Security of Encryption Schemes in Weakened Random Oracle Models

Akinori Kawachi* Akira Numayama* Keisuke Tanaka* Keita Xagawa*

March 5, 2010

Abstract

Liskov proposed several weakened versions of the random oracle model, called *weakened random oracle models* (WROMs), to capture the vulnerability of ideal compression functions, which are expected to have the standard security of hash functions, i.e., collision resistance, second-preimage resistance, and one-wayness properties. The WROMs offer additional oracles to break such properties of the random oracle. In this paper, we investigate whether public-key encryption schemes in the random oracle model essentially require the standard security of hash functions by the WROMs. In particular, we deal with four WROMs associated with the standard security of hash functions; the standard, collision tractable, second-preimage tractable, first-preimage tractable ones (ROM, CT-ROM, SPT-ROM, and FPT-ROM, respectively), done by Numayama et al. for digital signature schemes in the WROMs. We obtain the following results: (1) The OAEP is secure in all the four models. (2) The encryption schemes obtained by the Fujisaki-Okamoto conversion (FO) are secure in the SPT-ROM. However, some encryption schemes with FO are insecure in the FPT-ROM. (3) We consider two artificial variants wFO and dFO of FO for separation of the WROMs in the context of encryption schemes. The encryption schemes with wFO (dFO, respectively) are secure in the CT-ROM (ROM, respectively). However, some encryption schemes obtained by wFO (dFO, respectively) are insecure in the SPT-ROM (CT-ROM, respectively). These results imply that standard encryption schemes such as the OAEP and FO-based one do not always require the standard security of hash functions. Moreover, in order to make our security proofs complete, we construct an efficient sampling algorithm for the binomial distribution with exponentially large parameters, which was left open in Numayama et al.'s paper.

Keywords: public-key encryption schemes, weakened random oracle models, OAEP, Fujisaki-Okamoto conversion.

*Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan. {kawachi, numayam4, keisuke, xagawa5}@is.titech.ac.jp

Contents

1 Introduction

Background: In order to design new cryptographic schemes, we often follow the random oracle methodology [BR93]. First, we analyze the security of cryptographic schemes, by idealizing hash functions as truly random functions called the *random oracle*. When it comes to implementations of these schemes, we replace the random oracles by cryptographic hash functions such as MD5 [Riv92] and SHA-1 [Nat02]. This replacement is called an instantiation of the random oracle.

The random oracle methodology causes a trade-off between efficiency and provable security. The schemes proven secure in the random oracle model (ROM) are in general more efficient than those proven secure in the standard model. However, the security proofs in the ROM do not directly guarantee the security in the standard model, i.e., an instantiation of the random oracle might make the cryptographic schemes insecure. Even worse, several recent works [CGH04, GK03, BBP04] showed that some schemes secure in the ROM have no secure instantiation.

There are several properties of the ROM to prove the security of cryptographic properties. In particular, the ROM is expected to satisfy the one-wayness, second-preimage resistance, and collision resistance properties. We call these properties as the *standard security of hash functions*. These properties are indeed critical in many schemes for their security proofs. For example, the security of the Full-Domain-Hash (FDH) signature schemes (e.g., [BR96]), which are secure in the ROM, relies on the collision-resistance property of the ROM. That is, if we can obtain two distinct messages m, m' such that $H(m) = H(m')$ and the signature $\sigma = \mathsf{Sig}(H(m))$, then we can obtain a valid forgery (m', σ), where H is a hash function and Sig is a signing algorithm. Leurent and Nguyen also presented the attacks extracting the secret keys on several *hash-then-sign* type signature schemes and identity-based encryption schemes if the underlying hash functions are not collision resistant [LN09].

Recent progress on the attacks against cryptographic hash functions such as MD5 and SHA-1 raises the question on the assumption that hash functions are collision resistant and one-way (e.g.,[WY05, WYY05, AS09]). Therefore, it is significant to investigate whether the collision resistance property (as well as the one-wayness and second-preimage resistance properties, which are weaker notions than the collision resistance one) of the ROM is essential to prove the security of the schemes or not. More generally, it is worth classifying the schemes by the first-preimage, second-preimage, and collision resistance properties of the ROM that their security essentially requires.

Weak versions of random oracle models: Several works recently highlighted some specific properties of the ROM for secure cryptographic constructions in the ROM.

Nielsen proposed the *non-programmable* random oracle model where the random oracle is not *programmable* [Nie02]. In this model, one cannot set the values that the random oracle answers to some convenient values. It was showed in [Nie02] that a non-interactive non-committing encryption scheme exists in the ROM (assuming that trapdoor permutations exists), but not in the *non-programmable* random oracle model.

Unruh proposed a ROM with *oracle-dependent* auxiliary inputs [Unr07]. In this setting, adversaries obtain an auxiliary input that contains information with respect to the random oracle (e.g. collisions). He showed that the RSA-OAEP encryption scheme [BR95] is secure in the ROM even under the presence of *oracle-dependent* auxiliary inputs.

Liskov proposed several weakened versions of the random oracle model, called *weakened random oracle models* (WROMs), which offer additional oracles to break some properties of the random oracle [Lis07]. These model captures the situation that adversaries are given an attack algorithm for breaking some specific property of the functions. For example, the first-preimage tractable random oracle model offers the random oracle and the first-preimage oracle associated with the random oracle, which returns a first-preimage of the random oracle to adversaries. This first-preimage oracle then corresponds to the attack to the first preimage property of a hash function. We can replace the additional oracle to

others such as the second-preimage and collision ones that correspond to the attack to the properties. Thus, the WROMs can capture vulnerability of hash functions even if the parties are allowed to utilize ideal ones as in the ROM. By using WROMs, Liskov constructed hash functions based on weak ideal compression functions and proved it is indifferentiable from the random oracle.

Several results already analyzed the security in the WROMs. Hoch and Shamir applied Liskov's idea to prove the indifferentiability of another hash construction [HS08]. Pasini and Vaudenay also applied Liskov's idea to the security analysis of digital signature schemes [PV07]. They considered the security of *hash-then-sign* type signature schemes in the first-preimage tractable random oracle model. Numayama, Isshiki, and Tanaka formalized the WROMs, which allows us to formally analyze the security of the schemes [NIT08]. By using these models, they classified several digital signature schemes by the properties of the ROM. Fischlin and Lehmann also proposed a weakened random oracle model in a similar way to Liskov's one in the context of secure combiners [FL07].

Our contributions: In this paper, we investigate whether public-key encryption schemes constructed in the ROM essentially require the standard security of hash functions by further extending the direction originated from Liskov. In particular, we consider their security in the standard, collision tractable, second-preimage tractable, and first-preimage tractable random oracle models (ROM, CT-ROM, SPT-ROM, and FPT-ROM, respectively for short). Note that they are ordered according to their strengths, i.e., the security of encryption schemes in the FPT-ROM implies that in the SPT-ROM and such implications hold between each adjacent two models.

We demonstrate that the security notions in the four WROMs can be strictly separated in the context of encryption schemes. For the separation, we focus on the security of the encryption schemes obtained by the Fujisaki-Okamoto conversion (FO) [FO99], its two artificial variants (dFO and wFO), and the OAEP [BR95]. Precisely, we prove the following four statements:

1. OAEP is IND-CCA2 secure in the FPT-ROM.
2. FO is IND-CCA2 secure in the SPT-ROM, but *not* IND-CPA secure in the FPT-ROM.
3. wFO is IND-CCA2 secure in the CT-ROM, but *not* IND-CCA2 secure in the SPT-ROM.
4. dFO is IND-CCA2 secure in the ROM, but *not* IND-CCA2 secure in the CT-ROM.

We summarize the security of four schemes in Table 1.

scheme/model	ROM	CT-ROM	SPT-ROM	FPT-ROM
OAEP	secure			
FO	secure			insecure
wFO	secure		insecure	
dFO	secure	insecure		

Table 1: Security of four schemes.

This separation suggests that some public-key encryption schemes essentially require the standard security of hash functions. These notions were also separated in the context of digital signature schemes in [NIT08]. We stress that the role of the collision and second-preimage oracles in encryption schemes is not as clear as that in digital signature schemes. For example, it is easy to see that the collision oracle, breaking the collision resistance property of the random oracle, directly makes a simple scheme vulnerable, but not so easy for the case of encryption schemes. Actually, we need to develop new proof techniques for the (in)security of encryption schemes under additional oracles.

It also suggests that standard encryption schemes such as the OAEP and FO-based ones do not always require the standard security of hash functions for the random oracle. We believe that our results do not only give an example of the first application of the WROMs to encryption schemes, but they are also of independent interest. As far as we know, our results give the first evidence that the OAEP encryption scheme can be used in a practical application even without the first-preimage resistance property, i.e., the one-wayness property. In other words, the OAEP remains secure even if we remove the first-preimage resistance property. This can also be said on FO-based encryption schemes on the second-preimage resistance property.

On the security of the OAEP, Kiltz and Pietrzak recently showed that there is no construction for padding-based encryption schemes including the OAEP that has a black-box reduction from ideal trapdoor permutations to its IND-CCA2 security in [KP09]. However, they wrote in the paper that the security proof in the ROM can be still a valid argument in practice. We believe so is our security proof in the WROMs.

For the security proof, we explicitly show how to sample approximately in polynomial time from binomial distributions with exponentially large parameters, that is, a polynomial-time sampling algorithm whose output distribution is statistically close to the binomial distribution. For this algorithm, we arrange and combine sampling algorithms that run over real numbers proposed in the field of statistics [Dev86, AD74, AD80, Rel72], and give a precise analysis for discretization.

It should be noted that on the security proofs of the digital signature schemes in the WROMs [NIT08], Numayama et al. assumed such an efficient sampling algorithm and thus gave no explicit construction. They left the construction of the sampling algorithm as an open problem. By the sampling algorithm we explicitly show, it is no longer necessary to assume the sampling algorithm in their security proofs of the digital signature schemes [NIT08] as well as those of the public-key encryption scheme in this paper.

The sampling algorithm shown in this paper is adapted for cryptographic use since the statistical closeness to the original distribution is measured by the total variation distance, which is standard in cryptography but not usually required in statistics. The sampling algorithm is useful for other cryptographic tasks as in Numayama et al.'s and this paper.

Comparisons with other models: As mentioned above, a few models that weaken the power of the random oracle were already proposed such as the non-programmable model [Nie02] and the oracle-dependent auxiliary input model [Unr07].

The non-programmable model is not simply comparable with WROMs since the programmability does not imply the collision resistance and vice versa. The target of the oracle-dependent auxiliary input model partially overlaps that of the WROMs.

For a simple comparison, we now focus on the security of the OAEP in both models. Unruh showed a similar result as ours for the OAEP encryption scheme [Unr07]. He proposed a random oracle model where oracle-dependent auxiliary inputs are allowed. In his setting, the adversary of some cryptographic protocol obtains an auxiliary input that contains the information (e.g., collisions) on the random oracle. He showed that the OAEP encryption scheme [BR95] is still secure in the random oracle model even in his model. This result indicates an important fact that the security of the OAEP encryption scheme does not depend on the collision resistance property since the oracle-dependent auxiliary input can contain a sufficiently long list of collisions.

Our results also present the security of the OAEP in a weak version of the random oracle. However, there are at least two differences between Unruh's result and ours. First, the random oracle model with the oracle-dependent auxiliary input does not completely capture the *adaptive* security of hash functions, and this model still has the second-preimage resistance and the first-preimage resistance properties. Hence, only by his result, we cannot say whether these two properties are necessary or not in order to prove the security of the OAEP encryption scheme. In contrast to Unruh's result, our result clearly

shows that the two adaptive securities of hash functions such as the first-preimage resistance and the second-preimage resistance are not necessary to prove the security of the OAEP encryption scheme.

Second, Unruh constructed the reduction algorithm which breaks the partial-domain one-wayness of the underlying trapdoor permutation using the adversary which breaks the IND-CCA2 security of the OAEP encryption scheme. The running time of the reduction algorithm is not bounded by any polynomial. Therefore, he use the security amplification technique for the partial-domain one-wayness. By using this technique, he can avoid employing a stronger assumption that even quasi-polynomial time adversary cannot break the partial-domain one-wayness, and can prove the security under the standard partial-domain one-wayness against polynomial-time adversary.

In contrast to Unruh's result, we construct the polynomial-time reduction algorithm using the adversary, and hence we do not require the security amplification technique for the partial-domain one-wayness, which can be considered as a simplification of Unruh's proof.

Organization: In Section 2, we describe the details of the WROMs and their properties. We also discuss the simulation methods that are applicable to these models. In Section 3, after reviewing the encryption schemes we consider, we show their (in)security in the WROMs. Appendices contain several technical details. Appendix A reviews the simulation methods of WROMs by Numayama et al. Appendices **??** and B proves the technical lemmas, Lemmas 2.3 and 2.4, respectively. Proofs of the security of dFO, wFO, FO, and OAEP are in Appendices C, D, E, and F, respectively. In Appendix G, we give an overview of the approximation sampling algorithms. Appendix H reviews the standard notions and the arithmetic operations and Appendix I reviews the definitions of the standard distributions. In Appendix J, we show several inequalities for the distributions. In Appendix K, we rigously analyze the approximation sampling algorithms for several distributions.

Notation: Before starting technical parts of this paper, we introduce our notation used in the rest of the paper. For a table $\mathbb{T} = \{(x, y)\}$, we define $\mathbb{T}(y) = \{(x', y') \in \mathbb{T} \mid y' = y\}$. For a distribution D, $x \leftarrow D$ denotes that x is sampled according to D. The function $D(x)$ stands for the probability function of the distribution D.

Let $s \leftarrow S$ denote that s is sampled from the uniform distribution over a finite set S. $\#S$ denotes the number of elements in S. For a probabilistic Turing machine $\mathcal{A}$ and its input x, let $\mathcal{A}(x)$ denote the output distribution of $\mathcal{A}$ on input x.

We usually denote by k a security parameter of a cryptographic scheme in this paper. We also denote by k' length of plaintexts unless it is specified. k' is implicitly assumed to be polynomially related to the security parameter k, that is, $k' = k^{\Theta(1)}$. We say a function $f(k)$ is negligible in k if $f(k) = 2^{-\omega(\log k)}$. For two distributions D_1 and D_2 over a finte set S, whose density functions are denoted by f_{D_1} and f_{D_2}, we denote the statistical distance (the total variation distance) between them by $\Delta(D_1, D_2)$, defined by $\frac{1}{2}\sum_{s\in S}\left|f_{D_1}(s) - f_{D_2}(s)\right|$. We say two distributions D_1 and D_2 are statistically close if $\Delta(D_1, D_2) = 2^{-\omega(\log k)}$.

2 The Weakened Random Oracle Models

In this section, we first review the definitions of the WROMs. Next, we present an important property called *weak uniformity* of the WROMs, which is useful for security proofs of encryption schemes. We also discuss the simulation methods of [NIT08] used for the security proofs in the WROMs.

2.1 Definitions of the Weakened Random Oracle Models

To give formal definitions of the WROMs, we define some notation. Let X and Y be finite sets. Let H be a hash function chosen randomly from all of the functions from X to Y. We denote by $\mathbb{T}_H$ the table $\{(x, H(x)) \mid x \in X\}$. We identify the hash function H with the table $\mathbb{T}_H$.

We next define the random oracle and the additional oracles associated with $H : X \to Y$ as follows. (For more details, see [NIT08].)

Random oracle $\mathcal{RO}^H$: Given x, return y such that $(x, y) \in \mathbb{T}_H$.

Collision oracle $\mathcal{CO}^H$: On the query, first pick one entry $(x, y) \in \mathbb{T}_H$ uniformly at random. If there is no other entry $(x', y) \in \mathbb{T}_H$, then answer $\perp$. Otherwise, pick one entry $(x', y) \in \mathbb{T}_H$ satisfying $x \neq x'$ uniformly at random and answer (x, x').

Second-preimage oracle $\mathcal{SPO}^H$: Given (x, y), if $(x, y) \notin \mathbb{T}_H$ answer $\perp$. If there is no other entry $(x', y) \in \mathbb{T}_H$, then answer $\perp$. Otherwise, pick one entry $(x', y) \in \mathbb{T}_H$ satisfying $x \neq x'$ uniformly at random and answer x'.

First-preimage oracle $\mathcal{FPO}^H$: Given y, if there is any entry $(x, y) \in \mathbb{T}_H$ then return such an x uniformly at random. Otherwise return $\perp$.

