signature is accepted with respect to pk and rejected otherwise.

We require that for all $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Gen}(1^k)$ and for all message $\mathsf{m} \in \mathcal{M}$, $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \mathsf{Sig}(\mathsf{sk}, \mathsf{m})) = 1$ should be satisfied.

In the rest of the paper we omit pk, sk and write $\mathsf{Ver}(\mathsf{m}, \sigma)$ as $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$, and $\mathsf{Sig}(\mathsf{m})$ as $\mathsf{Sig}(\mathsf{sk}, \mathsf{m})$ for short.

Security Notions. A widely accepted standard security notion was defined by Goldwasser, Micali and Rivest [13], as the existential unforgeability under the adaptive chosen message attack (EUF-CMA).

Definition 1. *A polynomial-time oracle query machine $\mathcal{A}$ is said to break the signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *if after making signing queries adaptively, it outputs, with non-negligible probability, a valid forgery that was never queried.*

Definition 2 (EUF-CMA)**.** *A signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is said to be secure if there is no polynomial-time oracle query machine that breaks the scheme.*

3 Our Models

We formalize the notion of weakened random oracle models that were mentioned by Liskov [10]. Each of our models provides a random oracle together with another oracle that breaks some property of the random oracle model. First we review the random oracle model, and then propose three models.

3.1 The Random Oracle Model (ROM)

Let X, Y be finite sets. The random oracle model has a hash function h chosen randomly from all of the functions from X to Y and the random oracle associated with h. A hash function h can be considered as a hash table $\mathbb{T}_h$ which defines the correspondence of the elements in X with the elements in Y. In this model, all of the parties (including the adversary) have access to the random oracle. When the hash value of x is queried, the random oracle answers the corresponding value y in $\mathbb{T}_h$.

In almost all the proofs employing the random oracles, the reduction algorithms simulate the random oracles with embedding the target problem instances. We consider how to simulate the random oracle except for the embedding. In a standard way, we simulate the random oracle maintaining a table $\mathbb{T}$ that is initially empty as follows. When the hash value of x is queried, if there is an entry $(\tilde{x}, \tilde{y}) \in \mathbb{T}$ such that $x = \tilde{x}$, then return $\tilde{y}$; otherwise pick uniformly $y \leftarrow Y$, insert (x, y) in the hash table $\mathbb{T}$, and return y.

Alternatively, we propose a different algorithm RO to simulate the random oracle. We manage a hash table $\mathbb{T}$ and a table $\mathbb{L}$ that are initially empty. The table $\mathbb{T}$ does the same role as above, whereas the table $\mathbb{L}$ manages the number of elements in X that map to $y \in Y$. For example, if there is $(y, n) \in \mathbb{L}$ then it is expected that there are exactly n elements in X that map to $y \in Y$. When we insert (x, y) in the hash table $\mathbb{T}$ such that y is not yet in $\mathbb{T}$, we also determine the number n of preimages of y and add (y, n) to the table $\mathbb{L}$.

Algorithm RO(x)*:*

1. If there is an entry $(\tilde{x}, \tilde{y}) \in \mathbb{T}$ such that $x = \tilde{x}$, then return $\tilde{y}$.
2. Compute the following value:

$$p = \frac{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))}{\#X - \#\mathbb{T}}.$$

   (p is the probability to answer $\tilde{y} \in Y$ that is not new, i.e. $(\tilde{x}, \tilde{y}) \in \mathbb{T}$ for some $\tilde{x}$.)
3. Flip a biased coin with probability $\Pr[\alpha = 0] = p$.
   (Decide whether the simulation returns a new value or not. "$\alpha = 0$" indicates "not new", and "$\alpha = 1$" indicates "new".)
4. If $\alpha = 0$,
   then pick y according to the following distribution, and go to Step 8.

$$y \leftarrow \mathcal{D},$$
$$\text{where } f_{\mathcal{D}}(y) = \frac{n - \#\mathbb{T}(y)}{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))} \text{ for } (y, n) \in \mathbb{L}.$$

```
5. If $\alpha \neq 0$,
       then pick uniformly $y \leftarrow Y \setminus \bigcup_{(\tilde{y},\tilde{n})\in\mathbb{L}}\{\tilde{y}\}$.
6.     Pick $n'$ according to the following binomial distribution:
```

$$n' \leftarrow \mathrm{B}(\#X - \sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} \tilde{n} - 1, \frac{1}{\#Y - \#\mathbb{L}}).$$

```
       ($n'$ is the number of preimages of $y$ excluding $(x, y)$.)
7.     Set $n = n' + 1$ and insert $(y, n)$ in $\mathbb{L}$.

8. Insert $(x, y)$ in $\mathbb{T}$, and return $y$.
```

Remark 1. In the rest of the paper, we denote by $\mathbb{T}_h$ the table in the ROM (CT-ROM, SPT-ROM, FPT-ROM), and denote by $\mathbb{T}$ and $\mathbb{L}$ the tables in the simulation.

In order to analyze this algorithm, we assume that we can efficiently sample from the binomial distribution $\mathrm{B}(N, p)$ perfectly. There are quite many papers on the efficient sampling from the binomial distribution [14]. However, we could neither find precise analysis of their methods nor analyze precisely by ourselves. Therefore, we have to employ the following assumption in the analyses of all of our simulations.

Assumption 1. *There is a polynomial-time machine $\mathcal{A}$ such that the distribution $\mathcal{A}(N, p)$ output by the algorithm $\mathcal{A}$ is equal to the binomial distribution* $\mathrm{B}(N, p)$*, where N is a positive integer and* $0 \leq p \leq 1$.

Lemma 1. *The simulation of the random oracle is perfect. That is, the distribution on the outputs of the random oracle is equal to the distribution on the outputs of Algorithm* RO.

Proof. We consider the probability that Algorithm RO replies y^* as the hash value of x^*. Fix the tables $\mathbb{T}$ and $\mathbb{L}$ at an arbitrary point according to Algorithm RO . Let $\mathbb{T}^*$ and $\mathbb{L}^*$ be the tables after replying y^* for the hash value of x^*.

First, we consider the case where y^* is not new (i.e. $(y^*, n^*) \in \mathbb{L}$). Note that in this case $\mathbb{L}^* = \mathbb{L}$. Let $\mathsf{old}(x^*, y^*)$ be the event $\mathbb{T}^* = \mathbb{T} \cup (x^*, y^*) \wedge \mathbb{L}^* = \mathbb{L}$.