Remark 2.1. *We usually identify the random oracle and the underlying hash function. However, in this paper as in [NIT08], we explicitly distinguish them by regarding the random oracle as an interface to the underlying hash function. This setting helps us to make the* WROMs *with an additional oracle well-defined.*

The formal definitions of the WROMs are given as follows. The WROMs consist of three components, a hash function h chosen randomly from all of the functions from X to Y, the random oracle, and the additional oracle associated with h. The models are called the CT-ROM, SPT-ROM, and FPT-ROM, if the additional oracle is the collision, second-preimage, and first-preimage oracle, respectively.

Remark 2.2. *The collision oracle may output $\perp$ even if there exists a collision (x, x') in the table. This stems from the simulation method of Numayama et al. [NIT08], and causes no serious problems. Note that the collision oracle outputs $\perp$ with probability $(1 - 1/\#Y)^{\#X-1}$. In the case where $\#X \geq \#Y$, we can find a collision with polynomially many queries since $(1 - 1/\#Y)^{\#X-1} \leq \exp(-(\#X - 1)/\#Y)$. In the case where $\#Y = k^{O(1)} \cdot \#X$, we can again find a collision with polynomially many queries since $(1 - 1/\#Y)^{\#X-1} \leq 1 - 1/k^{O(1)}$. Finally, in the case where $\#Y = k^{\omega(1)} \cdot \#X$, the following lemma shows that there are no collisions with overwhelming probability.*

Lemma 2.3. *Let $H : X \to Y$ be the hash function, and n_y the number of preimages of y under the function H, that is, $n_y = \#\mathbb{T}_H(y)$. Let* BAD *denote the event that there is some y such that $n_y > L$. Then for all sufficiently large Y, we have $\Pr_H[\mathsf{BAD}] < \frac{1}{(\#Y)^2}$, where $L = \frac{5 \ln \#Y}{\ln \ln \#Y} \frac{\#X}{\#Y}$ if $\#X \geq \#Y$, or $L = \frac{5 \ln \#Y}{\ln \ln \#Y}$ otherwise.*

The proof is obtained by the standard argument on the balls and bins game by regarding X and Y as sets of balls and bins, respectively. For the details on the game, see a standard textbook (e.g., [MR95]).

2.2 Difference from the Random Oracle Model

We observe an important difference between the ROM and WROMs by considering the ROM and FPT-ROM. In the both models, the function H, i.e., the table $\mathbb{T}_H$ is uniformly distributed.

In the ROM, if one queries some x that has never been queried to the random oracle, the value of $H(x)$ is uniformly distributed regardless of the past queries. That is, the knowledge of the past queries

does not affect the entries not queried in the table. This property of the ROM is called *uniformity*. In contrast to the situation in the ROM, when it comes to the FPT-ROM, this property is not attained. Recall that the first-preimage oracle *uniformly* returns one of the preimages, say x, of queried value y. If the first-preimage oracle leaks a number of preimages of y, the value of $H(x)$ is *not* uniformly distributed for an x not queried yet.

In order to observe this situation, let us consider the following extreme case. Let $y^* = H(x^*)$ for some $x^* \in X$ and suppose that y^* has the unique preimage x^*. Then the first-preimage oracle always returns the same x^* on the input y^*, which convinces us that the number of the preimages of y^* is exactly 1. This implies that the other $x \neq x^*$ does not take a value y^* under H. Therefore, the random oracle no longer has the uniformity in the FPT-ROM. This is a critical difference between the ROM and FPT-ROM since we often make use of the uniformity in the security proofs of the public-key encryption schemes.

We prove the following lemma to overcome this barrier in the WROMs, which states that the WROMs still has weak uniformity instead of the uniformity. The weak uniformity is still useful for the security proofs of the public-key encryption schemes in the WROMs. See Appendix B for the proof.

Lemma 2.4 (Weak Uniformity)**.** *In the* WROMs*, the output distribution of the random oracle is statistically close to the uniform distribution. More formally, it is stated as follows. Let $H : X \to Y$ be the hash function in the* WROMs*. Let $\mathcal{A}$ be a probabilistic oracle Turing machine that makes at most q queries to the random oracle $\mathcal{RO}^H$ and the additional oracle O^H, where O^H represents one of the additional oracles $\mathcal{CO}^H$, $\mathcal{SPO}^H$, and $\mathcal{FPO}^H$. $V_{\mathcal{A},H}(x)$ denotes the random variable that represents the hash value $\mathcal{RO}^H(x)$, where $x \leftarrow \mathcal{A}^{\mathcal{RO}^H,O^H}$ and the correspondence $(x, H(x)) \in \mathbb{T}_H$ is not answered by the two oracles.*

Then, for any $\mathcal{A}$, the following holds:

$$\Delta(V_{\mathcal{A},H}(x), U_Y) \leq \begin{cases} \frac{1}{\#Y}\left(5q + 1 + \frac{4q^2}{\#Y} + 20q\frac{\ln \#Y}{\ln\ln \#Y}\right) & \text{if } \#X \geq \#Y, \\ \frac{1}{\#X}\left(5q + 1 + \frac{4q^2}{\#X} + 20q\frac{\ln \#Y}{\ln\ln \#Y}\right) & \text{if } \#X < \#Y. \end{cases}$$

Here, the probability is taken over random choices of the hash function H and the random coin of $\mathcal{A}$.

2.3 Simulation Methods

In almost all the security proofs in the ROM, the reduction algorithms simulate the random oracles. When it comes to the security proofs in the WROMs, the reduction algorithms have to simulate both the random and the additional oracle, which makes differences of the simulation methods in the WROMs from those in the ROM.

Numayama et al.'s methods: Numayama et al. proposed the simulation methods for WROMs, but they required an unproven assumption. Let $\mathrm{B_N}(N, p)$ denote the binomial distribution with parameters N and p whose probability function is $f_{\mathrm{B_N}}(x \mid N, p) = \binom{N}{x} p^x (1-p)^{N-x}$ for $x = 0, \ldots, N$, where the parameters N and p take values approximately $\#X$ and $1/\#Y$ for a hash function $H : X \to Y$, say, $(N, p) = (2^{128}, 2^{-128})$. Their simulation methods required the efficient sampler for $\mathrm{B_N}(N, p)$ with exponentially large N and small p, and they assumed its existence.

Assumption 2.5. *There is a probabilistic Turing machine* $\mathsf{B_N}$ *such that the output distribution* $\mathsf{B_N}(N, p)$ *on inputs N and p is equal to the binomial distribution* $\mathrm{B_N}(N, p)$ *and it runs in polynomial time in* $\log N$ *and* $\log p^{-1}$*, where N is a positive integer and* $0 \leq p \leq 1$ *is a rational number.*

Under this assumption, they constructed the simulation algorithms, RO, CO, SPO, and FPO, for the security proofs in the WROMs as given in the following proposition. See Appendix A for the details of the algorithms.

Proposition 2.6 (Simulation Method [NIT08])**.** *We can perfectly simulate the random oracle, the collision oracle, second-preimage oracle, and first-preimage oracle in the* WROMs *under Assumption 2.5. That is, the output distributions of the random oracle, collision oracle, second-preimage oracle, and first-preimage oracle in the* WROMs *are identical to the output distributions of the algorithms* RO*,* CO*,* SPO*, and* FPO*, under Assumption 2.5.*

Removing the assumption: For the security proof in the WROMs of digital signature schemes in [NIT08] and encryption schemes in this paper, it is sufficient to utilize a weaker sampling algorithm that generates a distribution *not equal but statistically close* to the binomial distribution $\mathrm{B}_N(N, p)$. Then, their security proofs can work by just adding negligibly small errors induced by the statistical distance in their analyses.

There are quite many papers (e.g., [Rel72]) on the efficient sampling methods from the binomial distribution in the field of statistics. However, their basic computation model is totally different from the model in the cryptography. As far as the authors' knowledge, all these results are based on the computation model that directly manipulates *real* numbers without errors. If we translate them to those in the bit computation model used in the cryptography, we have to bound the statistical distance between the real distribution and the output distribution generated by the sampling algorithms in the bit computation model rather than the real-number one. Numayama et al. mentioned that they could neither find precise analyses of the statistical distance, nor construct the sampling algorithms by themselves in [NIT08]. Therefore, they had to put the above assumption.

In fact, there is an efficient sampling algorithm appropriate for our purpose in the real-number computation model [Rel72]. We modify the algorithm and rigorously analyze the error bound in the bit computation model. We can finally obtain the following theorem on the sampling algorithm.

Theorem 2.7. *For ϵ, there is a probabilistic Turing machine B_N such that, for the output distribution $\mathsf{B}_\mathsf{N}(N, p)$ on inputs N, p, the statistical distance between $\mathsf{B}_\mathsf{N}(N, p)$ and $\mathrm{B}_N(N, p)$ is at most ϵ and it runs in polynomial time in $\log N, \log p^{-1}$ and $\log \epsilon^{-1}$, where N is a positive integer and $0 \leq p \leq 1, 0 < \epsilon \leq 1$ are rational numbers.*

Note that the algorithm can control the error parameter ϵ. This property is useful in cryptographic applications for the security proofs even if the other parameters N and p are not sufficiently large. We put the details of the algorithm and its analysis in Appendices.

As a result, we can remove the above assumption and obtain the following theorem.

Theorem 2.8 (Simulation Method without Assumption 2.5)**.** *We can statistically simulate the random oracle, collision oracle, second-preimage oracle, and first-preimage oracle in the* WROMs*. That is, the output distributions of the oracles in the* WROMs *are statistically close to the output distributions of the algorithms* RO*,* CO*,* SPO*, and* FPO*, respectively.*

3 The Encryption Schemes and Their Security in the Weakened Random Oracle Models

In this section, we examine the security in the WROMs of the public-key encryption schemes. We particularly discuss separations for notions of ROM, CT-ROM, SPT-ROM, and FPT-ROM by showing (in)security of public-key encryption schemes obtained by the Fujisaki-Okamoto conversion (FO) and its two variants (dFO and wFO), and OAEP.

Public-key encryption schemes: We first give notation and notions for public-key encryption schemes briefly. For details, see standard textbooks, e.g., [KL07].

A public-key encryption scheme $\mathcal{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over a plaintext space $\mathcal{M}$ and a random coin space $\mathcal{R}$ is defined by the following three algorithms. Let k denote the security parameter.

Key Generation: On input 1^k, the key generation algorithm $\mathsf{Gen}(1^k)$ produces a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

Encryption: Given a public key pk, a plaintext $m \in \mathcal{M}$, and a random string $r \in \mathcal{R}$, the encryption algorithm $\mathsf{Enc}_{\mathsf{pk}}(m; r)$ outputs a ciphertext c corresponding to the plaintext m.

Decryption: Given a secret key sk and ciphertext c, the decryption algorithm $\mathsf{Dec}_{\mathsf{sk}}(c)$ outputs the plaintext $m \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$ corresponding to the ciphertext c.

We require the perfect completeness, that is, for every $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{Gen}(1^k)$, every plaintext $m \in \mathcal{M}$, and every random string $r \in \mathcal{R}$, it should be satisfied that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m; r)) = m$.

We only consider three standard security notions for public-key encryption schemes, the one-wayness against chosen-plaintext attack (OW-CPA), the indistinguishability against chosen-plaintext attack (IND-CPA), and the indistinguishability against adaptive chosen-ciphertext attack (IND-CCA2).

For $\gamma = \gamma(k)$, we say $\mathcal{PKE}$ is γ-uniform if for any key pair $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{Gen}(1^k)$, any $m \in \mathcal{M}$, and $c \in \{0,1\}^*$, we have $\Pr_{r \leftarrow \mathcal{R}}[c = \mathsf{Enc}_{\mathsf{pk}}(m; r)] \leq \gamma$. There exists a OW-CPA public-key encryption scheme with γ-uniformity (e.g., the ElGamal encryption scheme).

Brief review for FO**:** Fujisaki and Okamoto proposed a conversion, called the Fujisaki-Okamoto (FO) conversion, to obtain highly secure public-key encryption schemes in the ROM [FO99]. Since the standard one-time pad satisfies the requirement of the FO conversion, we fix the one-time pad as the symmetric-key encryption scheme used in the FO conversion for simplicity.

Let $\mathcal{PKE}$ be a OW-CPA secure and γ-uniform public-key encryption scheme over a plaintext space $\mathcal{M}$ and a randomness space $\mathcal{R}$. Then the FO conversion converts $\mathcal{PKE}$ to an IND-CCA2 secure one $\mathcal{PKE}' = \mathrm{FO}(\mathcal{PKE})$ over a plaintext space $\mathcal{M}' = \{0,1\}^{k'}$ and a randomness space $\mathcal{R}' = \mathcal{M}$, where k' denotes the length of plaintexts, which is polynomially related to the security parameter k. The encryption procedure of $\mathcal{PKE}'$ is given as follows: For a plaintext $m \in \mathcal{M}' = \{0,1\}^{k'}$ and a random string $r \in \mathcal{R}' = \mathcal{M}$, the ciphertext is

$$(c_1, c_2) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(m, r)), G(r) \oplus m),$$

where $H : \{0,1\}^{k'} \times \mathcal{M} \to \mathcal{R}$ and $G : \mathcal{M} \to \{0,1\}^{k'}$ are hash functions modeled as the random oracles. The decryption procedure is given as follows: For a given ciphertext (c_1, c_2), decrypt c_1 by sk and obtain r. Then, extract m by $c_2 \oplus G(r)$ and verify $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; H(m, r))$. If not output $\perp$. Roughly speaking, $H(m, r)$ ensures that if a ciphertext (c_1, c_2) is valid then the encryptor producing (c_1, c_2) knows corresponding m and r.

3.1 The First Variant dFO

We introduce the first artificial variant dFO and show that dFO is secure in the ROM, but not secure in general in the CT-ROM.

The variant dFO converts a public-key encryption scheme $\mathcal{PKE}$ (with the one-time pad) to another public-key encryption scheme $\mathcal{PKE}' = \mathrm{dFO}(\mathcal{PKE})$ similarly to FO. The encryption procedure of $\mathcal{PKE}'$ is defined as follows. For a plaintext $m \in \mathcal{M}' = \{0,1\}^{k'}$ and a random string $r \in \mathcal{R}' = \mathcal{M}$, the ciphertext of $\mathcal{PKE}'$ is

$$(c_1, c_2) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(F(m), r)), G(r) \oplus m),$$

where $F : \{0,1\}^{k'} \to \mathcal{P}$, $G : \mathcal{M} \to \{0,1\}^{k'}$, and $H : \mathcal{P} \times \mathcal{M} \to \mathcal{R}$, for an appropriate set $\mathcal{P}$, are hash functions modeled as the random oracle. Formal description is in Table 1.

Key Generation		Encryption		Decryption	
Input:	1^k	Input:	$m \in \{0,1\}^{k'}$	Input:	(c_1, c_2)
1:	$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$	1:	$r \leftarrow \mathcal{M}$	1:	$r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$
Output:	$(\mathsf{pk}, \mathsf{sk})$	2:	$g \leftarrow G(r)$	2:	$g \leftarrow G(r)$
		3:	$h \leftarrow H(F(m), r)$	3:	$m \leftarrow c_2 \oplus g$
		4:	$c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r; h)$	4:	$h \leftarrow H(F(m), r)$
		5:	$c_2 \leftarrow m \oplus g$	5:	If $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ set $o \leftarrow m$
		Output:	(c_1, c_2)	6:	Otherwise set $o \leftarrow \perp$
				Output:	o

Figure 1: $\mathcal{PKE}'$ obtained by the dFO conversion.

The idea to weaken the conversion is summarized as follows: Recall that $H(m, r)$ in the FO conversion can be considered as encryptor's signature (or a proof of knowledge) on m and r. To make it vulnerable by a collision, we introduce a new random oracle F and replace $H(m, r)$ with $H(F(m), r)$. The replacement does not harm the security in the random oracle model, while it can be exploited by the presence of the collision oracle CO^F.

Formally, we have following theorems on the (in)security. The proof of Theorem 3.1 is in Appendix C.

Theorem 3.1. *Assume that $\mathcal{PKE}$ is a* OW-CPA *secure and γ-uniform public-key encryption scheme for some negligible γ. Then, $\mathcal{PKE}' = \mathrm{dFO}(\mathcal{PKE})$ is* IND-CCA2 *secure in the* ROM *if $\#\mathcal{P} = 2^{\omega(\log k)}$.*

Theorem 3.2. *Let $\mathcal{PKE}$ be a public-key encryption scheme. If $\#\mathcal{P} \leq 2^{k'}$ then $\mathcal{PKE}' = \mathrm{dFO}(\mathcal{PKE})$ is not* IND-CCA2 *secure in the* CT-ROM.