According to our method, for any y^* that is not new, we have

$$\begin{aligned}\Pr[\mathsf{old}(x^*, y^*)] &= p \times \frac{n^* - \#\mathbb{T}(y^*)}{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))} \\ &= \frac{n^* - \#\mathbb{T}(y^*)}{\#X - \#\mathbb{T}}. \end{aligned} \tag{1}$$

Second, we consider the case where y^* is new (i.e. $(y^*, n^*) \notin \mathbb{L}$). Let $N_X = \#X - \sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} \tilde{n}$ and $N_Y = \#Y - \#\mathbb{L}$. The former represents the number of elements in X that are not to be assigned to some $\tilde{y}$ such that there is $(\tilde{y}, \tilde{n}) \in \mathbb{L}$, and the latter represents the number of elements in Y that are not defined in $\mathbb{L}$. Let $\mathsf{new}(x^*, y^*, n^*)$ be the event $\mathbb{T}^* = \mathbb{T} \cup (x^*, y^*) \wedge \mathbb{L}^* = \mathbb{L} \cup (y^*, n^*)$.

According to our method, for any y^* that is new and for any $n^* = n' + 1$ such that $0 \le n' \le N_X - 1$, we have

$$\begin{aligned}\Pr[\mathsf{new}(x^*, y^*, n^*)] &= (1-p) \times \frac{1}{N_Y} \times \binom{N_X - 1}{n'} (\frac{1}{N_Y})^{n'} (1 - \frac{1}{N_Y})^{N_X - 1 - n'} \\ &= (1-p) \frac{n'+1}{N_X} \binom{N_X}{n'+1} (\frac{1}{N_Y})^{n'+1} (1 - \frac{1}{N_Y})^{N_X - 1 - n'}.\end{aligned}$$

Notice here that $\#\mathbb{T} = \sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} \#\mathbb{T}(\tilde{y})$, and we have

$$\begin{aligned}1 - p &= 1 - \frac{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} (\tilde{n} - \#\mathbb{T}(\tilde{y}))}{\#X - \#\mathbb{T}} \\ &= \frac{N_X}{\#X - \#\mathbb{T}}.\end{aligned}$$

Therefore we have

$$\Pr[\mathsf{new}(x^*, y^*, n^*)] = \frac{n^*}{\#X - \#\mathbb{T}} \binom{N_X}{n^*} (\frac{1}{N_Y})^{n^*} (1 - \frac{1}{N_Y})^{N_X - n^*}. \quad (2)$$

Now let us consider what the probabilities given by Equations (1) and (2) imply. Both of the probabilities are equivalent to the probability that a hash function h chosen in the ROM satisfies $(x^*, y^*) \in \mathbb{T}_h$ and $\#\mathbb{T}_h(y^*) = n^*$ under the condition where $(x, y) \in \mathbb{T}_h$ for any $(x, y) \in \mathbb{T}$ and $\#\mathbb{T}_h(y) = n$ for any $(y, n) \in \mathbb{L}$.

Therefore the distribution on the outputs of the random oracle is equal to the distribution on the outputs of Algorithm RO. □

3.2 The Collision Tractable Random Oracle Model (CT-ROM)

Let X, Y be finite sets. The *collision tractable* random oracle model has the collision oracle that is used to find collisions, in addition to a hash function h chosen randomly from all of the functions from X to Y and the random oracle associated with h. In this model the adversary has access to the collision oracle.

When the hash value of x is queried, the random oracle answers the corresponding value y in $\mathbb{T}_h$. When a collision is queried, the collision oracle answers as follows. The collision oracle picks uniformly one entry $(x, y) \in \mathbb{T}_h$. If there is no other entry $(x', y) \in \mathbb{T}_h$, then answers $\perp$. Otherwise, it picks uniformly one entry $(x', y) \in \mathbb{T}_h$ satisfying $x \neq x'$ and answers (x, x').

For this model, in addition to Algorithm RO, we construct an algorithm CO. Algorithms RO and CO are used to simulate the random oracle and the collision oracle, respectively. Algorithm CO uses the tables $\mathbb{T}$ and $\mathbb{L}$ that are commonly used in Algorithm RO.

Algorithm CO()*:*

1. Pick uniformly $x \leftarrow X$.
2. In order to obtain the hash value $y = h(x)$, run Algorithm RO(x).

3. If $n = 1$ for $(y, n) \in \mathbb{L}$, then return $\perp$.
4. If $n \neq 1$ for $(y, n) \in \mathbb{L}$, then compute the following value:

$$q_{(y,n)} = \frac{\#\mathbb{T}(y) - 1}{n - 1}.$$

   ($q_{(y,n)}$ is the probability to answer $\tilde{x} \in X$ that is not new.)
5. Flip a biased coin with probability $\Pr[\beta = 0] = q_{(y,n)}$.
6. If $\beta = 0$, then pick uniformly one entry $(\tilde{x}, y) \in \mathbb{T}$ satisfying $x \neq \tilde{x}$ and return $(x, \tilde{x})$.
7. If $\beta \neq 0$, then pick uniformly $x' \leftarrow X$ such that there is no entry $(x', \tilde{y}) \in \mathbb{T}$ for any $\tilde{y} \in Y$.
8. Insert (x', y) in $\mathbb{T}$, and return (x, x').

Corollary 1. *The simulations of the random oracle and the collision oracle are perfect. That is, the distribution on the outputs of the random oracle and the collision oracle is equal to the distribution on the outputs of Algorithms* RO *and* CO.

Proof. From Lemma 1, the simulation of the random oracle is perfect. In this simulation, the table $\mathbb{L}$ indicates that the number of preimages of y. This implies that the simulation of the collision oracle are perfect. □

3.3 The Second-Preimage Tractable Random Oracle Model (SPT-ROM)

Let X, Y be finite sets. The *second-preimage tractable* random oracle model has the second-preimage oracle that is used to find second-preimages, in addition to a hash function h chosen randomly from all of the functions from X to Y and the random oracle associated with h. In this model the adversary has access to the second-preimage oracle.

When the hash value of x is queried, the random oracle answers the corresponding value y in $\mathbb{T}_h$. When a second-preimage of (x, y) is queried, the second-preimage oracle answers as follows. If it has not answered that h maps x to y, it answers $\perp$. If there is only one entry $(\tilde{x}, \tilde{y}) \in \mathbb{T}_h$ such that $y = \tilde{y}$, then it answers $\perp$. Otherwise, it answers uniformly one x' such that $(x', y) \in \mathbb{T}_h$ satisfying $x' \neq x$.