Proof. We construct the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND-CCA2 security of $\mathcal{PKE}'$, which exploits the collision oracle CO^F of F.

The adversary $\mathcal{A}_1$, on input pk, first queries to CO^F. If the answer is $\perp$, then the adversary flips a random fair coin b', outputs b', and halts. Otherwise, it obtains a collision (m_1, m_2) of F and outputs it as a challenge. The adversary $\mathcal{A}_2$ receives the target ciphertext $(c_1^*, c_2^*) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_b), r)), G(r) \oplus m_b)$ for some $r \in \mathcal{R}'$. It queries $(c_1', c_2') = (c_1^*, c_2^* \oplus m_0 \oplus m_1)$ to the decryption oracle and obtains m_{1-b}, since

$$c_1' = \mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_0), r)) = \mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_1), r)),$$
$$c_2' = G(r) \oplus m_b \oplus m_0 \oplus m_1 = G(r) \oplus m_{1-b}.$$

Hence, the adversary can answer $b' = b$ correctly.

Finally, we upper-bound the probability that the collision oracle outputs $\perp$, which stems from the definition of the collision oracle. The probability is bounded by $(1 - 1/\#\mathcal{P})^{2^{k'}-1} \leq \exp(-(2^{k'} - 1)/\#\mathcal{P}) \leq 1/\sqrt{e}$. This completes the proof. □

3.2 The Second Variant wFO

We next introduce the second artificial variant wFO and show that the obtained scheme by wFO is secure in the CT-ROM, however not generally secure in the SPT-ROM.

The encryption procedure of $\mathcal{PKE}' = \mathrm{wFO}(\mathcal{PKE})$ is given as follows. For a plaintext $m \in \mathcal{M}' = \{0,1\}^{k'}$ and random strings $(r, s) \in \mathcal{R}' = \mathcal{M} \times \mathcal{S}$, the ciphertext of $\mathcal{PKE}'$ is

$$(c_1, c_2, c_3) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(F(m, s), r)), G(r) \oplus m, s),$$

Key Generation		Encryption		Decryption	
Input:	1^k	Input:	$m \in \{0,1\}^{k'}$	Input:	(c_1, c_2, c_3)
1:	$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$	1:	$r \leftarrow \mathcal{M}$	1:	$r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$
Output:	$(\mathsf{pk}, \mathsf{sk})$	2:	$g \leftarrow G(r)$	2:	$g \leftarrow G(r)$
		3:	$s \leftarrow \mathcal{S}$	3:	$m \leftarrow c_2 \oplus g$
		4:	$h \leftarrow H(F(m,s),r)$	4:	$h \leftarrow H(F(m,c_3),r)$
		5:	$c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r;h)$	5:	If $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r;h)$ set $o \leftarrow m$
		6:	$c_2 \leftarrow m \oplus g$	6:	Otherwise set $o \leftarrow \perp$
		7:	$c_3 \leftarrow s$	Output:	o
		Output:	(c_1, c_2, c_3)		

Figure 2: $\mathcal{PKE}'$ obtained by the wFO conversion.

where $F : \{0,1\}^{k'} \times \mathcal{S} \to \mathcal{P}$, $G : \mathcal{M} \to \{0,1\}^{k'}$, and $H : \mathcal{P} \times \mathcal{M} \to \mathcal{R}$ are hash functions modeled as the random oracles. The formal definition is in Table 2.

Notice that $(H(F(m,s),r),s)$ is a proof of knowledge on (m,r,s) which resists a collision on F however is vulnerable by a second-preimage attack against F as in Numayama et al. [NIT08].

We can show that the obtained scheme is IND-CCA2 secure in the CT-ROM by using Lemma 2.4. See Appendix D for the proof.

Theorem 3.3. *Suppose that $\mathcal{PKE}$ is a* OW-CPA *secure and γ-uniform public-key encryption scheme for some negligible γ. Then, $\mathcal{PKE}' = \mathrm{wFO}(\mathcal{PKE})$ is* IND-CCA2 *secure in the* CT-ROM *if $\#\mathcal{P}^{-1}$ and $\#\mathcal{S}^{-1}$ are negligible in k.*

However, its security is broken under the presence of the second-preimage oracle for F.

Theorem 3.4. *Let $\mathcal{PKE}$ be a public-key encryption. If $\#\mathcal{P} \leq 2^{k'} \cdot \#\mathcal{S}$, then the scheme $\mathcal{PKE}' = \mathrm{wFO}(\mathcal{PKE})$ is not* IND-CCA2 *secure in the* SPT-ROM.

Proof. We construct the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that exploits the second-preimage oracle $\mathcal{SPO}^F$ associated to F. The adversary $\mathcal{A}_1$ chooses random distinct plaintexts m_0 and m_1 and queries them to the challenger. The challenger responses

$$(c_1^*, c_2^*, c_3^*) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_b,s),r)), G(r) \oplus m_b, s).$$

Receiving (c_1^*, c_2^*, c_3^*), the adversary $\mathcal{A}_2$ queries (m_0, s) to the second-preimage oracle $\mathcal{SPO}^F$. If it receives $\perp$ from the second-preimage oracle, then it flips a random fair coin b', outputs b', and halts. Otherwise, it obtains $(m', s') \neq (m_0, s)$ such that $F(m_0, s) = F(m', s')$. So, the adversary queries

$$(c_1', c_2', c_3') = (c_1^*, c_2^* \oplus m_0 \oplus m', s')$$

to the decryption oracle. Notice that, if (c_1^*, c_2^*, c_3^*) is the valid ciphertext of m_0, then we have

$$\begin{aligned} c_1' &= \mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_0,s),r)) = \mathsf{Enc}_{\mathsf{pk}}(r; H(F(m',s'),r)), \\ c_2' &= G(r) \oplus m_0 \oplus m_0 \oplus m' = G(r) \oplus m', \\ c_3' &= s', \end{aligned}$$

and (c_1', c_2', c_3') is a valid ciphertext for m'. On the other hand, if the ciphertext is the encryption of m_1, we have

$$(c_1', c_2', c_3') = (\mathsf{Enc}_{\mathsf{pk}}(r; H(F(m_1,s),r)), G(r) \oplus m_1 \oplus m_0 \oplus m', s').$$

Thus, if $f = F(m_1, s)$ is equal to $F(m_1 \oplus m_0 \oplus m', s')$ the decryption oracle returns $m_1 \oplus m_0 \oplus m' (\neq m')$. Otherwise, the decryption oracle returns $\perp$.

Thus, if the answer is m', then the adversary concludes that (c_1^*, c_2^*, c_3^*) is the ciphertext of m_0, that is, it outputs $b' = 0$. Otherwise, the adversary concludes that it is the ciphertext of m_1, that is, it outputs $b' = 1$. Therefore, $\mathcal{A}$ can output the correct answer unless $\mathcal{A}$ receives $\perp$ from the second-preimage oracle.

We finally bound the probability that the oracle outputs $\perp$. It is bounded by $(1 - 1/\#\mathcal{P})^{2^{k'} \cdot \#\mathcal{S} - 1} \leq \exp(-(2^{k'} \cdot \#\mathcal{S} - 1)/\#\mathcal{P}) \leq 1/\sqrt{e}$ as required. This completes the proof. □

3.3 The Original Fujisaki-Okamoto Conversion

We next show that the obtained scheme by the conversion FO with the one-time pad is secure in the SPT-ROM, but not secure in the FPT-ROM in some parameter setting.

Let $G : \mathcal{M} \to \{0,1\}^{k'}$ and $H : \{0,1\}^{k'} \times \mathcal{M} \to \mathcal{R}$ be hash functions modeled as the random oracles. Recall the encryption procedure of $\mathcal{PKE}' = \mathrm{FO}(\mathcal{PKE})$. For a plaintext $m \in \mathcal{M}' = \{0,1\}^{k'}$ and a random string $r \in \mathcal{R}' = \mathcal{M}$, the ciphertext is $(\mathsf{Enc}_{\mathsf{pk}}(r; H(m,r)), G(r) \oplus m)$. The scheme is in Figure 3.

Key Generation		Encryption		Decryption	
Input:	1^k	Input:	$m \in \{0,1\}^{k'}$	Input:	(c_1, c_2)
1:	$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)$	1:	$r \leftarrow \mathcal{M}$	1:	$r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$
Output:	$(\mathsf{pk}, \mathsf{sk})$	2:	$g \leftarrow G(r)$	2:	$g \leftarrow G(r)$
		3:	$h \leftarrow H(m,r)$	3:	$m \leftarrow c_2 \oplus g$
		4:	$c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r; h)$	4:	$h \leftarrow H(m,r)$
		5:	$c_2 \leftarrow m \oplus g$	5:	If $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ set $o \leftarrow m$
		Output:	(c_1, c_2)	6:	Otherwise set $o \leftarrow \perp$
				Output:	o

Figure 3: $\mathcal{PKE}'$ obtained by the FO conversion.

Modifying the existing proofs, we can show the scheme is secure in the SPT-ROM using Lemma 2.4. The proof appears in Appendix E.

Theorem 3.5. *Suppose that $\mathcal{PKE}$ is* OW-CPA *secure and γ-uniform for some negligible γ. Then, $\mathcal{PKE}' = \mathrm{FO}(\mathcal{PKE})$ is* IND-CCA2 *secure in the* SPT-ROM.

However, the presence of the first-preimage oracle for G violates the IND-CPA security of $\mathcal{PKE}'$ in some parameter settings. Note that if m is $0^{k'}$, the second component of the ciphertext is $G(r)$, which is vulnerable the first-preimage oracle of G.

Theorem 3.6. *Let $C = \#\mathcal{M}/2^{k'}$. Assume that $C = k^{O(1)}$. Then, $\mathcal{PKE}' = \mathrm{FO}(\mathcal{PKE})$ is not* IND-CPA *secure in the* FPT-ROM.

Proof. We prove the theorem by constructing the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which exploits the first-preimage oracle of G, $\mathcal{FPO}^G$. The adversary $\mathcal{A}_1$, on input pk, queries $m_0 = 0^{k'}$ and $m_1 = 1^{k'}$ to the challenger. The adversary $\mathcal{A}_2$, on input the target ciphertext (c_1^*, c_2^*), queries c_2^* to the first-preimage oracle of G. If it obtains $\tilde{r}$, it checks that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(\tilde{r}; H(0^{k'}, \tilde{r}))$. If the check passes, the adversary outputs $b' = 0$. Otherwise, it flips a random fair coin b', outputs b', and halts.

It is obvious that if $b = 0$ and $\tilde{r} = r$, the adversary answers correctly, that is, it outputs $b' = b$. If $b = 1$, the preimage of the query $G(r) \oplus 1^{k'}$ never equals to r since $G(r) \neq G(r) \oplus 1^{k'}$. Hence, the adversary's check fails if $b = 1$.

We estimate the probability that the adversary wins. By Lemma 2.3, with probability at least $1-2^{-2k'}$, there is no preimage of size larger than L, where if $C \geq 1$ then $L = 5Ck' \ln 2/(\ln k' + \ln\ln 2) \leq 4Ck'/\ln k'$ and otherwise $L = 5k' \ln 2/(\ln k' + \ln\ln 2) \leq 4k'/\ln k'$ for all sufficiently large k'.

Let Good denote the event that $r \leftarrow \mathcal{FPO}_G(G(r))$. We then have $\Pr[\mathsf{Good}] \geq (1 - 2^{-2k'})/L$. Hence, we obtain that

$$\begin{aligned}
\Pr[b' = b] &= \Pr[b' = 0 \mid b = 0 \wedge \mathsf{Good}] \Pr[b = 0 \wedge \mathsf{Good}] \\
&\quad + \Pr[b' = 0 \mid b = 0 \wedge \neg\mathsf{Good}] \Pr[b = 0 \wedge \neg\mathsf{Good}] \\
&\quad + \Pr[b' = 1 \mid b = 1] \Pr[b = 1] \\
&= 1 \cdot \frac{1}{2} \cdot \Pr[\mathsf{Good}] + \frac{1}{2} \cdot \frac{1}{2} \cdot (1 - \Pr[\mathsf{Good}]) + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{4} \Pr[\mathsf{Good}] \geq \frac{1}{2} + \frac{1 - 2^{-2k'}}{4L}.
\end{aligned}$$

and $4L$ is a polynomial in the security parameter k. This completes the proof. □

As shown above, the FO conversion is not secure in the FPT-ROM, but there is a way to modify it so as to maintain the security in the FPT-ROM. Naito, Wang, and Ohta proposed the conversion method that converts a cryptosystem secure in the ROM to that secure even in the FPT-ROM [NWO09]. In the case of the FO conversion, the public key is (pk, c), where $c \leftarrow \{0,1\}^k$, and the ciphertext is

$$(c_1, c_2) = (\mathsf{Enc}_{\mathsf{pk}}(r; H(c, m, r)), G(c, r) \oplus m),$$

where the domains of H and G are modified. Intuitively, this change makes the first-preimage oracles, $\mathcal{FPO}^H$ and $\mathcal{FPO}^G$, useless.

3.4 OAEP

We finally focus on the OAEP and present its IND-CCA2 security in the FPT-ROM. For the security parameter k, let k_0 and k_1 be functions in k, where $k_0 < k - k_0$. Let F be a family of partial-domain one-way trapdoor permutations of a domain $\{0,1\}^{k-k_0} \times \{0,1\}^{k_0}$. (See [FOPS04] for the definition of the partial-domain one-wayness.) Furthermore, let G and H be hash functions such that $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$ and $H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$. Then, the OAEP encryption scheme based on F is described in Fig. 4. We obtain the following theorem that states the security of the OAEP encryption scheme in the

Key Generation		Encryption		Decryption	
Input:	1^k	Input:	$m \in \{0,1\}^{k-k_0-k_1}$, f_{pk}	Input:	c, g_{sk}
1:	$(f_{\mathsf{pk}}, g_{\mathsf{sk}}) \leftarrow F$	1:	$r \leftarrow \{0,1\}^{k_0}$	1:	$s \,\|\|\, t \leftarrow g_{\mathsf{sk}}(c)$
Output:	$(f_{\mathsf{pk}}, g_{\mathsf{sk}})$	2:	$s \leftarrow (m \,\|\| \, 0^{k_1}) \oplus G(r)$	2:	$r \leftarrow t \oplus H(s)$
		3:	$t \leftarrow H(s) \oplus r$	3:	$M \leftarrow s \oplus G(r)$
		4:	$c \leftarrow f_{\mathsf{pk}}(s \,\|\| \, t)$	5:	If $M = m \,\|\| \, 0^{k_1}$ set $o \leftarrow m$
		Output:	c	6:	Otherwise set $o \leftarrow \perp$
				Output:	o

Figure 4: OAEP

FPT-ROM.

Theorem 3.7. *Let F be a family of partial-domain one-way trapdoor permutations. Then, the OAEP encryption scheme based on F is* IND-CCA2 *secure in the* FPT-ROM.

See Appendix F for the proof.

4 Future Work

It should be noted that our WROMs are based on a simplified variant, which Numayama et al. [NIT08] and Pasini and Vaudenay [PV07] also adopted, of the original WROMs of Liskov [Lis07].

The original WROMs consists of the ideal compression function $h : \{0,1\}^{k+k'} \to \{0,1\}^k$ of *fixed input length* and the first-preimage oracle. Then, he discussed the security of the *flexible input-length* hash functions $H^h : \{0,1\}^* \to \{0,1\}^k$ employing h as the component in the context of indifferentiability [MRH04]. A random oracle H is often instantiated by employing a compression h. (See, e.g., the survey in [LN09, Section 2].) Therefore, his work reflects the attacks against the compression function of MD5 and SHA-1 rather than the construction H.

On the contrary, we (and similarly [NIT08, PV07]) discussed the *monolithic* random oracle H and the additional oracles associated with H. Hence, our model has a gap from such a realistic instantiation of the random oracle in some sense. We leave filling this gap as future work.

Except for the FO conversion, there are several conversion methods in the ROM, such as REACT [OP01] and GEM [CHJ+02]. It would also be interesting as future work to examine the security of these conversion methods in the WROMs.

Acknowledgements

We thank anonymous reviewers for their helpful comments. This research was supported in part by NTT Information Sharing Platform Laboratories, JSPS Global COE program "Computationalism as Foundation for the Sciences," KAKENHI 18300002, KAKENHI 19-55201, and the Japan Science and Technology Agency, Strategic Japanese-French Cooperative Program "Quantum Computer: Theory and Feasibility."

A Simulation Algorithms of Numayama et al.

In this section, we review the details of the algorithms RO, CO, SPO, and FPO which simulate the random oracle, the collision oracle, the second-preimage oracle, and the first-preimage oracle in the WROMs, respectively.

In each of WROMs, two tables $\mathbb{T}$ and $\mathbb{L}$ are shared by the simulation algorithm RO for the random oracle and the simulation algorithm for the additional oracle, e.g., CO in the CT-ROM. In the start of the simulations, both tables are empty. In the simulations, $\mathbb{T}$ will contain the pair of values (x, y) such that $x = h(y)$ and $\mathbb{L}$ will contain the pair of values (y, n) such that $n = \#\{x \in X \mid x = h(y)\}$. For the table $\mathbb{T} = \{(x, y)\}$, we define $\mathbb{T}(y) = \{(x', y') \in \mathbb{T} \mid y' = y\}$.

First, we review how the algorithm RO runs on input $\hat{x}$ in detail. If the hash value of $\hat{x}$ is already determined, then the algorithm RO returns it. Otherwise, there are two situations depending on whether the algorithm RO returns *old* y which is already appeared in the table $\mathbb{T}$ or the algorithm RO returns *new* y which is not yet appeared in the table $\mathbb{T}$. There are $(\#X - \#\mathbb{T})$ elements whose hash values are not yet determined, and among them there are $\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))$ elements whose hash values are expected to be old y. Therefore, the algorithm RO returns old y or new y with this ratio. In case of old y, the algorithm RO picks old y according to the number of the preimages of each old y. In case of new y, the algorithm RO picks new y uniformly at random, and defines the number of preimages of y. The whole algorithm is presented in Algorithm 2.

Algorithm RO($\hat{x}$)

1. If $(\hat{x}, y) \in \mathbb{T}$ for some y, then output y.
2. Let p be the following probability,
$$p = \frac{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))}{\#X - \#\mathbb{T}}.$$
3. With probability p, output old y as Steps (a)-(b).
   (a) pick $y \leftarrow D$ according to the following distribution.
   $$f_D(y) = \frac{n - \#\mathbb{T}}{\sum_{(\tilde{y},\tilde{n})\in\mathbb{L}}(\tilde{n} - \#\mathbb{T}(\tilde{y}))} \text{ for } (y, n) \in \mathbb{L}.$$
   (b) insert $(\hat{x}, y)$ in $\mathbb{T}$ and output y.
4. With probability $1 - p$ output new y as Steps (a)-(d).
   (a) pick $y \leftarrow Y \setminus \bigcup_{(\tilde{y},\tilde{n})\in\mathbb{L}}\{\tilde{y}\}$ uniformly at random.
   (b) $n' \leftarrow \mathrm{B}_N(\#X - \sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} \tilde{n} - 1, \frac{1}{\#Y - \#\mathbb{L}})$. ($\mathrm{B}_N(N, p)$ denotes the binomial distribution with parameters N and p.)
   (c) $n \leftarrow n' + 1$.
   (d) insert (y, n) in $\mathbb{L}$, insert $(\hat{x}, y)$ in $\mathbb{T}$, and output y.

Algorithm 2: Simulation method of the random oracle.

Next, we review how the algorithm CO runs in detail. First, it picks $x \leftarrow X$ as the original oracle does. If the hash value of x is not determined, it obtains the hash value y by the algorithm RO. If $n = 1$, which implies that there is only one preimage of y, then the algorithm CO returns $\perp$. Otherwise, there

are two situations depending on whether the algorithm CO returns *new x* which is not yet appeared in the table $\mathbb{T}$ or the algorithm CO returns *old x* which is already appeared in the table $\mathbb{T}$. There are n elements whose hash values are expected to be y, and among them there are $\#\mathbb{T}(y)$ elements which are already appeared in the table $\mathbb{T}$. Therefore, the algorithm CO returns old x or new x with this ratio. In case of old x, the algorithm CO picks old x according to both the current table $\mathbb{T}$ and the number of the preimages of each old y defined in the table $\mathbb{L}$. In case of new x, the algorithm CO picks new x uniformly at random. The whole algorithm is presented in Algorithm 3.

Algorithm CO()

1. Pick $x \leftarrow X$.
2. Invoke algorithm RO() and obtains $y \leftarrow \mathsf{RO}(x)$.
3. Look up $(n, y) \in \mathbb{L}$.
4. If $n = 1$, output $\perp$.
5. If $n \neq 1$, then compute the threshold $q = \frac{\#\mathbb{T}(y)-1}{n-1}$.
6. With probability q output x with an old element.
   (a) pick one entry uniformly from $\mathbb{T}$ such that $(\tilde{x}, y) \in \mathbb{T}$, and output $(x, \tilde{x})$.
7. Otherwise output x with a new element.
   (a) pick uniformly $x' \leftarrow X$ such that $(x', \tilde{y}) \notin \mathbb{T}$ for any $\tilde{y}$.
   (b) insert (x', y) in $\mathbb{T}$, and output (x, x').

Algorithm 3: Simulation method of the collision oracle.

We next review how the algorithm SPO runs on input $(\hat{x}, \hat{y})$ in detail. Since $(\hat{x}, \hat{y})$ must be in $\mathbb{T}$, the algorithm can obtain $(\hat{y}, n)$ from $\mathbb{L}$. If $n = 1$, which implies that there is only one preimage of $\hat{y}$, then the algorithm SPO returns $\perp$. Otherwise, it returns x as the algorithm CO does. The whole algorithm is presented in Algorithm 4.

Algorithm SPO($\hat{x}, \hat{y}$)

1. If $(\hat{x}, \hat{y}) \notin \mathbb{T}$, output $\perp$.
2. If $n = 1$ for $(\hat{y}, n) \in \mathbb{L}$, output $\perp$.
3. If $n \neq 1$ for $(\hat{y}, n) \in \mathbb{L}$, then compute the probability $q = \frac{\#\mathbb{T}(\hat{y})-1}{n-1}$.
4. With probability q output old x.
   (a) pick one entry $(\tilde{x}, \hat{y}) \in \mathbb{T}$ such that $\tilde{x} \neq \hat{x}$ uniformly at random, and output $\tilde{x}$.
5. Otherwise output new x.
   (a) pick $x \leftarrow X$ such that $(x, \tilde{y}) \notin \mathbb{T}$ for any $\tilde{y}$ uniformly at random.
   (b) insert $(x, \hat{y})$ in $\mathbb{T}$, and output x.

Algorithm 4: Simulation method of the second-preimage oracle.

Finally, we review how the algorithm FPO runs on input $\hat{y}$ in detail. If $\hat{y}$ is not yet determined (i.e., the number n of preimages of $\hat{y}$ is not determined either), then the algorithm FPO first defines the number n of preimages of $\hat{y}$. If $n = 0$, which implies that there is no preimage of $\hat{y}$, then the algorithm FPO returns $\perp$. Otherwise, the algorithm FPO returns x as the algorithm CO does. Note that the ratio in this case is not equal to that in the algorithms CO and SPO. The whole algorithm is presented in Algorithm 5.

Algorithm FPO($\hat{y}$)

1. If $(\hat{y}, n) \notin \mathbb{L}$ for any n, then pick $n \leftarrow \mathrm{B}_N(\#X - \sum_{(\tilde{y},\tilde{n})\in\mathbb{L}} \tilde{n}, \frac{1}{\#Y - \#\mathbb{L}})$, and insert $(\hat{y}, n) \in \mathbb{L}$, then output $\perp$.
2. If $n \neq 0$ for $(\hat{y}, n) \in \mathbb{L}$, then compute the probability $q = \frac{\#\mathbb{T}(\hat{y})}{n}$.
3. With probability q output old x.
   (a) pick one entry uniformly from $\mathbb{T}$ such that $(\tilde{x}, \hat{y}) \in \mathbb{T}$, and output $\tilde{x}$.
4. Otherwise output new x.
   (a) pick uniformly $x \leftarrow X$ such that $(x, \tilde{y}) \notin \mathbb{T}$.
   (b) insert $(x, \hat{y})$ in $\mathbb{T}$, and output x.

Algorithm 5: Simulation method of the first-preimage oracle.

B Proof of Lemma 2.4

We now start the proof of our main lemma.

Proof. In order to bound the statistical distance, we consider the algorithm RO instead of considering the random oracle $\mathcal{RO}^H$. It makes no difference because the distribution of the outputs of the algorithm RO is identical to the distribution of the outputs of the random oracle $\mathcal{RO}^H$ by Proposition 2.6.

We denote by "*old y*" the value y which already appeared in the interaction with the two oracles, i.e., the correspondence (x, y) is already determined. We denote by "*new y*" the value y which did not yet appear. Furthermore, we use the same notation BAD as in Lemma 2.3. That is, BAD denotes the event that there is some y such that the number of preimages of y, n_y, is larger than L.

Now, we evaluate the probability $\Pr[V_{\mathcal{A},H}(x) = y]$ according to the algorithm RO.

Case 1: *new y*:

$$\Pr[V_{\mathcal{A},H}(x) = \textit{new } y] = \frac{\#X - \sum n}{\#X - \#\mathbb{T}} \cdot \frac{1}{\#Y - \#\mathbb{L}}.$$

Let n_y be the number of preimages of y under the function h. Then conditioned on $n_y \leq L$ for all y (i.e.,

the event BAD does not occur), this probability is bounded by

$$\begin{aligned}
p_{\mathsf{low}} &= \frac{\#X - qL}{\#X} \cdot \frac{1}{\#Y} \\
&\leq \Pr[V_{\mathcal{A},H}(x) = \textit{new } y \mid \neg\mathsf{BAD}] \\
&\leq \frac{\#X}{\#X - q} \cdot \frac{1}{\#Y - q} \\
&\leq \frac{1}{\#Y} \cdot \frac{1}{1 - \frac{q}{\#X}} \cdot \frac{1}{1 - \frac{q}{\#Y}} \\
&\leq \frac{1}{\#Y}\left(1 + \frac{2q}{\#X}\right)\left(1 + \frac{2q}{\#Y}\right) \\
&= p_{\mathsf{up}}.
\end{aligned}$$

Then, for *new y* we have

$$\begin{aligned}
&\left|\Pr[V_{\mathcal{A},H}(x) = y] - \Pr[U_Y = y]\right| \\
&\leq \left|\Pr[V_{\mathcal{A},H}(x) = y \mid \neg\mathsf{BAD}] - \Pr[U_Y = y \mid \neg\mathsf{BAD}]\right| + \Pr[\mathsf{BAD}] \\
&\leq \left|\Pr[V_{\mathcal{A},H}(x) = y \mid \neg\mathsf{BAD}] - \frac{1}{\#Y}\right| + \Pr[\mathsf{BAD}] \\
&\leq p_{\mathsf{up}} - p_{\mathsf{low}} + \Pr[\mathsf{BAD}] \\
&\leq \frac{1}{\#Y}\left(\left(1 + \frac{2q}{\#X}\right)\left(1 + \frac{2q}{\#Y}\right) - \left(1 - \frac{qL}{\#X}\right)\right) + \Pr[\mathsf{BAD}] \\
&= \frac{1}{\#Y}\left(\frac{q(L+2)}{\#X} + \frac{2q}{\#Y} + \frac{4q^2}{\#X\#Y}\right) + \Pr[\mathsf{BAD}].
\end{aligned}$$

Case 2: *old y*:

$$\Pr[V_{\mathcal{A},H}(x) = \textit{old } y] = \frac{n - \#\mathbb{T}(y)}{\#X - \#\mathbb{T}}.$$

Then conditioned on $n_y \leq L$ for all y (i.e., the event BAD does not occur), this probability is bounded by,

$$\Pr[V_{\mathcal{A},H}(x) = \textit{old } y \mid \neg\mathsf{BAD}] \leq \frac{L}{\#X - q}.$$

Then, for *old y* we have

$$\begin{aligned}
&\left|\Pr[V_{\mathcal{A},H}(x) = y] - \Pr[U_Y = y]\right| \\
&\leq \left|\Pr[V_{\mathcal{A},H}(x) = y \mid \neg\mathsf{BAD}] - \Pr[U_Y = y \mid \neg\mathsf{BAD}]\right| + \Pr[\mathsf{BAD}] \\
&\leq \left|\Pr[V_{\mathcal{A},H}(x) = y \mid \neg\mathsf{BAD}] - \frac{1}{\#Y}\right| + \Pr[\mathsf{BAD}] \\
&\leq \Pr[V_{\mathcal{A},H}(x) = y \mid \neg\mathsf{BAD}] + \frac{1}{\#Y} + \Pr[\mathsf{BAD}] \\
&\leq \frac{L}{\#X - q} + \frac{1}{\#Y} + \Pr[\mathsf{BAD}].
\end{aligned}$$

Now we can bound the statistical distance between the distribution on $H_{\mathcal{A},h}(x)$ and the uniform

distribution on Y as follows:

$$\begin{aligned}
&\sum_y \left|\Pr[V_{\mathcal{A},H}(x)=y]-\Pr[U_Y=y]\right| \\
&= \sum_{new\ y} \left|\Pr[V_{\mathcal{A},H}(x)=y]-\Pr[U_Y=y]\right| + \sum_{old\ y} \left|\Pr[V_{\mathcal{A},H}(x)=y]-\Pr[U_Y=y]\right| \\
&\le \left(\frac{q(L+2)}{\#X}+\frac{2q}{\#Y}+\frac{4q^2}{\#X\#Y}\right)+\frac{qL}{\#X-q}+\frac{q}{\#Y}+\#Y\cdot\Pr[\mathsf{BAD}] \\
&\le \frac{2q}{\#X}+\frac{3q}{\#Y}+\frac{4q^2}{\#X\#Y}+\frac{2qL}{\#X-q}+\#Y\cdot\Pr[\mathsf{BAD}].
\end{aligned}$$

Finally by applying the lemma above, if $\#X \ge \#Y$ then we have

$$\begin{aligned}
&\sum_y \left|\Pr[V_{\mathcal{A},H}(x)=y]-\Pr[U_Y=y]\right| \\
&\le \frac{2q}{\#X}+\frac{3q}{\#Y}+\frac{4q^2}{\#X\#Y}+\frac{2q}{\#X-q}\cdot\frac{5\ln\#Y}{\ln\ln\#Y}\cdot\frac{\#X}{\#Y}+\frac{1}{\#Y} \\
&\le \frac{1}{\#Y}\left(5q+1+\frac{4q^2}{\#Y}+4q\cdot\frac{5\ln\#Y}{\ln\ln\#Y}\right).
\end{aligned}$$

Otherwise we have

$$\begin{aligned}
&\sum_y \left|\Pr[V_{\mathcal{A},H}(x)=y]-\Pr[U_Y=y]\right| \\
&\le \frac{2q}{\#X}+\frac{3q}{\#Y}+\frac{4q^2}{\#X\#Y}+\frac{2q}{\#X-q}\cdot\frac{5\ln\#Y}{\ln\ln\#Y}+\frac{1}{\#Y} \\
&\le \frac{1}{\#X}\left(5q+1+\frac{4q^2}{\#X}+4q\cdot\frac{5\ln\#Y}{\ln\ln\#Y}\right).
\end{aligned}$$

Therefore we have

$$\Delta(V_{\mathcal{A},H}(x),U_Y) \le \begin{cases} \frac{1}{\#Y}\left(5q+1+\frac{4q^2}{\#Y}+20q\frac{\ln\#Y}{\ln\ln\#Y}\right), & \text{if } \#X \ge \#Y, \\ \frac{1}{\#X}\left(5q+1+\frac{4q^2}{\#X}+20q\frac{\ln\#Y}{\ln\ln\#Y}\right), & \text{if } \#X < \#Y. \end{cases}$$

□

C Proof of the Security of dFO, Theorem 3.1

We prove Theorem 3.1 in the game style. We define a sequence of games and bound the advantage of the adversary in the IND-CCA2 game by showing each of the subsequent pairs of games is statistically close, and by relating the last game to the OW-CPA property of the underlying encryption scheme.

In order to prove the IND-CCA2 security, it is necessary to simulate the decryption oracle without knowing the secret key sk. This is done by using the following plaintext extractor PE as in the original proof [FO99].