For this model, in addition to Algorithm RO, we construct an algorithm SPO. Algorithms RO and SPO are used to simulate the random oracle and the second-preimage oracle, respectively. Algorithm SPO uses the tables $\mathbb{T}$ and $\mathbb{L}$ that are commonly used in Algorithm RO.

Algorithm SPO(x, y)*:*

1. If $(x, y) \notin \mathbb{T}$, then return $\perp$.
2. If $n = 1$ for $(y, n) \in \mathbb{L}$, then return $\perp$.
3. If $n \neq 1$ for $(y, n) \in \mathbb{L}$, then compute the following value:

$$q_{(y,n)} = \frac{\#\mathbb{T}(y) - 1}{n - 1}.$$

4. Flip a biased coin with probability $\Pr[\beta = 0] = q_{(y,n)}$.

5. If $\beta = 0$, then pick uniformly one entry $(\tilde{x}, y) \in \mathbb{T}$ satisfying $x \neq \tilde{x}$ and return $\tilde{x}$.
6. If $\beta \neq 0$, then pick uniformly $x' \leftarrow X$ such that there is no entry $(x', \tilde{y}) \in \mathbb{T}$ for any $\tilde{y} \in Y$.
7. Insert (x', y) in $\mathbb{T}$, and return x'.

Corollary 2. *The simulations of the random oracle and the second-preimage oracle are perfect. That is, the distribution on the outputs of the random oracle and the second-preimage oracle is equal to the distribution on the outputs of Algorithms* RO *and* SPO.

3.4 The First-Preimage Tractable Random Oracle Model (FPT-ROM)

Let X, Y be finite sets. The *first-preimage tractable* random oracle model has the first-preimage oracle that is used to find first-preimages, in addition to a hash function h chosen randomly from all of the functions from X to Y and the random oracle associated with h. In this model the adversary has access to the first-preimage oracle.

When the hash value of x is queried, the random oracle answers the corresponding value y in $\mathbb{T}_h$. When a first-preimage of y is queried, the first-preimage oracle answers as follows. If there is no $(\tilde{x}, \tilde{y}) \in \mathbb{T}_h$ such that $y = \tilde{y}$, then answers $\perp$. Otherwise it answers uniformly one $\tilde{x}$ such that $(\tilde{x}, \tilde{y}) \in \mathbb{T}_h$ satisfying $y = \tilde{y}$.

For this model, in addition to Algorithm RO, we construct an algorithm FPO. Algorithms RO and FPO are used to simulate the random oracle and the first-preimage oracle, respectively. Algorithm FPO uses the tables $\mathbb{T}$ and $\mathbb{L}$ that are commonly used in Algorithm RO.

Algorithm FPO(y)*:*

1. If there is no entry $(y, \tilde{n}) \in \mathbb{L}$ then pick n according to the binomial distribution:

$$n \leftarrow \mathrm{B}(\#X - \sum_{(\tilde{y},\tilde{n}) \in \mathbb{L}} \tilde{n}, \frac{1}{\#Y - \#\mathbb{L}}).$$

2. Insert (y, n) in $\mathbb{L}$.
3. If $n = 0$ for $(y, n) \in \mathbb{L}$, then return $\perp$.
4. If $n \neq 0$ for $(y, n) \in \mathbb{L}$, then compute the following value:

$$q_{(y,n)} = \frac{\#\mathbb{T}(y)}{n}.$$

5. Flip a biased coin with probability $\Pr[\beta = 0] = q_{(y,n)}$.
6. If $\beta = 0$, then pick uniformly one entry $(\tilde{x}, y) \in \mathbb{T}$ and return $\tilde{x}$.
7. If $\beta \neq 0$, then pick uniformly $x \leftarrow X$ such that there is no entry $(x, \tilde{y}) \in \mathbb{T}$ for any $\tilde{y} \in Y$.
8. Insert (x, y) in $\mathbb{T}$, and return x.

Corollary 3. *The simulations of the random oracle and the first-preimage oracle are perfect. That is, the distribution on the outputs of the random oracle and the first-preimage oracle is equal to the distribution on the outputs of Algorithms* RO *and* FPO.

Proof. From Lemma 1 the simulation of the random oracle is perfect. Now let us consider the case where the table $\mathbb{L}$ is updated in Algorithm FPO. In the following, we use the same notation as in Lemma 1. The number n^* is defined according to the binomial distribution described in Steps 1 and 2. The probability that the table $\mathbb{L}$ is updated to be $\mathbb{L}^*$ such that $\mathbb{L}^* = \mathbb{L} \cup (y^*, n^*)$ in Steps 1 and 2 is equal to the probability that a hash function h chosen in the FPT-ROM satisfies $\#\mathbb{T}_h(y^*) = n^*$ under the condition that $(x, y) \in \mathbb{T}_h$ for any $(x, y) \in \mathbb{T}$ and $\#\mathbb{T}_h(y) = n$ for any $(y, n) \in \mathbb{L}$. In Algorithm FPO, the table $\mathbb{L}$ correctly indicates the number of the preimages of y. This implies that the simulations of the random oracle and the first-preimage oracle are perfect. □

4 Security of Signature Schemes

In this section, we consider the security of RSA-FDH [3] and RSA-PFDH [12] in four variants of the random oracle models. We also propose new signature schemes called RSA-PFDH$^+$ and RSA-PFDH$^\oplus$, and consider the security in four variants of the random oracle models.

We review the RSA assumption on which the security of four schemes are based.

Definition 3 (The RSA Generator). *The RSA generator* RSA*, which on input 1^k, randomly choose distinct $k/2$-bit primes p, q and computes the RSA modulus $N = pq$. It randomly picks $e \leftarrow \mathbb{Z}_{\phi(N)}$ and computes d such that $ed = 1 \bmod \phi(N)$, where $\phi(\cdot)$ is Euler's totient function. Finally the RSA generator* RSA *outputs (N, e, d).*

Assumption 2 (The RSA Assumption). *A polynomial-time machine $\mathcal{A}$ is said to solve the RSA problem if given an RSA challenge (N, e, z) where N, e is generated by* RSA(1^k) *and $z \leftarrow \mathbb{Z}_N^*$, it outputs $z^{1/e} \bmod N$ with non-negligible probability.*

The RSA assumption is that there is no polynomial-time machine that solves the RSA problem.

4.1 RSA-FDH

In this section, we show that RSA-FDH [3] is secure in the ROM, but not secure in the CT-ROM.

The Scheme. We review RSA-FDH [3] .
Let $\mathcal{M} = \{0, 1\}^l$ be the message space and h a hash function such as

$$h : \{0, 1\}^l \to \{0, 1\}^k.$$

Then RSA-FDH is described as follows.

Gen(1^k)	Sig(m)	Ver(m, σ)

$(N, e, d) \leftarrow$ RSA(1^k)
pk $\leftarrow (N, e)$
sk $\leftarrow (N, d)$
return (pk, sk)
	$y \leftarrow h(\mathsf{m})$
	$\sigma \leftarrow y^d \bmod N$
	return σ
		$y \leftarrow \sigma^e \bmod N$
		if $h(\mathsf{m}) = y$
		  return 1
		else
		  return 0

The Security. RSA-FDH is secure in the ROM. More precisely the following proposition was proved [3,15]. We omit the proof, see [3,15] for details.

Proposition 1. *In the* ROM*, if the RSA assumption holds, there is no polynomial-time oracle query machine that breaks* RSA-FDH *by making queries to the signing oracle and the random oracle for h.*

We show that RSA-FDH is insecure in the CT-ROM.

Theorem 1. *In the* CT-ROM*, there exists a polynomial-time oracle query machine $\mathcal{A}$ that breaks* RSA-FDH *by making queries to the signing oracle and the collision oracle for h with probability at least* $1 - e^{-(2^l-1)/2^k}$.

Proof. We construct an algorithm $\mathcal{A}$ as follows.

1. Query to the collision oracle, and obtain ξ.
2. If $\xi = \perp$ then abort, otherwise $(\mathsf{m}_1, \mathsf{m}_2) \overset{p}{\leftarrow} \xi$, where $h(\mathsf{m}_1) = h(\mathsf{m}_2)$.
3. Query the signature of m_1 to the signing oracle, and obtain a signature σ.
4. Output (m_2, σ) as a valid forgery.

If $\mathcal{A}$ does not abort, then $\mathcal{A}$ can output a valid forgery. Therefore it is sufficient to bound the probability that $\mathcal{A}$ aborts (abort). In the following we use the same notation as in Section 3. Let $X = \{0,1\}^l$, $Y = \{0,1\}^k$, and $N = \#X$, $p = \frac{1}{\#Y}$. Then we have

$$\begin{aligned}\Pr[\mathsf{abort}] &= \Pr[\xi = \perp] \\ &= \Pr_{x,h}[\#\mathbb{T}_h(y) \leq 1 \text{ for } (x,y) \in \mathbb{T}_h] \\ &= \Pr[n' \leftarrow \mathrm{B}(N-1, p) : n' = 0] \\ &= (1-p)^{N-1} \\ &\leq e^{-p(N-1)}.\end{aligned}$$

For example, in the case of $\#X = \#Y \geq 2$, we can bound this value as

$$\Pr[\mathsf{abort}] \leq e^{-(1-p)} \leq e^{-1/2}.$$

Therefore $\mathcal{A}$ can output a valid forgery with probability at least $1 - e^{-1/2}$. □

4.2 RSA-PFDH

In this section, we show that RSA-PFDH [12] is secure in the CT-ROM, but not secure in the SPT-ROM.

The Scheme. We review RSA-PFDH [12].
Let $\mathcal{M} = \{0,1\}^l$ be the message space and h hash function such as

$$h : \{0,1\}^{l+k_1} \to \{0,1\}^k.$$

Then RSA-PFDH is described as follows.

$\mathsf{Gen}(1^k)$	$\mathsf{Sig}(\mathsf{m})$	$\mathsf{Ver}(\mathsf{m}, \sigma)$

$(N, e, d) \leftarrow \mathsf{RSA}(1^k)$
$\mathsf{pk} \leftarrow (N, e)$
$\mathsf{sk} \leftarrow (N, d)$
return $(\mathsf{pk}, \mathsf{sk})$
	$r \leftarrow \{0, 1\}^{k_1}$
	$y \leftarrow h(\mathsf{m} \parallel r)$
	$x \leftarrow y^d \bmod N$
	$\sigma \leftarrow (r, x)$
	return σ
		$(r, x) \overset{p}{\leftarrow} \sigma$
		$y \leftarrow x^e \bmod N$
		if $h(\mathsf{m} \parallel r) = y$
		   return 1
		else
		   return 0

The Security. We show RSA-PFDH is secure in the CT-ROM. Intuitively, in order to break RSA-PFDH in a straightforward way, it would be necessary to obtain a collision $\mathsf{m} \parallel r, \mathsf{m}' \parallel r'$ such that $h(\mathsf{m} \parallel r) = h(\mathsf{m}' \parallel r')$ and the signature of m. However the randomness in the signature makes it difficult to make use of collisions, which are also randomly provided by the collision oracle.

Theorem 2. *In the* CT-ROM, *for all polynomial-time oracle query machines that break* RSA-PFDH *with probability $\epsilon_{\rm euf}$ by making q_s, q_h, and q_h^c queries to the signing oracle, the random oracle for h, and the collision oracle for h, respectively, there exists a probabilistic machine that solves the RSA problem with probability $\epsilon_{\rm rsa}$ such that*

$$\epsilon_{\rm euf} \le \epsilon_{\rm rsa} + \frac{1}{2^k - Q_1} + \frac{(Q_1)^2}{2^k} + \frac{q_s Q_2}{2^{k_1}} + (1 - \frac{Q_2}{2^{l+k_1}})^{-1} \frac{(Q_1)^2}{2^k - Q_1},$$

where $Q_1 = q_s + q_h + q_h^c + 1$ and $Q_2 = q_s + q_h + 2q_h^c + 1$.

Proof. (*Sketch*) We start with the original attack game with respect to EUF-CMA in the CT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which solves the RSA problem. Let (N, e, y) be the RSA challenge. Let $\mathrm{dist}(i, j)$ be the difference between the probability that the adversary outputs a valid forgery in the $\mathbf{Game}_i$ and that in the $\mathbf{Game}_j$.