The plaintext extractor PE: The plaintext extractor shares the three tables $\mathbb{T}_F$, $\mathbb{T}_G$, and $\mathbb{T}_H$ that are involved in the simulation algorithms RO_F, RO_G, and RO_H, respectively. Given a decryption query $c=(c_1,c_2)$, PE inspects each entry $(\mu, f_\mu) \in \mathbb{T}_F$, $(\gamma, g_\gamma) \in \mathbb{T}_G$, and $(f,\gamma,h_{f,\gamma}) \in \mathbb{T}_H$. For each $(\gamma, g_\gamma) \in \mathbb{T}_G$, it obtains $\mu \leftarrow c_2 \oplus g_\gamma$. It next picks $(\mu, f) \in \mathbb{T}_F$ and picks $(f,\gamma,h) \in \mathbb{T}_H$. It checks whether $c_1 = \mathsf{Enc}_{\mathsf{pk}}(\gamma,h)$. If they hold, PE outputs μ as the decryption of c and stops. Otherwise, the extractor returns $\perp$.

Sequence of games: We start with the original attack game with respect to IND-CCA2 in the ROM, and modify it step by step in order to obtain a game directly related to the adversary which breaks the OW-CPA property of $\mathcal{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- Game_0: The original attack game with respect to IND-CCA2 in the ROM. A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated by using the key generation algorithm of Gen. The adversary $\mathcal{A}$ is given the public key pk and has access to the decryption oracle $\mathcal{D}$, the random oracles $\mathcal{RO}^F$, $\mathcal{RO}^G$, and $\mathcal{RO}^H$. At some point in the game the adversary $\mathcal{A}$ is expected to output a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext c^* of m_b. This ciphertext c^* is constructed as follows:

  $$r^* \leftarrow \mathcal{M}, \qquad g^* \leftarrow \mathcal{RO}^G(r^*), \qquad c_2^* \leftarrow g^* \oplus m_b,$$
  $$f^* \leftarrow \mathcal{RO}^F(m_b), \qquad h^* \leftarrow \mathcal{RO}^H(f^*, r^*), \qquad c_1^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r^*; h^*).$$

  Then the ciphertext (c_1^*, c_2^*) is given to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit b'.

- $\mathsf{Game}_{0.5}$: We replace the random oracles $\mathcal{RO}^F$, $\mathcal{RO}^G$, and $\mathcal{RO}^H$ with the algorithms RO_F, RO_G, and RO_H respectively. These algorithms are obtained by the standard "on-the-fly" method.

  Furthermore we replace the decryption oracle $\mathcal{D}$ with the algorithm D which simply runs the decryption algorithm using secret key sk.

- Game_1: We change the time for generating r^*. The challenger first chooses r^+ uniformly at random and obtains $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_2: We modify the above game, by hooking queries to the algorithms RO_G and RO_H. If the query to the algorithm contains r^+, the challenger stops. Otherwise, the query is passed to the algorithms.

- Game_3: We make the decryption algorithm D reject an undetermined r. That is, the algorithm D outputs $\perp$ if $(r, *) \notin \mathbb{T}_G$ for $r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$ in step 2. In the after games, the algorithm D does not query to RO_G.

- Game_4: We modify the generation of g^+. The challenger uses $g^+ \leftarrow \{0,1\}^{k'}$ instead of $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_5: We modify the generation of h^+. The challenger chooses $h^+ \leftarrow \mathcal{R}$ instead of $h^+ \leftarrow \mathsf{RO}_H(\mathsf{RO}_F(m_b), r^+)$. Hence, $c^* = (c_1^*, c_2^*)$ is $(\mathsf{Enc}_{\mathsf{pk}}(r^+; h^+), g^+ \oplus m_b)$.

- Game_6: We make the decryption algorithm D reject an undetermined m. That is, D outputs $\perp$ if $(m, *) \notin \mathbb{T}_F$ in step 4. Additionally, we make D reject an undetermined (f, r). I.e., D outputs $\perp$ if $(f, r, *) \notin \mathbb{T}_H$ in step 5. Notice that the algorithm D does not query to RO_F, RO_G, and RO_H anymore.

- Game_7: Finally, we replace the decryption algorithm D with the plaintext extractor PE.

Sequence of lemmas: In the proofs of the following lemmas, we repeatedly use Lemma C.1 below in order to bound the distance of each of the subsequent pairs of games.

Lemma C.1. *Let E_1, E_2, F_1, and F_2 be events defined on a probability space. If the followings hold:*

$$|\Pr[E_1 \wedge \neg F_1] - \Pr[E_2 \wedge \neg F_2]| \leq \delta \text{ and } \Pr[F_1] = \Pr[F_2] = \epsilon,$$

then we have

$$|\Pr[E_1] - \Pr[E_2]| \leq \delta + \epsilon.$$

Proof.

$$\begin{aligned}
|\Pr[E_1] - \Pr[E_2]| &= |\Pr[E_1 \wedge \neg F_1] + \Pr[E_1 \wedge F_1] - \Pr[E_2 \wedge \neg F_2] - \Pr[E_2 \wedge F_2]| \\
&\leq |\Pr[E_1 \wedge \neg F_1] - \Pr[E_2 \wedge \neg F_2]| + |\Pr[E_1 \wedge F_1] - \Pr[E_2 \wedge F_2]| \\
&\leq \delta + \left| \Pr[E_1|F_1] \cdot \Pr[F_1] - \Pr[E_2|F_2] \cdot \Pr[F_2] \right| \\
&= \delta + \left| \Pr[E_1|F_1] - \Pr[E_2|F_2] \right| \cdot \epsilon \\
&\leq \delta + \epsilon.
\end{aligned}$$

□

Let q_F, q_G, and q_H denote the number of queries made by the adversary to the random oracles for F, G, and H, respectively. q_D denotes the number of queries made by the adversary to the decryption oracle. We denote by S_0 the event $b' = b$ in the Game_0 and use a similar notation S_i in any subsequent game. We denote by $\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}}$ the advantage of the adversary in the IND-CCA2 game in the ROM. Then by definition, we have $\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} = |2\Pr[S_0] - 1|$. We can bound this probability by the following lemmas.

In the following, we denote by AskG and AskH the event that the adversary queries r^* to the random oracle for G and the event that the adversary queries (g, r^*) to the random oracle for H for some g, respectively. Let $\mathsf{AskR} = \mathsf{AskG} \vee \mathsf{AskH}$.

Note that

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq |\Pr[S_0] - \Pr[\neg S_0]| \leq \Pr[\mathsf{AskR}_0] + |\Pr[S_0 \wedge \neg \mathsf{AskR}_0] - \Pr[\neg S_0 \wedge \neg \mathsf{AskR}_0]|.$$

Lemma C.2. Game_0 *and* $\mathsf{Game}_{0.5}$ *are identical and we have*

$$\Pr[S_0] = \Pr[S_{0.5}].$$

Proof. The algorithms RO_F, RO_G, and RO_H can simulate the random oracles for the hash functions F, G, and H, respectively. Therefore, Game_0 and $\mathsf{Game}_{0.5}$ are identical,

□

Lemma C.3. $\mathsf{Game}_{0.5}$ *and* Game_1 *are identical.*

Proof. There is no change except the timing for the generation of r^*. It is obvious that the change does not affect the games. Therefore $\mathsf{Game}_{0.5}$ and Game_1 are identical.

□

Lemma C.4. Game_1 *and* Game_2 *are identical if the event* AskR *does not occur. Hence, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_2] + |\Pr[S_2 \wedge \neg \mathsf{AskR}_2] - \Pr[\neg S_2 \wedge \neg \mathsf{AskR}_2]|.$$

Proof. If the one of two events occurs, the challenger in Game_2 stops, but continues the game in Game_1. On the other hand, if the two events does not occur, the two games are identical. Therefore, we have that

$$\Pr[\mathsf{AskR}_2] = \Pr[\mathsf{AskR}_1], \Pr[S_2 \wedge \neg \mathsf{AskR}_2] = \Pr[S_1 \wedge \neg \mathsf{AskR}_1], \Pr[\neg S_2 \wedge \neg \mathsf{AskR}_2] = \Pr[\neg S_1 \wedge \neg \mathsf{AskR}_1].$$

The inequation thus follows from the above equations.

□

Lemma C.5. Game_2 *and* Game_3 *are identical if the event* FailG_3 *does not occur, where* FailG *denotes the event that the adversary asks a valid ciphertext* c *but* r *is not contained in* $\mathbb{T}_G$ *in some decryption query in* Game_3*. Then, we have*

$$\begin{aligned} |\Pr[\mathsf{AskR}_3] - \Pr[\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\ |\Pr[S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\ |\Pr[\neg S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3]. \end{aligned}$$

In particular, we have that

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_3] + |\Pr[S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[\neg S_3 \wedge \neg\mathsf{AskR}_3]| + 3\Pr[\mathsf{FailG}_3].$$

Additionally,

$$\Pr[\mathsf{FailG}_3] \leq q_D \cdot \left(\frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma\right).$$

Proof. The first part is trivial, since the decryption algorithms are equal if FailG does not occur.

Let Fail_k denote the event that FailG_3 firstly occurs at the k-th query to the decryption oracle. Obviously, $\Pr[\mathsf{FailG}_3] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$.

Suppose that the k-th query to the decryption oracle is $c = (c_1, c_2)$ and the event Fail_k occurs. This means that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ for some h', where $(r, *) \notin \mathbb{T}_G$, and D_2 obtains $g \leftarrow \mathsf{RO}_G(r)$, $m \leftarrow c_2 \oplus g$, $f \leftarrow \mathsf{RO}_F(m)$ and $h \leftarrow \mathsf{RO}_H(f, r)$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$.

We split the event into the following four cases:

1. $(m, f) \in \mathbb{T}_F$ and $(f, r, h) \in \mathbb{T}_H$.
2. $(m, *) \notin \mathbb{T}_F$ but $(f', r, h) \in \mathbb{T}_H$ for some f'.
3. $(m, f) \in \mathbb{T}_F$ but $(f, r, *) \notin \mathbb{T}_H$.
4. $(m, *) \notin \mathbb{T}_F$ and $(*, r, *) \notin \mathbb{T}_H$.

In the case 1, for any $(m, f) \in \mathbb{T}_F$, there is the corresponding triplet $(f, r, h) \in \mathbb{T}_H$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ in the worst case. Hence, the probability that $g \leftarrow \mathsf{RO}_G(r)$ satisfies $m' = c_2 \oplus g$ for any $(m', *) \in \mathbb{T}_F$ is at least that that Fail_k occurs in this case. The probability is simply upper bounded by $q_F/2^{k'}$.

In the case 2, we assume that every triplet in $\mathbb{T}_H$ is corresponding to c_1 in the worst case, that is every triplet is in the form (f', r, h') such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$. But, since the value f is not determined, the probability that $f \leftarrow \mathsf{RO}_F(m)$ equals to the one of the elements in triplets, is at most $q_H/\#\mathcal{P}$.

In the case 3, we have simply the upper bound γ because h is not determined.

In the case 4, since h is not determined yet, we have the upper bound γ.

By summing up, we have

$$\begin{aligned} \Pr[\mathsf{Fail}_k] &\leq \frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma, \\ \Pr[\mathsf{Fail}_k] &\leq q_D \cdot \left(\frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma\right). \end{aligned}$$

□

Lemma C.6. Game_3 *and* Game_4 *are identical if the event* AskG *does not occur. In particular, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_4] + |\Pr[S_4 \wedge \neg\mathsf{AskR}_4] - \Pr[\neg S_4 \wedge \neg\mathsf{AskR}_4]| + 3\Pr[\mathsf{FailG}_3].$$

Proof. Obviously, if the event AskG does not occur, these two games are identical.

□

Lemma C.7. Game_4 *and* Game_5 *are identical if the event* AskR *does not occur. Thus, we obtain that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_5] + |\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| + 3\Pr[\mathsf{FailG}_3].$$

Proof. Obviously, if the event AskR does not occur, these two games are identical.

□

Lemma C.8. *In* Game_5*, we have*

$$|\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| = 0.$$

Proof. Since AskR_5 does not occur, the adversary $\mathcal{A}$ cannot know g^+, which is uniformly distributed over $\{0,1\}^{k'}$, and we conclude the proof.

□

Lemma C.9. Game_5 *and* Game_6 *are identical if* FailD *does not occur, where* FailD *denotes the event that* D_6 *fails in some decryption query to the decryption oracle but* D_5 *succeeds, where* D_i *denotes the decryption algorithm in* Game_i*. We have that*

$$\Pr[\mathsf{AskR}_5] \leq \Pr[\mathsf{AskR}_6] + \Pr[\mathsf{FailD}].$$

Proof. Obviously, if the event FailD does not occur, these two games are identical. Thus, we have that

$$|\Pr[\mathsf{AskR}_5] - \Pr[\mathsf{AskR}_6]| \leq \Pr[\mathsf{FailD}]$$

and the inequality in the statement.

□

Lemma C.10. *In* Game_6*, we have*

$$\Pr[\mathsf{FailD}] = q_D\left(\frac{q_H}{\#\mathcal{P}} + 2\gamma\right).$$

Proof. Let D_5 and D_6 be the decryption algorithms in Game_5 and Game_6, respectively. Let Fail_k denote D_6 firstly fails in the k-th query to the decryption oracle but D_5 succeeds. So, we have $\Pr[\mathsf{FailD}] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$.

Suppose that the k-th ciphertext $c = (c_1, c_2)$ as the decryption query.

Since D_5 succeeds, we have that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ for some $r \in \mathcal{M}$ and $h \in \mathcal{R}$, and $(r, g) \in \mathbb{T}_G$. Additionally, since g is now fixed, $m = c_2 \oplus g$ is also fixed. Moreover, we can fix $f \leftarrow \mathsf{RO}_F(m)$. Finally, we have that, for $\tilde{h} \leftarrow \mathsf{RO}_H(f, r)$, $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; \tilde{h})$, since the final check of D_5 is passed.

On the other hand, D_6 fails if f and $\tilde{h}$ are not determined.

We split Fail_k into the following three cases:

1. $(m, *) \notin \mathbb{T}_F$ and $(f, r, \tilde{h}) \in \mathbb{T}_H$ for some f,
2. $(m, f) \in \mathbb{T}_F$ for some f but $(f, r, *) \notin \mathbb{T}_H$.
3. $(m, *) \notin \mathbb{T}_F$ and $(*, r, *) \notin \mathbb{T}_H$.

In the case 1, D_5 succeeds if $(f', r, \tilde{h}) \in \mathbb{T}_H$ where $f' \leftarrow \mathsf{RO}_F(m)$. We can upper bound this probability by $q_H/\#\mathcal{P}$ since f' is chosen uniformly at random from $\mathcal{P}$.

In the case 2, D_5 succeeds if $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ where $h' \leftarrow \mathsf{RO}_H(f, r)$. This probability is at most γ because h' is chosen uniformly at random from $\mathcal{R}$.

In the case 3, D_5 succeeds if $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ where $h' \leftarrow \mathsf{RO}_H(f', r)$ and $f' \leftarrow \mathsf{RO}_F(m)$. Since h' is not determined, this probability is at most γ.

Summing up them, we have

$$\Pr[\mathsf{Fail}_k] \leq \frac{q_H}{\#\mathcal{P}} + 2\gamma$$

and conclude the proof.

□

Lemma C.11. Game_6 *and* Game_7 *are identical. We have* $\Pr[\mathsf{AskR}_6] = \Pr[\mathsf{AskR}_7]$.

Proof. Recall that the decryption algorithm D_6 in Game_6 does not query to any random oracle. Hence, we safely replace D_6 with PE.

□

Lemma C.12. *In* Game_7*, we have*

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H) \cdot \boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

Proof. We construct an adversary $\mathcal{B}$ against the $\mathsf{OW\text{-}CPA}$ security of the underlying scheme $\mathcal{PKE}$ from the adversary $\mathcal{A}$ in Game_7. The description of the new adversary $\mathcal{B}$ is as follows:

- $\mathcal{B}$ first chooses $g^+ \leftarrow \{0,1\}^{k'}$. Receiving $(\mathsf{pk}, c_1^* = \mathsf{Enc}_{\mathsf{pk}}(r^+; h^+))$ from its challenger, where $r^+ \leftarrow \mathcal{M}$ and $h^+ \leftarrow \mathcal{R}$, $\mathcal{B}$ feeds pk to $\mathcal{A}$. On decryption queries, $\mathcal{B}$ runs the plaintext extractor PE.
- Receiving m_0 and m_1 from $\mathcal{A}$, $\mathcal{B}$ generates the target ciphertext. First it queries m_0 and m_1 to RO_F to determine the hash values f_0 and f_1 of m_0 and m_1, respectively. Then, it flips a fair coin $b \leftarrow \{0,1\}$ and computes $c_2^* \leftarrow g^+ \oplus m_b$. Then, it feeds (c_1^*, c_2^*) to $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs b'. Then, $\mathcal{B}$ randomly chooses r from $\mathbb{T}_G$ and $\mathbb{T}_H$ and outputs r.