- $\mathbf{Game}_0$: The original attack game with respect to EUF-CMA in the CT-ROM.
- $\mathbf{Game}_1$: We replace the random oracle and the collision oracle with Algorithms RO and CO in Section 3, respectively. Let us denote by $\mathbb{T}$ and $\mathbb{L}$ the tables commonly used in Algorithms RO and CO. Then, we have

$$\mathrm{dist}(0, 1) = 0.$$

- $\mathbf{Game}_2$: We remove Steps 2–4 in Algorithm RO, and set $\alpha = 1$ (i.e. Algorithm RO always answers a new value). Then, we have

$$\mathrm{dist}(1, 2) \le (1 - \frac{Q_2}{2^{l+k_1}})^{-1} \frac{(Q_1)^2}{2^k - Q_1}.$$

- $\mathbf{Game}_3$: When the signing algorithm runs Algorithm RO on input $\mathsf{m} \parallel r$, if $(\mathsf{m} \parallel r, \tilde{y})$ is already in the table $\mathbb{T}$ for some $\tilde{y}$, then Algorithm RO aborts. Then, we have

$$\mathrm{dist}(2, 3) \le \frac{q_s Q_2}{2^{k_1}}.$$

- **Game$_4$:** Instead of randomly choosing $y \in \mathbb{Z}_N$ and setting $h(\mathsf{m} \parallel r) = y$, Algorithm RO chooses y as follows.
  - If the hash value is queried by the signing algorithm,
    1. then, Algorithm RO randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow x^e \bmod N$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{r}, y)$ is already in the table $\mathbb{T}$ for some $\tilde{\mathsf{m}}, \tilde{r}$, then Algorithm RO aborts.
  - If the hash value is queried by the adversary,
    1. then, Algorithm RO randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow zx^e \bmod N$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{r}, y)$ is already in the table $\mathbb{T}$ for some $\tilde{\mathsf{m}}, \tilde{r}$, then Algorithm RO aborts.

  Then, we have
$$\mathrm{dist}(3,4) \le \frac{(Q_1)^2}{2^k}.$$
- **Game$_5$:** We modify the signing algorithm in the computation y^d to search (x, y) such that $x^e = y$, instead of using the secret key d. Then, we have
$$\mathrm{dist}(4,5) = 0.$$

If the adversary outputs a valid forgery (m^*, σ^*) in the last game, it satisfies the equation $h(\mathsf{m}^* \parallel r^*) = y = (x^*)^e \bmod N$ where $(r^*, x^*) \overset{p}{\leftarrow} \sigma^*$. In order to satisfy the equation, the adversary must have queried the hash value of $\mathsf{m}^* \parallel r^*$ with probability at least $1 - \frac{1}{2^k - Q_1}$, and then we know the value x such that $y = zx^e \bmod N$. We can invert the RSA challenge z by computing $z^{1/e} = x^*/x^{-1} \bmod N$. □

Next, we show that $\mathsf{RSA\text{-}PFDH}$ is insecure in the $\mathsf{SPT\text{-}ROM}$.

Theorem 3. *In the* $\mathsf{SPT\text{-}ROM}$*, there exists a polynomial-time oracle query machine* $\mathcal{A}$ *that breaks* $\mathsf{RSA\text{-}PFDH}$ *by making queries to the signing oracle, the random oracle for h, and the second-preimage oracle for h, with probability at least* $1 - e^{-(2^{l+k_1}-1)/2^k} - \frac{1}{2^l}$.

Proof. We construct an algorithm $\mathcal{A}$ as follows.

1. Query the signature of m to the signing oracle, and obtain a signature σ.
2. $(r, x) \overset{p}{\leftarrow} \sigma$.
3. Query the hash value of $\mathsf{m} \parallel r$ to the random oracle for h, and obtain $y = h(\mathsf{m} \parallel r)$.
4. Query the second-preimage of $(\mathsf{m} \parallel r, y)$ to the second-preimage oracle, and obtain ξ.
5. If $\xi = \perp$ then abort_1, otherwise $\mathsf{m}' \parallel r' \overset{p}{\leftarrow} \xi$, where $h(\mathsf{m} \parallel r) = h(\mathsf{m}' \parallel r')$.
6. If $\mathsf{m}' = \mathsf{m}$ then abort_2, otherwise $\sigma' \leftarrow (r', x)$.
7. Output (m', σ') as a valid forgery.

If $\mathcal{A}$ does not abort, then $\mathcal{A}$ can output a valid forgery. Therefore it is sufficient to bound the probability that $\mathcal{A}$ aborts. In the following we use the same notation as in Section 3. Let $X = \{0,1\}^{l+k_1}$, $Y = \{0,1\}^k$, and $N = \#X$, $p = \frac{1}{\#Y}$. Then, we have
$$\Pr[\mathsf{abort}] \le \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2].$$

The first probability is evaluated in a similar way as in Theorem 1. We have

$$\mathsf{Pr}[\mathsf{abort}_1] = (1-p)^{N-1} \leq e^{-p(N-1)}.$$

The second probability is bounded as

$$\mathsf{Pr}[\mathsf{abort}_2] = \mathsf{Pr}[\mathsf{m} = \mathsf{m}'] \leq \frac{1}{2^l}.$$

Thus, we have

$$\mathsf{Pr}[\mathsf{abort}] \leq e^{-p(N-1)} + \frac{1}{2^l}.$$

For example, in the case of $\#X = \#Y \geq 2$, we can bound this value as

$$\mathsf{Pr}[\mathsf{abort}] = e^{-(1-p)} + \frac{1}{2^l} \leq e^{-1/2} + \frac{1}{2^l}.$$

Therefore, $\mathcal{A}$ can output a valid forgery with probability at least $1 - e^{-1/2} - \frac{1}{2^l}$. □

4.3 RSA-PFDH$^+$

In this section, we propose RSA-PFDH$^+$, and show that RSA-PFDH$^+$ is secure in the SPT-ROM, but not secure in the FPT-ROM.

The Scheme. We construct RSA-PFDH$^+$.
Let $\mathcal{M} = \{0,1\}^l$ be the message space and g, h hash functions such that

$$g : \{0,1\}^{k_1} \to \{0,1\}^{k_1},\ h : \{0,1\}^{l+k_1} \to \{0,1\}^k.$$

Then RSA-PFDH$^+$ is described as follows.

$\mathsf{Gen}(1^k)$	$\mathsf{Sig}(\mathsf{m})$	$\mathsf{Ver}(\mathsf{m}, \sigma)$
$(N, e, d) \leftarrow \mathsf{RSA}(1^k)$	$r \leftarrow \{0,1\}^{k_1}$	$(r, x) \overset{p}{\leftarrow} \sigma$
$\mathsf{pk} \leftarrow (N, e)$	$s \leftarrow g(r)$	$y \leftarrow x^e \bmod N$
$\mathsf{sk} \leftarrow (N, d)$	$y \leftarrow h(\mathsf{m} \parallel s)$	$s \leftarrow g(r)$
return (pk, sk)	$x \leftarrow y^d \bmod N$	if $h(\mathsf{m} \parallel s) = y$
	$\sigma \leftarrow (r, x)$	return 1
	return σ	else
		return 0

The Security. We show RSA-PFDH$^+$ is secure in the SPT-ROM. Intuitively, the adversary similar to that described in Theorem 3 does not work well. The reason is as follows. The adversary queries the signature of m, and obtain $\sigma = (r, x)$. For $s = g(r), y = h(\mathsf{m} \parallel s)$, the adversary then queries the second-preimage of y to the second-preimage oracle for h, and obtain $\mathsf{m}' \parallel s'$. However the adversary would not know a preimage of s', and would not obtain r' such that $s' = g(r')$. Therefore this straightforward way does not work.

Theorem 4. *In the* SPT-ROM, *for all polynomial-time oracle query machines that break* RSA-PFDH$^+$ *with probability* ϵ_{euf} *by making* q_s, q_g, q_h, *and* q_g^{sp}, q_h^{sp} *queries to* *the signing oracle, the random oracles for g, h, and the second-preimage oracles for g, h, respectively, there exists a probabilistic machine that solves the RSA problem with probability ϵ_{rsa} such that*

$$\epsilon_{\text{euf}} \leq \epsilon_{\text{rsa}} + \frac{1}{2^k - (q_s + q_h)} + \frac{(q_s + q_h)^2}{2^k} + \frac{q_s Q_h}{2^{k_1}} + \frac{q_g^{sp}(q_s + q_g)}{2^{l+k_1} - Q_h}$$
$$+ \frac{(q_s + q_g)Q_h}{2^{k_1} - (q_s + q_g)} + (1 - \frac{Q_h}{2^{l+k_1}})^{-1} \frac{(Q_h)^2}{2^k - (q_s + q_h)} + (1 - \frac{Q_g}{2^{k_1}})^{-1} \frac{(Q_g)^2}{2^{k_1} - (q_s + q_g)},$$

where $Q_h = q_s + q_h + q_h^{sp} + 1, Q_g = q_s + q_g + q_g^{sp} + 1$.

Proof. (Sketch) We start with the original attack game with respect to EUF-CMA in the SPT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which solves RSA problem. Let (N, e, y) be the RSA challenge.

- **Game$_0$:** The original attack game with respect to EUF-CMA in the SPT-ROM.
- **Game$_1$:** We replace the random oracles for g and h with Algorithms RO_g and RO_h, and also replace the second-preimage oracles for g and h with Algorithms SPO_g and SPO_h in Section 3, respectively. Let us denote by $\mathbb{T}_1$ and $\mathbb{L}_1$ the tables commonly used in Algorithms RO_g and SPO_g, and also denote by $\mathbb{T}_2$ and $\mathbb{L}_2$ the tables commonly used in Algorithms RO_h and SPO_h.
- **Game$_2$:** We remove Steps 2–4 in Algorithms RO_g and RO_h, and set $\alpha = 1$ (i.e. Algorithms RO_g and RO_h always answer a new value).
- **Game$_3$:** In Algorithm RO_g at Step 5 (i.e. $s \leftarrow \{0,1\}^{k_1} \setminus \bigcup_{(\tilde{s},\tilde{n})\in\mathbb{L}_1}\{\tilde{s}\}$), if $(\tilde{\mathsf{m}} \parallel s, \tilde{y})$ is already in the table $\mathbb{T}_2$ for some $\tilde{r}$, then Algorithm RO_g aborts.
- **Game$_4$:** In Algorithm SPO_h at Step 6 (i.e. $\mathsf{m}' \parallel s' \leftarrow \{0,1\}^{l+k_1} \setminus \bigcup_{(\tilde{\mathsf{m}}\parallel\tilde{s},\tilde{y})\in\mathbb{T}_2}\{\tilde{\mathsf{m}} \parallel \tilde{s}\}$), if $(\tilde{r}, s')$ is already in the table $\mathbb{T}_1$ for some $\tilde{r}$, then Algorithm SPO_h aborts.
- **Game$_5$:** When the signing algorithm runs Algorithm RO_h on input $\mathsf{m} \parallel s$, if $(\mathsf{m} \parallel s, \tilde{y})$ is already in the table $\mathbb{T}_2$, then Algorithm RO_h aborts.
- **Game$_6$:** Instead of randomly choosing $y \in \mathbb{Z}_N$ and setting $h(\mathsf{m} \parallel s) = y$, Algorithm RO_h chooses y as follows.
  - If the hash value is queried by the signing algorithm,
    1. then, Algorithm RO_h randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow x^e \bmod N$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{s}, y)$ is already in the table $\mathbb{T}_2$ for some $\tilde{\mathsf{m}}, \tilde{s}$, then Algorithm RO_h aborts.
  - If the hash value is queried by the adversary,
    1. then, Algorithm RO_h randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow zx^e \bmod N$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{s}, y)$ is already in the table $\mathbb{T}_2$ for some $\tilde{\mathsf{m}}, \tilde{s}$, then Algorithm RO_h aborts.
- **Game$_7$:** We modify the signing algorithm in the computation y^d to search (x, y) such that $x^e = y$, instead of using the secret key d. Then, we have

If the adversary outputs a valid forgery (m^*, σ^*) in the last game, it satisfies the equation $h(\mathsf{m}^* \parallel s^*) = y = (x^*)^e \bmod N$ where $(r^*, x^*) \overset{p}{\leftarrow} \sigma^*, s^* = g(r^*)$. In order to satisfy the equation, the adversary must have queried the hash value of $\mathsf{m}^* \parallel s^*$, and then we know the value x such that $y = zx^e \bmod N$. We can invert the RSA challenge z by computing $z^{1/e} = x^*/x^{-1} \bmod N$. □

Next, we show that RSA-PFDH$^+$ is insecure in the FPT-ROM.

Theorem 5. *In the* FPT-ROM*, there exists a polynomial-time oracle query machine $\mathcal{A}$ that breaks* RSA-PFDH$^+$ *by making queries to the signing oracle, the random oracles for g, h, and the first-preimage oracles for g, h, with probability at least* $1 - e^{-(2^{l+k_1}-1)/2^k} - e^{-(1-\frac{1}{2^{k_1}})} - \frac{1}{2^l} - \frac{1}{2^{k_1}}$.