Notice that $\mathcal{B}$ simulates Game_7 perfectly and if AskR_7 occurs the one of two tables contains r^+. We have

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H)\boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

□

D Proof of the Security of wFO, Theorem 3.3

As in the previou section, we prove Theorem 3.3 in the game style. Again, in order to prove the $\mathsf{IND\text{-}CCA2}$ security, it is necessary to simulate the decryption oracle without knowing the secret key sk. This is done by using the following plaintext extractor PE as in the original proof [FO99].

The plaintext extractor PE**:** The plaintext extractor shares the three tables $\mathbb{T}_F$, $\mathbb{T}_G$, and $\mathbb{T}_H$ that are involved in the simulation algorithms for F, G, and H, respectively. Given a decryption query $c = (c_1, c_2, c_3)$, PE inspects each entry $(\mu, \sigma, f_{\mu,\sigma}) \in \mathbb{T}_F$, $(\gamma, g_\gamma) \in \mathbb{T}_G$, and $(f, \gamma, h_{f,\gamma}) \in \mathbb{T}_H$. For each $(\gamma, g_\gamma) \in \mathbb{T}_G$, it obtains $\mu \leftarrow c_2 \oplus g_\gamma$. It next picks $(\mu, c_3, f) \in \mathbb{T}_F$ and picks $(f, \gamma, h) \in \mathbb{T}_H$. It checks wheter $c_1 = \mathsf{Enc}_{\mathsf{pk}}(\gamma, h)$. If they hold, PE outputs μ as the decryption of c and stops. Otherwise, the extractor returns $\perp$.

Sequence of games: We start with the original attack game with respect to IND-CCA2 in the CT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which breaks the OW-CPA property of $\mathcal{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- Game_0: The original attack game with respect to IND-CCA2 in the CT-ROM. A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated by using the key generation algorithm of the wFO encryption scheme. The adversary $\mathcal{A}$ is given the public key pk and has access to the decryption oracle $\mathcal{D}$, the random oracles $\mathcal{RO}^F$, $\mathcal{RO}^G$, and $\mathcal{RO}^H$, and the collision oracles $\mathcal{CO}^F$, $\mathcal{CO}^G$, and $\mathcal{CO}^H$. At some point in the game the adversary $\mathcal{A}$ is expected to output a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext c^* of m_b. This ciphertext c^* is constructed as follows:

  $$\begin{array}{llll} r^* \leftarrow \mathcal{M}, & g^* \leftarrow \mathcal{RO}^G(r^*), & c_2^* \leftarrow g^* \oplus m_b, & \\ s^* \leftarrow \mathcal{S}, & f^* \leftarrow \mathcal{RO}^F(m_b, s^*), & h^* \leftarrow \mathcal{RO}^H(f^*, r^*), & c_1^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r^*; h^*). \end{array}$$

  Then the ciphertext $(c_1^*, c_2^*, c_3^* = s^*)$ is given to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit b'.

- $\mathsf{Game}_{0.5}$: We replace the oracles $\mathcal{RO}^F$, $\mathcal{RO}^G$, $\mathcal{RO}^H$, $\mathcal{CO}^F$, $\mathcal{CO}^G$, and $\mathcal{CO}^H$ with the algorithms RO_F, RO_G, RO_H, CO_F, CO_G, and CO_H respectively. These algorithms are obtained by simply modifying the algorithms RO and CO in Appendix A for F, G, and H.

  Furthermore we replace the decryption oracle $\mathcal{D}$ with the algorithm D which simply runs the decryption algorithm using the secret key sk.

- Game_1: We change the time for generating r^*. The challenger first chooses r^+ uniformly at random and obtains $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_2: We modify the above game, by hooking queries to the algorithms RO_G and RO_H. If the query to the algorithm contains r^+, the challenger stops. Otherwise, the query is passed to the algorithms. Additionally, the challenger hooks answers from the algorithms CO_G and CO_H. If the answer contains r^+, the challenger stops. Otherwise, the query is passed to the algorithms.

- Game_3: We make the decryption algorithm D reject an undetermined r. That is, the algorithm D outputs $\perp$ if $(r, *) \notin \mathbb{T}_G$ for $r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$. In the after games, the algorithm D does not query to RO_G.

- Game_4: We modify the generation of g^+. The challenger uses $g^+ \leftarrow \{0,1\}^{k'}$ instead of $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_5: We modify the generation of h^+. The challenger chooses $h^+ \leftarrow \mathcal{R}$ instead of $h^+ \leftarrow \mathsf{RO}_H(\mathsf{RO}_F(m_b, s^*), r^+)$. Hence, $c^* = (c_1^*, c_2^*, c_3^*)$ is $(\mathsf{Enc}_{\mathsf{pk}}(r^+; h^+), g^+ \oplus m_b, s^*)$, where $s^* \leftarrow \mathcal{S}$.

- Game_6: We make the decryption algorithm D reject an undetermined m. That is, D outputs $\perp$ if $(m, c_3, *) \notin \mathbb{T}_F$ in step 4. Additionally, we make D reject an undetermind (f, r). I.e., D outputs $\perp$ if $(f, r, *) \notin \mathbb{T}_H$ in step 5. In the after games, the algorithm D does not query to RO_F and RO_H.

- Game_7: Finally, we replace the decryption algorithm D with the plaintext extractor PE.

Sequence of lemmas: Let q_F, q_G, and q_H denote the number of queries made by the adversary to the oracles corresponding to F, G, and H, respectively. q_D denotes the number of queries made by the adversary to the decryption oracle. In the following, we denote by AskG^+ and AskH^+ the event that the adversary queries r^* to the random oracle for G and the event that the adversary query (f, r^*)

to the random oracle for H for some f, respectively. We denote by AskG^- and AskH^- the event that the adversary obtains r^* from the collision oracle for G and the event that the adversary obtains (f, r^*) from the collision oracle for H for some f, respectively. Let $\mathsf{AskG} = \mathsf{AskG}^+ \vee \mathsf{AskG}^-$, $\mathsf{AskH} = \mathsf{AskH}^+ \vee \mathsf{AskH}^-$, and $\mathsf{AskR} = \mathsf{AskG} \vee \mathsf{AskH}$.

Note that we have that

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq |\Pr[S_0] - \Pr[\neg S_0]| \leq \Pr[\mathsf{AskR}_0] + |\Pr[S_0 \wedge \neg\mathsf{AskR}_0] - \Pr[\neg S_0 \wedge \neg\mathsf{AskR}_0]|.$$

Lemma D.1. Game_0 *and* $\mathsf{Game}_{0.5}$ *are statistically close* [1] *and we have*

$$\Pr[S_0] = \Pr[S_{0.5}].$$

Proof. The algorithms RO_F, RO_G, and RO_H can simulate the random oracles for the hash functions F, G, and H. The algorithms CO_F, CO_G, and CO_H also can simulate the collision oracles for the hash funciotns F, G, and H, respectively. Therefore, Game_0 and $\mathsf{Game}_{0.5}$ are identical,

□

Lemma D.2. $\mathsf{Game}_{0.5}$ *and* Game_1 *are identical.*

Proof. There is no change except the timing for the generatin of r^*. It is obvious that the change does not affect the games. Therefore $\mathsf{Game}_{0.5}$ and Game_1 are identical.

□

Lemma D.3. Game_1 *and* Game_2 *are identical if the event* AskR *does not occur. Hence, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_2] + |\Pr[S_2 \wedge \neg\mathsf{AskR}_2] - \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2]|.$$

Proof. If the one of two events occur, the challenger in Game_2 stops, but continues the game in Game_1. On the other hand, if the two events does not occur, the two games are identical. Therefore, we have that

$$\Pr[\mathsf{AskR}_2] = \Pr[\mathsf{AskR}_1], \Pr[S_2 \wedge \neg\mathsf{AskR}_2] = \Pr[S_1 \wedge \neg\mathsf{AskR}_1], \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2] = \Pr[\neg S_1 \wedge \neg\mathsf{AskR}_1].$$

The inequation thus follows from the above equations.

□

Lemma D.4. Game_2 *and* Game_3 *are identical if the event* FailG_3 *does not occur, where* FailG *denotes the event that the adversary asks a valid ciphertext c but r is not contained in $\mathbb{T}_G$ in some decryption query in* Game_3*. Then, we have*

$$\begin{aligned} |\Pr[\mathsf{AskR}_3] - \Pr[\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\ |\Pr[S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\ |\Pr[\neg S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3]. \end{aligned}$$

Additionally,

$$\Pr[\mathsf{FailG}_3] \leq q_D \cdot \left(\frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + \mathsf{negl}_G + 2\mathsf{negl}_H\right),$$

[1] Although Theorem 2.8 states that there is a statistical distance, we ignore the above statistical distance, in order to simplify the analysis.

where

$$\mathsf{negl}_F \leq \begin{cases} \frac{1}{\#\mathcal{P}}\left(5q_F + 1 + \frac{4q_F^2}{\#\mathcal{P}} + 20q_F\frac{\ln \#\mathcal{P}}{\ln\ln \#\mathcal{P}}\right) & \text{if } 2^{k'}\cdot\#\mathcal{S} \geq \#\mathcal{P}, \\ \frac{1}{2^{k'}\cdot\#\mathcal{S}}\left(5q_F + 1 + \frac{4q_F^2}{2^{k'}\cdot\#\mathcal{S}} + 20q_F\frac{\ln \#\mathcal{P}}{\ln\ln \#\mathcal{P}}\right) & \text{if } 2^{k'}\cdot\#\mathcal{S} < \#\mathcal{P}, \end{cases}$$

$$\mathsf{negl}_G \leq \begin{cases} \frac{1}{2^{k'}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k'}} + 20q_G\frac{\ln 2^{k'}}{\ln\ln 2^{k'}}\right) & \text{if } \#\mathcal{M} \geq 2^{k'}, \\ \frac{1}{\#\mathcal{M}}\left(5q_G + 1 + \frac{4q_G^2}{\#\mathcal{M}} + 20q_G\frac{\ln 2^{k'}}{\ln\ln 2^{k'}}\right) & \text{if } \#\mathcal{M} < 2^{k'}, \end{cases}$$

$$\mathsf{negl}_H \leq \begin{cases} \frac{1}{\#\mathcal{R}}\left(5q_H + 1 + \frac{4q_H^2}{\#\mathcal{R}} + 20q_H\frac{\ln \#\mathcal{R}}{\ln\ln \#\mathcal{R}}\right) & \text{if } \#\mathcal{M}\cdot\#\mathcal{P} \geq \#\mathcal{R}, \\ \frac{1}{\#\mathcal{M}\cdot\#\mathcal{P}}\left(5q_H + 1 + \frac{4q_H^2}{\#\mathcal{M}\cdot\#\mathcal{P}} + 20q_H\frac{\ln \#\mathcal{R}}{\ln\ln \#\mathcal{R}}\right) & \text{if } \#\mathcal{M}\cdot\#\mathcal{P} < \#\mathcal{R}. \end{cases}$$

Proof. The first part is trivial, since the decryption algorithms are equal if FailG does not occur. To show the second part, we follow the arguments in [GMMV05] with little corrections.

Let Fail_k denote the event that FailG_3 firstly occurs at the k-th query to the decyrption oracle. Obviously, $\Pr[\mathsf{FailG}_3] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$.

Suppose that the k-th query to the decryption oracle is $c = (c_1, c_2, c_3)$ and the event Fail_k occurs. This means that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ for some h', where $(r, *) \notin \mathbb{T}_G$, and D_2 obtains $g \leftarrow \mathsf{RO}_G(r)$, $m \leftarrow c_2 \oplus g$, $f \leftarrow \mathsf{RO}_F(m, c_3)$ and $h \leftarrow \mathsf{RO}_H(f, r)$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$.

We split the event into the following four cases:

1. $(m, f) \in \mathbb{T}_F$ and $(f, r, h) \in \mathbb{T}_H$.
2. $(m, *) \notin \mathbb{T}_F$ but $(f', r, h) \in \mathbb{T}_H$ for some f'.
3. $(m, f) \in \mathbb{T}_F$ but $(f, r, *) \notin \mathbb{T}_H$.
4. $(m, *) \notin \mathbb{T}_F$ and $(*, r, *) \notin \mathbb{T}_H$.

In the case 1, for any $(m, f) \in \mathbb{T}_F$, there is the corresponding triplet $(f, r, h) \in \mathbb{T}_H$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ in the worst case. Hence, the probabity $g \leftarrow \mathsf{RO}_G(r)$ satisfies $m' = c_2 \oplus g$ for some $(m', *) \in \mathbb{T}_F$ is at least that Fail_k occurs in this case. The probaiblity is simply upper bounded by $q_F/2^{k'} + \mathsf{negl}_G$ by the help of Lemma 2.4.

In the case 2, we assume that every triplet in $\mathbb{T}_H$ is corresponding to c_1 in the worst case, that is every triplet is in the form (f', r, h') such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$. But, since the value f is not determined, the probability that f obtained by $\mathsf{RO}_F(m, c_3)$ is one of the elements in triplets, is at most $q_H/\#\mathcal{P} + \mathsf{negl}_F$.

In the case 3, we have simply the upper bound $\gamma + \mathsf{negl}_H$ because h is not determined.

In the case 4, since h is not determined yet, we have the upper bound $\gamma + \mathsf{negl}_H$.

By summing up, we have

$$\Pr[\mathsf{Fail}_k] \leq \frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + \mathsf{negl}_G + 2\mathsf{negl}_H,$$
$$\Pr[\mathsf{FailG}] \leq q_D \cdot \left(\frac{q_F}{2^{k'}} + \frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + \mathsf{negl}_G + 2\mathsf{negl}_H\right).$$

□

Lemma D.5. Game_3 *and* Game_4 *are almost identical if the event* AskG *does not occur. In particular, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_4] + |\Pr[S_4 \wedge \neg\mathsf{AskR}_4] - \Pr[\neg S_4 \wedge \neg\mathsf{AskR}_4]| + 3\Pr[\mathsf{FailG}_3] + 3\mathsf{negl}_G.$$

Proof. If the event AskG does not occur, in Game_3, $g^+ \leftarrow \mathsf{RO}_G(r^+)$ is almost uniformly at random from our weak uniformity lemma (Lemma 2.4). In Game_4, $g^+ \leftarrow \{0,1\}^{k'}$ is uniformly at random. Hence, two games differ only within negl_G.

□

Lemma D.6. Game_4 *and* Game_5 *are almost identical if the event* AskH *does not occur. In particular, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_5] + |\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| + 3\Pr[\mathsf{FailG}_3] + 3\mathsf{negl}_G + 3\mathsf{negl}_H.$$

Proof. If the event AskH does not occur, in Game_4, $h^+ \leftarrow \mathsf{RO}_H(\mathsf{RO}_F(m_b, s^*), r^+)$ is almost uniformly at random from our weak uniformity lemma (Lemma 2.4). In Game_5, $h^+ \leftarrow \mathcal{R}$ is uniformly at random. Hence, two games differ only within negl_H.

□

Lemma D.7. *In* Game_5*, we have*

$$|\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| \leq 0.$$

Proof. Since g^+ is uniformly distributed over $\{0,1\}^{k'}$ and the adversary cannot know g^+, the lemma follows. □

Lemma D.8. Game_5 *and* Game_6 *are identical if* FailD *does not occur, where* FailD *denotes the event that* D_6 *fails in some decryption query to the decryption oracle but* D_5 *suceeds, where* D_i *denotes the decryption algorithm in* Game_i*. We have that*

$$\Pr[\mathsf{AskR}_5] \leq \Pr[\mathsf{AskR}_6] + \Pr[\mathsf{FailD}].$$

Proof. It follows the argument in the proof of Theorem 3.1.

□

Lemma D.9. *In* Game_6*, we have*

$$\Pr[\mathsf{FailD}] \leq q_D \cdot \left(\frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + 2\mathsf{negl}_H\right).$$

Proof. Let Fail_k denote the D_6 firstly fails in the k-th query to the decryption oracle but D_5 succeeds. So, we have $\Pr[\mathsf{FailD}] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$.