Proof. We construct an algorithm $\mathcal{A}$ as follows.

```
1. Query the signature of $\mathsf{m}$ to the signing oracle, and obtain a signature $\sigma$.
2. $(r, x) \overset{p}{\leftarrow} \sigma$.
3. Query the hash value of $r$ to the random oracle for $g$, and obtain $s = g(r)$.
4. Query the hash value of $\mathsf{m} \| s$ to the random oracle for $h$, and obtain $y = h(\mathsf{m} \| s)$.
5. Query the first-preimage of $y$ to the first-preimage oracle for $h$, and obtain $\xi$.
6. $\mathsf{m}' \| s' \overset{p}{\leftarrow} \xi$, where $h(\mathsf{m}' \| s') = h(\mathsf{m} \| s)$.
7. If $\mathsf{m}' \| s' = \mathsf{m} \| s$ then $\mathsf{abort}_1$.
8. If $\mathsf{m} = \mathsf{m}'$ then $\mathsf{abort}_2$.
9. Query the first-preimage of $s'$ to the first-preimage oracle for $g$, and obtain $\eta$.
10. If $\eta = \perp$ then $\mathsf{abort}_3$, otherwise $r' \overset{p}{\leftarrow} \eta$.
11. $\sigma' \leftarrow (r', x)$.
12. Output $(\mathsf{m}', \sigma')$ as a valid forgery.
```

If $\mathcal{A}$ does not abort, then $\mathcal{A}$ can output a valid forgery. Therefore it is sufficient to bound the probability that $\mathcal{A}$ aborts. In the following we use the same notation as in Section 3. Let $X = \{0,1\}^{l+k_1}$, $Y = \{0,1\}^k$, $R = \{0,1\}^{k_1}$, $S = \{0,1\}^{k_1}$, $g : R \to S$, and $N_1 = \#X$, $p_1 = \frac{1}{\#Y}$ $N_2 = \#R$, $p_2 = \frac{1}{\#S}$. Then, we have

$$\Pr[\mathsf{abort}] \leq \Pr[\mathsf{abort}_1] + \Pr[\mathsf{abort}_2] + \Pr[\mathsf{abort}_3].$$

The first probability is evaluated in the a similar way as in Theorem 1. We have

$$\Pr[\mathsf{abort}_1] = (1 - p_1)^{N_1 - 1} \leq e^{-p_1(N_1-1)}.$$

The second probability is bounded as

$$\Pr[\mathsf{abort}_2] = \Pr[\mathsf{m} = \mathsf{m}'] \leq \frac{1}{2^l}.$$

Next, we evaluate the third probability as

$$\begin{aligned}\Pr[\mathsf{abort}_3] &= \Pr[\eta = \perp] \\ &= \Pr_{g,r,s'}[(r,s) \in \mathbb{T}_g \wedge \#\mathbb{T}_g(s') = 0] \\ &= \Pr_{g,r,s'}[(r,s) \in \mathbb{T}_g \wedge \#\mathbb{T}_g(s') = 0 \wedge s = s'] \\ &\quad + \Pr_{g,r,s'}[(r,s) \in \mathbb{T}_g \wedge \#\mathbb{T}_g(s') = 0 \wedge s \neq s'].\end{aligned}$$

The first probability is bounded as

$$\Pr[s = s'] \leq \frac{1}{2^{k_1}}.$$

The second probability is bounded as

$$\begin{aligned}\Pr_{g,r,s'}[(r,s)\in\mathbb{T}_g \wedge \#\mathbb{T}_g(s')=0 \mid s\neq s'] &= \Pr[n\leftarrow \mathrm{B}(N_2-1,p_2) : n=0]\\ &= (1-p_2)^{N_2-1}\\ &< e^{-p_2(N_2-1)}\\ &= e^{-(1-\frac{1}{2^{k_1}})}.\end{aligned}$$

Thus, we have

$$\Pr[\mathsf{abort}] < e^{-p_1(N_1-1)} + \frac{1}{2^l} + \frac{1}{2^{k_1}} + e^{-(1-\frac{1}{2^{k_1}})}.$$

For example, in the case of $\#X = \#Y \geq 2$, we can bound this value as

$$\begin{aligned}\Pr[\mathsf{abort}] &< e^{-(1-p_1)} + \frac{1}{2^l} + \frac{1}{2^{k_1}} + e^{-(1-\frac{1}{2^{k_1}})}\\ &< e^{-1/2} + e^{-(1-\frac{1}{2^{k_1}})} + \frac{1}{2^l} + \frac{1}{2^{k_1}}.\end{aligned}$$

Therefore, $\mathcal{A}$ can output a valid forgery with probability at least $1 - e^{-1/2} - e^{-(1-\frac{1}{2^{k_1}})} - \frac{1}{2^l} - \frac{1}{2^{k_1}}$. □

4.4 RSA-PFDH$^{\oplus}$

In this section, we propose RSA-PFDH$^{\oplus}$, and show that RSA-PFDH$^{\oplus}$ is secure in the FPT-ROM.

The Scheme. We construct RSA-PFDH$^{\oplus}$.
Let $\mathcal{M} = \{0,1\}^l$ be the message space and h a hash function such that

$$h : \{0,1\}^{l+k} \to \{0,1\}^k.$$

Then RSA-PFDH$^{\oplus}$ is described as follows.

Gen(1^k)	Sig(m)	Ver(m, σ)
$(N,e,d) \leftarrow \mathsf{RSA}(1^k)$	$r \leftarrow \{0,1\}^{k_1}$	$(r,x) \overset{p}{\leftarrow} \sigma$
pk $\leftarrow (N,e)$	$w \leftarrow h(\mathsf{m} \parallel r)$	$y \leftarrow x^e \bmod N$
sk $\leftarrow (N,d)$	$y \leftarrow w \oplus r$	$w \leftarrow h(\mathsf{m} \parallel r)$
return (pk, sk)	$x \leftarrow y^d \bmod N$	if $w \oplus r = y$
	$\sigma \leftarrow (r,x)$	return 1
	return σ	else
		return 0

The Security. We show RSA-PFDH$^{\oplus}$ is secure in the FPT-ROM. Intuitively, the adversary similar to that described in Theorem 5 does not work well. The reason is as follows.

The adversary queries the signature of m, and obtain $\sigma = (r, x)$. For $w = h(m \parallel r)$, the adversary queries the first-preimage of w to the first-preimage oracle, and obtain $\mathsf{m}' \parallel r'$. However, r' would not equal to r. Therefore this straightforward way does not work.

Theorem 6. *In the* FPT-ROM*, for all polynomial-time oracle query machines that break* RSA-PFDH$^+$ *with probability ϵ_{euf} by making q_s, q_h, and q_h^{fp} queries to the signing oracle, the random oracle for h, and the first-preimage oracle for h, respectively, there exists a probabilistic machine that solves the RSA problem with probability ϵ_{rsa} such that*

$$\epsilon_{\mathrm{euf}} \leq \epsilon_{\mathrm{rsa}} + \frac{1}{2^k - Q} + \frac{Q^2}{2^{k-1}} + \frac{q_s Q}{2^k} + (1 - \frac{Q}{2^{l+k}})^{-1} \frac{Q^2}{2^k - Q},$$

where $Q = q_s + q_h + q_h^{fp} + 1$.

Proof. (Sketch) We start with the original attack game with respect to EUF-CMA in the FPT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which solves RSA problem. Let (N, e, y) be the RSA challenge.

- **Game$_0$:** The original attack game with respect to EUF-CMA in the FPT-ROM.
- **Game$_1$:** We replace the random oracle and the first-preimage oracle with Algorithms RO and FPO in Section 3, respectively. Let us denote by $\mathbb{T}$ and $\mathbb{L}$ the tables commonly used in Algorithms RO and FPO.
- **Game$_2$:** We remove Steps 2–4 in Algorithm RO, and set $\alpha = 1$ (i.e. Algorithm RO always answers a new value).
- **Game$_3$:** When the signing algorithm runs Algorithm RO on input $\mathsf{m} \parallel r$, if $(\mathsf{m} \parallel r, \tilde{w})$ is already in the table $\mathbb{T}$ for some $\tilde{w}$, then Algorithm RO aborts.
- **Game$_4$:** Instead of randomly choosing $w \in \mathbb{Z}_N$ and setting $h(\mathsf{m} \parallel r) = w$, Algorithm RO chooses w as follows.
  - If the hash value of is queried by the signing algorithm,
    1. then, Algorithm RO randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow x^e \bmod N$. Then Algorithm RO sets $w = y \oplus r$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{r}, w)$ is already in the table $\mathbb{T}$ for some $\tilde{\mathsf{m}}, \tilde{r}$, then Algorithm RO aborts.
  - If the hash value is queried by the adversary,
    1. then, Algorithm RO randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow zx^e \bmod N$. Then Algorithm RO sets $w = y \oplus r$.
    2. If $(\tilde{\mathsf{m}} \parallel \tilde{r}, w)$ is already in the table $\mathbb{T}$ for some $\tilde{\mathsf{m}}, \tilde{r}$, then Algorithm RO aborts.
- **Game$_5$:** Instead of randomly choosing $\mathsf{m} \parallel r \in X$ and setting $h(\mathsf{m} \parallel r) = w$, Algorithm FPO chooses $\mathsf{m} \parallel r$ as follows.
  - If a first-preimage is queried by the adversary,
    1. then, Algorithm FPO randomly chooses $x \in \mathbb{Z}_N$ and $y \leftarrow zx^e \bmod N$. Then Algorithm FPO sets $r = y \oplus w$ and randomly chooses m.
    2. If $(\tilde{\mathsf{m}} \parallel r, \tilde{w})$ is already in the table $\mathbb{T}$ for some $\tilde{\mathsf{m}}, \tilde{w}$, then Algorithm FPO aborts.
- **Game$_6$:** We modify the signing algorithm in the computation y^d to search (x, y) such that $x^e = y$, instead of using the secret key d.

If the adversary outputs a valid forgery (m^*, σ^*), then it satisfies the equation $h(\mathsf{m}^* \parallel r^*) \oplus r^* = y = (x^*)^e \bmod N$ where $(r^*, x^*) \overset{p}{\leftarrow} \sigma^*$. In order to satisfy the equation, the adversary must have queried the hash value of $\mathsf{m}^* \parallel r^*$, and then we know the value x such that $y = zx^e \bmod N$. We can invert the RSA challenge z by computing $z^{1/e} = x^*/x^{-1} \bmod N$. □

5 Concluding Remarks

In this paper, by applying Liskov's idea, we have proposed the weakened random oracle models, i.e. the CT-ROM, the SPT-ROM, and the FPT-ROM.

The main purpose of this paper is to focus on the random oracle model and to capture its crucial properties which make the cryptosystems secure. Note that Halevi and Krawczyk [16] posed a question of exhibiting variants of the random oracle model where one can argue about functions that "behave randomly but are not collision resistant". Our formalization of the CT-ROM gives a partial answer to their question.

We do not intend to model the attacks recently presented by Wang et al. against MD5, SHA-1, etc [6,7]. One important extension/generalization of our research would be to study the weakness of cryptosystems by taking into consideration the recently presented attacks. This direction is out of our scope in this paper.

Another direction of research would be to replace the basic property of the ROM that each entry is chosen uniformly at random and independent of the other entries. For example, we can extend our result concerning the FPT-ROM to the random permutation model. In this case, we would consider the oracles for both directions of the permutation, that is, the ideal cipher with a fixed key.

In order to show the differences of our models, we have focused on the RSA-based signature schemes. By replacing the RSA function with a trapdoor one-way permutation with the multiplicatively homomorphic property (i.e. $f(x \cdot y) = f(x) \cdot f(y)$), we can generalize our results. The efficiency of the reduction would be the same as that of the RSA-based schemes. If a trapdoor one-way permutation does not have the multiplicatively homomorphic property, we can still generalize our results, but the reductions are not tight. When f has the multiplicatively homomorphic property, then we can embed the information of y (the challenge instance of one-wayness) into all of the hash values queried by the adversary. When f does not have the multiplicatively homomorphic property, we cannot embed in a similar way as in the case with the multiplicatively homomorphic property. Therefore we have to choose one hash value to embed the information of y.

In order to analyze the security of schemes, we have assumed that we can efficiently sample from the binomial distribution $\mathrm{B}(N, p)$ perfectly. We could relax this perfectness to statistically closeness by modifying the security proofs. Making polynomial-time algorithms or analyzing precisely the algorithms proposed before are also interesting problems found in this paper.

It is also interesting to analyze the security of other cryptosystems, e.g., encryption, identification, in our models.

Acknowledgements

We are grateful to Martijn Stam for giving us many valuable comments on both technical and editorial problems in the initial version of this paper. His suggestions also helps us to improve the representation of this paper. We would also like to thank anonymous referees for their constructive comments including possible future works. We mention some of these in the concluding remarks.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
3. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)
5. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. J. Cryptology 17(2), 81–104 (2004)
6. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
7. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
8. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007)
9. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
10. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
11. Pasini, S., Vaudenay, S.: Hash-and-sign with weak hashing made secure. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 338–354. Springer, Heidelberg (2007)
12. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
13. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
14. Ahrens, J.H., Dieter, U.: Computer methods for sampling from Gamma, Beta, Poisson and Binomial distributions. Computing 12, 223–246 (1974)
15. Coron, J.S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
16. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)