Suppose that the k-th decryption query is the ciphertext $c = (c_1, c_2, c_3)$. Since D_5 suceeds, we have that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ for some $r \in \mathcal{M}$ and $h \in \mathcal{R}$, and $(r, g) \in \mathbb{T}_G$. Additionally, since g is now fixed, $m = c_2 \oplus g$ is also fixed. Moreover, we can fix $f \leftarrow \mathsf{RO}_F(m, c_3)$. Finally, we have that, for $\tilde{h} \leftarrow \mathsf{RO}_H(f, r)$, $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; \tilde{h})$, since the final check of D_5 is passed.

On the other hand, D_6 fails if f and $\tilde{h}$ are not determined.

We split Fail_k into the following three cases:

1. $(m, c_3, *) \notin \mathbb{T}_F$ and $(f, r, \tilde{h}) \in \mathbb{T}_H$ for some f,
2. $(m, c_3, f) \in \mathbb{T}_F$ for some f but $(f, r, *) \notin \mathbb{T}_H$.
3. $(m, c_3, *) \notin \mathbb{T}_F$ and $(*, r, *) \notin \mathbb{T}_H$.

In the case 1, D_5 succeeds if $(f', r, \tilde{h}) \in \mathbb{T}_H$ where $f' \leftarrow \mathsf{RO}_F(m, c_3)$. We can upper bound this probability by $q_H/\#\mathcal{P} + \mathsf{negl}_F$ since f' is distributed according to almost uniform distribution over $\mathcal{P}$.

In the case 2, D_5 succeeds if $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ where $h' \leftarrow \mathsf{RO}_H(f, r)$. This probability is at most $\gamma + \mathsf{negl}_H$ because h' is distributed according to almost uniform distribution over $\mathcal{R}$.

By the similar way to the above, D_5 succeeds in if $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ where $h' \leftarrow \mathsf{RO}_H(f', r)$ and $f' \leftarrow \mathsf{RO}_F(m, c_3)$.

Summing up them, we have

$$\Pr[\mathsf{Fail}_k] \leq \frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + 2\mathsf{negl}_H,$$
$$\Pr[\mathsf{FailD}] \leq q_D \cdot \left(\frac{q_H}{\#\mathcal{P}} + 2\gamma + \mathsf{negl}_F + 2\mathsf{negl}_H\right).$$

□

Lemma D.10. Game_6 *and* Game_7 *are identical. We have* $\Pr[\mathsf{AskR}_6] = \Pr[\mathsf{AskR}_7]$.

Proof. Recall that the decryption algorithm D_6 in Game_6 does not query to any random oracle. Hence, we can safely replace D_6 with PE.

□

Lemma D.11. *In* Game_7*, we have*

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H) \cdot \boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

Proof. We construct an adversary $\mathcal{B}$ against the OW-CPA security of the underlying scheme $\mathcal{PKE}$ from the adversary $\mathcal{A}$ in Game_7. The description of the new adversary $\mathcal{B}$ is as follows:

- $\mathcal{B}$ first chooses $g^+ \leftarrow \{0,1\}^{k'}$. Receiving $(\mathsf{pk}, c_1^* = \mathsf{Enc}_{\mathsf{pk}}(r^+; h^+))$ from its challenger, where $r^+ \leftarrow \mathcal{M}$ and $h^+ \leftarrow \mathcal{R}$, $\mathcal{B}$ feeds pk to $\mathcal{A}$. On decryption queries, $\mathcal{B}$ runs the plaintext extractor PE.
- Receiving m_0 and m_1 from $\mathcal{A}$, $\mathcal{B}$ generates the target ciphertext. It chooses $s^+ \leftarrow \mathcal{S}$ and queries (m_0, s^+) and (m_1, s^+) to RO_F. Then, it flips a fair coin $b \leftarrow \{0,1\}$ and computes $c_2^* \leftarrow g^+ \oplus m_b$. Then, it feeds (c_1^*, c_2^*, s^+) to $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs b'. Then, $\mathcal{B}$ randomly chooses r from $\mathbb{T}_G$ and $\mathbb{T}_H$ and outputs r.

Notice that $\mathcal{B}$ simulates Game_7 perfectly. Since AskR_7 occurs, the one of two tables contains r^+. We have

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H)\boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

□

E Proof of the Security of FO, Theorem 3.5

We prove Theorem 3.5 in the game style. In order to prove the IND-CCA2 security, it is necessary to simulate the decryption oracle without knowing the secret key sk. This is done by using the following plaintext extractor PE in the original proof [FO99].

The plaintext extractor PE**:** The plaintext extractor shares the two tables $\mathbb{T}_G$ and $\mathbb{T}_H$ that are involved in the simulation algorithms RO_G and SPO_G, and the simulation algorithms RO_H and SPO_H, respectively. Given a decryption query $c = (c_1, c_2)$, PE inspects each entry $(\gamma, g_\gamma) \in \mathbb{T}_G$ and $(\mu, \gamma, h_{\mu,\gamma}) \in \mathbb{T}_H$. For each $(\gamma, g_\gamma) \in \mathbb{T}_G$, it obtains $\mu \leftarrow c_2 \oplus g_\gamma$. It next picks $(\mu, \gamma, h) \in \mathbb{T}_H$. It checks wheter $c_1 = \mathsf{Enc}_{\mathsf{pk}}(\gamma; h)$. If they hold, PE outputs μ as the decryption of c and stops. Otherwise, PE returns $\perp$.

Sequence of games: We start with the original attack game with respect to IND-CCA2 in the SPT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which breaks the OW-CPA property of $\mathcal{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

- Game_0: The original attack game with respect to IND-CCA2 in the SPT-ROM. A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated by using the key generation algorithm Gen. The adversary $\mathcal{A}$ is given the public key pk and has access to the decryption oracle $\mathcal{D}$, the random oracles $\mathcal{RO}^G$ and $\mathcal{RO}^H$, and the second-preimage oracles $\mathcal{SPO}^G$ and $\mathcal{SPO}^H$. At some point in the game the adversary $\mathcal{A}$ is expected to output a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext c^* of m_b. This ciphertext c^* is constructed as follows:

  $$r^* \leftarrow \mathcal{M}, \qquad g^* \leftarrow \mathcal{RO}^G(r^*), \qquad c_2^* \leftarrow g^* \oplus m_b,$$
  $$h^* \leftarrow \mathcal{RO}^H(m_b, r^*), \qquad c_1^* \leftarrow \mathsf{Enc}_{\mathsf{pk}}(r^*; h^*).$$

  Then the ciphertext (c_1^*, c_2^*) is given to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit b'.

- $\mathsf{Game}_{0.5}$: We replace the oracles $\mathcal{RO}^G$, $\mathcal{RO}^H$, $\mathcal{SPO}^G$, and $\mathcal{SPO}^H$ with the algorithms RO_G, RO_H, SPO_G, and SPO_H respectively. These algorithms are obtained by simply modifying the algorithms RO and SPO in Appendix A for G and H.

  Furthermore we replace the decryption oracle $\mathcal{D}$ with the algorithm D which simply runs the decryption algorithm using the secret key sk.

- Game_1: We change the time for generating r^*. The challenger first chooses r^+ uniformly at random and obtains $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_2: We modify the above game, by hooking queries to the algorithms RO_G and RO_H. If the query to the algorithm contains r^+, the challenger stops. Otherwise, the query is passed to the algorithms. Additionally, the challenger hooks answers from the algorithms SPO_G and SPO_H. If the answer contains r^+, the challenger stops. Otherwise, the query is passed to the algorithms.

- Game_3: We make the decryption algorithm D reject an undetermined r. That is, the algorithm D outputs $\perp$ if $(r, *) \notin \mathbb{T}_G$ for $r \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_1)$. In the after games, the algorithm D does not query to RO_G.

- Game_4: We modify the generation of g^+. The challenger uses $g^+ \leftarrow \{0,1\}^{k'}$ instead of $g^+ \leftarrow \mathsf{RO}_G(r^+)$.

- Game_5: We modify the generation of h^+. The challenger chooses $h^+ \leftarrow \mathcal{R}$ instead of $h^+ \leftarrow \mathsf{RO}_H(m_b, r^+)$. Hence, $c^* = (c_1^*, c_2^*)$ is $(\mathsf{Enc}_{\mathsf{pk}}(r^+; h^+), g^+ \oplus m_b)$.

- Game_6: We make D reject an undetermind (m, r). I.e., D outputs $\perp$ if $(m, r, *) \notin \mathbb{T}_H$ in step 4. In the after games, the algorithm D does not query to and RO_H.

- Game_7: Finally, we replace the decryption algorithm D with the plaintext extractor PE.

Sequence of lemmas: Let q_G and q_H denote the number of queries made by the adversary to the oracles corresponding to G and H, respectively. q_D denotes the number of queries made by the adversary to the decryption oracle. In the following, we denote by AskG^+ and AskH^+ the event that the adversary queries r^* to the random oracle for G and the event that the adversary query (m, r^*) to the random oracle for H for some m, respectively. We denote by AskG^- and AskH^- the event that the adversary obtains r^* from the collision oracle for G and the event that the adversary obtains (m, r^*) from the collision

oracle for H for some m, respectively. Let $\mathsf{AskG} = \mathsf{AskG}^+ \vee \mathsf{AskG}^-$, $\mathsf{AskH} = \mathsf{AskH}^+ \vee \mathsf{AskH}^-$, and $\mathsf{AskR} = \mathsf{AskG} \vee \mathsf{AskH}$.

Note that we have that

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq |\Pr[S_0] - \Pr[\neg S_0]| \leq \Pr[\mathsf{AskR}_0] + |\Pr[S_0 \wedge \neg\mathsf{AskR}_0] - \Pr[\neg S_0 \wedge \neg\mathsf{AskR}_0]|.$$

Lemma E.1. Game_0 *and* $\mathsf{Game}_{0.5}$ *are statistically close* [2] *and we have*

$$\Pr[S_0] = \Pr[S_{0.5}].$$

Proof. The algorithms RO_G and RO_H can simulate the random oracles for the hash functions G and H. The algorithms SPO_G and SPO_H also can simulate the collision oracles for the hash funciotns G and H, respectively. Therefore, Game_0 and $\mathsf{Game}_{0.5}$ are identical,

□

Lemma E.2. $\mathsf{Game}_{0.5}$ *and* Game_1 *are identical.*

Proof. There is no change except the timing for the generatin of r^*. It is obvious that the change does not affect the games. Therefore $\mathsf{Game}_{0.5}$ and Game_1 are identical.

□

Lemma E.3. Game_1 *and* Game_2 *are identical if the event* AskR *does not occur. Hence, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_2] + |\Pr[S_2 \wedge \neg\mathsf{AskR}_2] - \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2]|.$$

Proof. If the one of two events occur, the challenger in Game_2 stops, but continues the game in Game_1. On the other hand, if the two events does not occur, the two games are identical. Therefore, we have that

$$\Pr[\mathsf{AskR}_2] = \Pr[\mathsf{AskR}_1], \Pr[S_2 \wedge \neg\mathsf{AskR}_2] = \Pr[S_1 \wedge \neg\mathsf{AskR}_1], \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2] = \Pr[\neg S_1 \wedge \neg\mathsf{AskR}_1].$$

The inequation thus follows from the above equations.

□

Lemma E.4. Game_2 *and* Game_3 *are identical if the event* FailG_3 *does not occur, where* FailG *denotes the event that the adversary asks a valid ciphertext c but r is not contained in $\mathbb{T}_G$ in some decryption query in* Game_3*. Then, we have*

$$\begin{aligned}
|\Pr[\mathsf{AskR}_3] - \Pr[\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\
|\Pr[S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3], \\
|\Pr[\neg S_3 \wedge \neg\mathsf{AskR}_3] - \Pr[\neg S_2 \wedge \neg\mathsf{AskR}_2]| &\leq \Pr[\mathsf{FailG}_3].
\end{aligned}$$

Additionally,

$$\Pr[\mathsf{FailG}_3] \leq q_D \cdot \left(\frac{q_H}{2^{k'}} + \gamma + \mathsf{negl}_G + \mathsf{negl}_H\right),$$

where

$$\mathsf{negl}_G \leq \begin{cases} \frac{1}{2^{k'}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k'}} + 20q_G\frac{\ln 2^{k'}}{\ln\ln 2^{k'}}\right) & \text{if } \#\mathcal{M} \geq 2^{k'}, \\ \frac{1}{\#\mathcal{M}}\left(5q_G + 1 + \frac{4q_G^2}{\#\mathcal{M}} + 20q_G\frac{\ln 2^{k'}}{\ln\ln 2^{k'}}\right) & \text{if } \#\mathcal{M} < 2^{k'}, \end{cases}$$

$$\mathsf{negl}_H \leq \begin{cases} \frac{1}{\#\mathcal{R}}\left(5q_H + 1 + \frac{4q_H^2}{\#\mathcal{R}} + 20q_H\frac{\ln \#\mathcal{R}}{\ln\ln \#\mathcal{R}}\right) & \text{if } 2^{k'} \cdot \#\mathcal{M} \geq \#\mathcal{R}, \\ \frac{1}{2^{k'}\cdot\#\mathcal{M}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k'}\cdot\#\mathcal{M}} + 20q_H\frac{\ln \#\mathcal{R}}{\ln\ln \#\mathcal{R}}\right) & \text{if } 2^{k'} \cdot \#\mathcal{M} < \#\mathcal{R}. \end{cases}$$

[2] Although Theorem 2.8 states that there is a statistical distance, we ignore the above statistical distance, in order to simplify the analysis.

Proof. The first part is trivial, since the decryption algorithms are equal if FailG does not occur. To show the second part, we follow the arguments in [GMMV05] with little corrections.

Let Fail_k denote the event that FailG_3 firstly occurs at the k-th query to the decyrption oracle. Obviously, $\Pr[\mathsf{FailG}_3] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$.

Suppose that the k-th query to the decryption oracle is $c = (c_1, c_2)$ and the event Fail_k occurs. This means that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ for some h', where $(r, *) \notin \mathbb{T}_G$, and D_2 obtains $g \leftarrow \mathsf{RO}_G(r)$, $m \leftarrow c_2 \oplus g$, and $h \leftarrow \mathsf{RO}_H(m, r)$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$.

We split the event into the following two cases:

1. $(m, r, h) \in \mathbb{T}_H$.
2. $(m, r, *) \notin \mathbb{T}_H$.

In the case 1, for any $(m, r, h) \in \mathbb{T}_H$ such that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ in the worst case. Hence, the probabity that $g \leftarrow \mathsf{RO}_G(r)$ satisfies $m = c_2 \oplus g$ for $(m, r, h) \in \mathbb{T}_H$ is at least that that Fail_k occurs in this case. This probaiblity is simply upper bounded by $q_H/2^{k'} + \mathsf{negl}_G$.

In the case 2, we have the upper bound $\gamma + \mathsf{negl}_H$ because h is not determined.

By summing up, we have

$$\Pr[\mathsf{Fail}_k] \leq \frac{q_H}{2^{k'}} + \gamma + \mathsf{negl}_G + \mathsf{negl}_H,$$
$$\Pr[\mathsf{FailG}_3] \leq q_D \cdot \left(\frac{q_H}{2^{k'}} + \gamma + \mathsf{negl}_G + \mathsf{negl}_H\right).$$

□

Lemma E.5. Game_3 *and* Game_4 *are almost identical if the event* AskG *does not occur. In particular, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_4] + |\Pr[S_4 \wedge \neg\mathsf{AskR}_4] - \Pr[\neg S_4 \wedge \neg\mathsf{AskR}_4]| + 3\Pr[\mathsf{FailG}_3] + 3\mathsf{negl}_G.$$

Proof. If the event AskG does not occur, in Game_3, $g^+ \leftarrow \mathsf{RO}_G(r^+)$ is almost uniformly at random from our weak uniformity lemma (Lemma 2.4). In Game_4, $g^+ \leftarrow \{0,1\}^{k'}$ is uniformly at random. Hence, two games differ only within negl_G.

□

Lemma E.6. Game_4 *and* Game_5 *are almost identical if the event* AskH *does not occur. In particular, we have that*

$$\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} \leq \Pr[\mathsf{AskR}_5] + |\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| + 3\Pr[\mathsf{FailG}_3] + 3\mathsf{negl}_G + 3\mathsf{negl}_H.$$

Proof. If the event AskH does not occur, in Game_4, $h^+ \leftarrow \mathsf{RO}_H(m_b, r^+)$ is almost uniformly at random from our weak uniformity lemma (Lemma 2.4). In Game_5, $h^+ \leftarrow \mathcal{R}$ is uniformly at random. Hence, two games differ only within negl_H.

□

Lemma E.7. *In* Game_5*, we have*

$$|\Pr[S_5 \wedge \neg\mathsf{AskR}_5] - \Pr[\neg S_5 \wedge \neg\mathsf{AskR}_5]| = 0.$$

Proof. If AskR_5 does not occur, the adversary cannot know g^+. Hence, $m_b \oplus g^+$ is uniformly distributed and the lemma follows.

□

Lemma E.8. Game_5 *and* Game_6 *are identical if* FailD *does not occur, where* FailD *denotes the event that* D_6 *fails in some decryption query to the decryption oracle but* D_5 *suceeds, where* D_i *denotes the decryption algorithm in* Game_i*. We have that*

$$\Pr[\mathsf{AskR}_5] \leq \Pr[\mathsf{AskR}_6] + \Pr[\mathsf{FailD}].$$

Proof. It follows the argument in the proof of Theorem 3.1.

□

Lemma E.9. *In* Game_6*, we have*

$$\Pr[\mathsf{FailD}] \leq q_D \cdot (\gamma + \mathsf{negl}_H).$$

Proof. Let Fail_k denote the D_6 firstly fails in the k-th query to the decryption oracle but D_5 succeeds. So, we have $\Pr[\mathsf{FailD}] = \sum_{k=1}^{q_D} \Pr[\mathsf{Fail}_k]$. Suppose that the k-th ciphertext $c = (c_1, c_2)$ as the decryption query.

Since D_5 suceeds, we have that $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h)$ for some $r \in \mathcal{M}$ and $h \in \mathcal{R}$, and $(r, g) \in \mathbb{T}_G$. Additionally, since g is now fixed, $m = c_2 \oplus g$ is also fixed. Hence, we have that, for $\tilde{h} \leftarrow \mathsf{RO}_H(m, r)$, $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; \tilde{h})$, since the final check of D_5 is passed. On the other hand, D_6 fails if $\tilde{h}$ is not determined, that is, $(m, r, *) \notin \mathbb{T}_H$.

We can upper bound the probability that $h' \leftarrow \mathsf{RO}_H(m, r)$ and $c_1 = \mathsf{Enc}_{\mathsf{pk}}(r; h')$ by $\gamma + \mathsf{negl}_H$ sicne h' is distributed according to almost uniform distribution over $\mathcal{R}$.

Summing up them, we have

$$\Pr[\mathsf{Fail}_k] \leq \gamma + \mathsf{negl}_H,$$
$$\Pr[\mathsf{FailD}] \leq q_D \cdot (\gamma + \mathsf{negl}_H).$$

□

Lemma E.10. Game_6 *and* Game_7 *are identical. We have* $\Pr[\mathsf{AskR}_6] = \Pr[\mathsf{AskR}_7]$.

Proof. Recall that the decryption algorithm D_6 in Game_6 does not query to any random oracle. Hence, we safely replase D_6 with PE.

□

Lemma E.11. *In* Game_7*, we have*

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H) \cdot \boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

Proof. We construct an adversary $\mathcal{B}$ against the OW-CPA security of the underlying scheme $\mathcal{PKE}$ from the adversary $\mathcal{A}$ in Game_7. The description of the new adversary $\mathcal{B}$ is as follows:

- $\mathcal{B}$ first chooses $g^+ \leftarrow \{0,1\}^{k'}$. Receiving $(\mathsf{pk}, c_1^* = \mathsf{Enc}_{\mathsf{pk}}(r^+; h^+))$ from its challenger, where $r^+ \leftarrow \mathcal{M}$ and $h^+ \leftarrow \mathcal{R}$, $\mathcal{B}$ feeds pk to $\mathcal{A}$. On decryption queries, $\mathcal{B}$ runs the plaintext extractor PE.
- Receiving m_0 and m_1 from $\mathcal{A}$, $\mathcal{B}$ generates the target ciphertext. It flips a fair coin $b \leftarrow \{0,1\}$ and computes $c_2^* \leftarrow g^+ \oplus m_b$. Then, it feeds (c_1^*, c_2^*) to $\mathcal{A}$.
- Finally, $\mathcal{A}$ outputs b'. Then, $\mathcal{B}$ randomly chooses r from $\mathbb{T}_G$ and $\mathbb{T}_H$ and outputs r.

Notice that $\mathcal{B}$ simulates Game_7 perfectly and if AskR_7 occurs the one of two tables contains r^+. We have

$$\Pr[\mathsf{AskR}_7] \leq (q_G + q_H)\boldsymbol{Adv}^{\mathsf{OW}}_{\mathsf{CPA}}.$$

□

F Proof of the Security of OAEP, Theorem 3.7

Now we prove Theorem 3.7 after the original proof of the OAEP encryption scheme in the ROM [FOPS04]. In order to prove the IND-CCA2 security, it is necessary to simulate the decryption oracle without knowing the secret key sk. This is done by using the following plaintext extractor PE as in [FOPS04].

The plaintext extractor PE: The plaintext extractor shares the two tables $\mathbb{T}_G$ and $\mathbb{T}_H$ that are commonly used in the simulation algorithm $\mathsf{RO}_G, \mathsf{FPO}_G$ and $\mathsf{RO}_H, \mathsf{FPO}_H$, respectively. Given a decryption query $y = f(s,t)$, PE inspects each entry $(\gamma, g_\gamma) \in \mathbb{T}_G$ and $(\delta, h_\delta) \in \mathbb{T}_H$. For each combination of elements, it defines the following values:

$$\sigma = \delta, \quad \theta = \gamma \oplus h_\delta, \quad \mu = g_\gamma \oplus \delta,$$

and checks whether $y = f(\sigma, \theta)$ and μ has the form $m \parallel 0^{k_1}$. If both of these hold, PE outputs m as the decryption of y and stops. If no such pair is found, the extractor returns $\perp$.

Sequence of games: We start with the original attack game with respect to IND-CCA2 in the FPT-ROM, and modify it step by step in order to obtain a game directly related to the adversary which breaks the partial-domain one-wayness property of the underlying trapdoor permutation.

- Game_0: The original attack game with respect to IND-CCA2 in the FPT-ROM. A pair of keys $(\mathsf{pk}, \mathsf{sk})$ is generated by using the key generation algorithm of the OAEP encryption scheme. Let $f = f_{\mathsf{pk}}$ denote the trapdoor permutation and let $g = g_{\mathsf{sk}}$ denote its inverse. The adversary $\mathcal{A}$ is given the public key pk and has access to the decryption oracle $\mathcal{D}$, the random oracles $\mathcal{RO}^G$ and $\mathcal{RO}^H$, and the first-preimage oracles $\mathcal{FPO}^G$ and $\mathcal{FPO}^H$. At some point in the game the adversary is expected to output a pair of messages (m_0, m_1). Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext y^* of m_b. This ciphertext y^* is constructed as follows:

  $$r^* \leftarrow \{0,1\}^{k_0}, \qquad s^* \leftarrow (m_b \parallel 0^{k_1}) \oplus \mathcal{RO}^G(r^*), \qquad t^* \leftarrow r^* \oplus \mathcal{RO}^H(s^*),$$
  $$x^* \leftarrow (s^*, t^*), \qquad y^* \leftarrow f(x^*).$$

  Then the ciphertext y^* is given to $\mathcal{A}$. Finally, the adversary $\mathcal{A}$ outputs a bit b'.

- $\mathsf{Game}_{0.5}$: We replace the random oracles $\mathcal{RO}^G$ and $\mathcal{RO}^H$, and the first-preimage oracles $\mathcal{FPO}^G$ and $\mathcal{FPO}^H$ with the algorithms $\mathsf{RO}_G, \mathsf{RO}_H$, and $\mathsf{FPO}_G, \mathsf{FPO}_H$, respectively [3]. These algorithms are obtained by simply modifying the algorithms RO and FPO in Appendix A for G and H.

  Furthermore we replace the decryption oracle $\mathcal{D}$ with the algorithm D which simply runs the decryption algorithm using secret key sk

- Game_1: We modify the above game, by moving the generation of the seed r^* and the image $\mathsf{RO}_G(r^*)$ to the beginning of the game. That is, we randomly pick ahead of time some $r^+ \leftarrow \{0,1\}^{k_0}$, and use r^+ and $\mathsf{RO}_G(r^+)$ instead of r^* and $\mathsf{RO}_G(r^*)$, respectively. The game obeys the following rule:

  - Rule: $r^* = r^+$ and $s^* = (m_b \| 0^{k_1}) \oplus \mathsf{RO}_G(r^+)$. The other variables are generated as described above, i.e., $t^* = r^+ \oplus \mathsf{RO}_H(s^*)$, $x^* = s^* \parallel t^*$, and $y^* = f(x^*)$.

[3] In the proof [FOPS04] in the ROM, they implicitly make the replacement of the random oracles with the algorithms Std.RO.

- Game_2: We modify the above game, by replacing the image $\mathsf{RO}_G(r^+)$ by randomly chosen g^+ in the construction of s^*. (The challenger *does not* set the pair (r^+, g^+) in the table of G.) That is, we randomly pick ahead of time some $g^+ \leftarrow \{0,1\}^{k-k_0}$ and use g^+ instead of $\mathsf{RO}_G(r^+)$. The game obeys the following rule:

  - Rule: $r^* = r^+$ and $s^* = (m_b \| 0^{k_1}) \oplus g^+$. The other variables are generated as described above, i.e., $t^* = r^+ \oplus \mathsf{RO}_H(s^*)$, $x^* = s^* \parallel t^*$, and $y^* = f(x^*)$.

- Game_3: We now move the generation of s^* and $\mathsf{RO}_H(s^*)$ to the beginning of the game and make it independent of anything else. That is, we randomly pick a head of time some $s^+ \leftarrow \{0,1\}^{k-k_0}$, and use s^+ and $\mathsf{RO}_H(s^+)$ instead of s^* and $\mathsf{RO}_H(s^*)$, respectively. The game obeys the following rule:

  - Rule: $g^+ = (m_b \parallel 0^{k_1}) \oplus s^+$ and $t^* = r^+ \oplus \mathsf{RO}_H(s^+)$.

- Game_4: We modify the above game, by replacing the image $\mathsf{RO}_H(s^+)$ by randomly chosen h^+. (Again, the challenger *does not* set the pair (s^+, h^+) in the table of H.) That is, we randomly pick a head of time some $h^+ \leftarrow \{0,1\}^{k_0}$, and use h^+ instead of $\mathsf{RO}_H(s^+)$. The game obeys the following the rule:

  - Rule: $g^+ = (m_b \parallel 0^{k_1}) \oplus s^+$ and $t^* = r^+ \oplus h^+$.

- Game_5: Again we change the generation of the challenge ciphertext. We now pick $t^+ \leftarrow \{0,1\}^{k_0}$ and replace t^* by t^+. Then the ciphertext $y^* = f(s^+, t^+)$ is a uniformly chosen image of f.

- Game_6: We now change the decryption algorithm D. We make the decryption algorithm D reject all ciphertexts $y = f(s,t)$ such that the hash value of the corresponding $r = t \oplus \mathsf{RO}_H(s)$ has not been determined yet, i.e., r has not been previously queried to RO_G or r has not been replied by FPO_G.

  Note that from now on the decryption algorithm D does not make a new query to RO_G any more, because the necessary query has been made already.

- Game_7: We further change the decryption algorithm D. We make the decryption algorithm D additionally reject all ciphertexts $y = f(s,t)$ such that the hash value of s has not been determined yet, i.e., s has not been previously queried to RO_H or s has not been replied by FPO_H.

  Note that from now on the decryption algorithm D does not make a new query to RO_H any more, because the necessary query has been made already.

- Game_8: Now the decryption algorithm D only decrypts ciphertext y such that corresponding r and s have been already determined, respectively, and hence we can replace the decryption algorithm D by the plaintext extractor PE which perfectly simulates the decryption algorithm D without knowing the secret key sk.

Sequence of lemmas: Let q_G and q_H denote the number of queries made by the adversary to both the random oracle and the first-preimage oracle for G and H, respectively. Let q_D denote the number of queries made by the adversary to the decryption oracle. We denote by S_0 the event $b' = b$ in the Game_0 and use a similar notation S_i in any subsequent game. Furthermore, we denote by $\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}}$ the advantage of the adversary in the IND-CCA2 game in the FPT-ROM. Then by definition, we have $\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} = 2\left|\Pr[S_0] - \frac{1}{2}\right|$. We can bound this probability by the following lemmas.

Lemma F.1. Game_0 *and* $\mathsf{Game}_{0.5}$ *are statistically to close* [4]*, we have*

$$\Pr[S_0] = \Pr[S_{0.5}].$$

Proof. By Theorem 2.8, the algorithms RO_G and FPO_G and RO_H and FPO_H can simulate the random oracles and the first-preimage oracle for the hash functions G and H. Therefore, Game_0 and $\mathsf{Game}_{0.5}$ are statistically close, and we have

$$\Pr[S_0] = \Pr[S_{0.5}].$$

□

Lemma F.2. $\mathsf{Game}_{0.5}$ *and* Game_1 *are identical, and we have*

$$\Pr[S_{0.5}] = \Pr[S_1].$$

Proof. The seed r^* is independent of anything else that is appear before generating the challenge ciphertext. Therefore moving the generation of the seed r^* to the beginning of the game does not change the game. Therefore $\mathsf{Game}_{0.5}$ and Game_1 are identical, and we have

$$\Pr[S_{0.5}] = \Pr[S_1].$$

□

Lemma F.3. Game_1 *and* Game_2 *are statistically close if the hash value of* r^+ *is not determined and if* $g^\diamond = (m_{1-b} \,\|\, 0^{k_1}) \oplus s^*$ *is not queried to* FPO_G*, and we have*

$$|\Pr[S_1] - \Pr[S_2]| \le \Pr[\mathsf{AskG}_2] + \mathsf{negl}_G,$$
$$\mathsf{negl}_G = \frac{1}{2^{k_0}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80q_G\frac{k-k_0}{\log(k-k_0)}\right),$$

where $q_G = q_D + q_G$*,* AskG_2 *is the event that the hash value of* r^+ *is determined in* Game_2 *or the value* $g^\diamond$ *is queried to* FPO_G*. Furthermore, in* Game_2 *we have*

$$\Pr[S_2] = \frac{1}{2}.$$

Proof. Notice that in Game_1, the adversary could win if it queries $g^\diamond$ to FPO_G. Since $k - k_0 > k_0$ if it obtains $\perp$ with high probability, it could determine that m_{1-b} is not implanted in y^*. In Game_2 the adversary cannot win the game, since g^+ contains no information corresponding to m_b and m_{1-b}. Conditioned on the event that $g^\diamond$ is not queried to FPO_G, by Lemma 2.4, Game_1 and Game_2 are statistically close, if the hash value of r^+ is not determined, i.e., r^+ is not queried to RO_G and r^+ is not replied from FPO_G. More precisely, let AskG_2^+ denote the event that r^+ is queried to RO_G in the Game_2 by the adversary $\mathcal{A}$ or the decryption algorithm D, and let AskG_2^- denote the event that $\mathsf{RO}_G(r^+)$ or g^+ is queried from FPO_G by the adversary $\mathcal{A}$ and the reply is r^+. Additionally, let $\mathsf{AskG}_2^\diamond$ denote the event that $g^\diamond$ is queried to FPO_G in the Game_2 by the adversary $\mathcal{A}$. Furthermore, we denote $\mathsf{AskG}_2 = \mathsf{AskG}_2^+ \vee \mathsf{AskG}_2^- \vee \mathsf{AskG}_2^\diamond$. We use similar notations AskG_i^-, AskG_i^+, $\mathsf{AskG}_i^\diamond$, and AskG_i for any subsequent game. Then, from the fact that $k_0 < k - k_0$, we have

$$|\Pr[S_1|\neg\mathsf{AskG}_1] - \Pr[S_2|\neg\mathsf{AskG}_2]| \le \mathsf{negl}_G,$$

[4] Although Theorem 2.8 states that there is a statistical distance, we ignore the above statistical distance, in order to simplify the analysis.

where

$$\mathsf{negl}_G = \frac{1}{2^{k_0}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80q_G\frac{k-k_0}{\log(k-k_0)}\right),$$

and hence

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[\mathsf{AskG}_2] + \mathsf{negl}_G.$$

Furthermore, in Game_2 g^+ is just used in x^* but does not appear anywhere else in the computation. Thus, the distribution on the challenge ciphertext y^* does not depend on b, and hence $\Pr[S_2] = \frac{1}{2}$.

□

Lemma F.4. Game_2 *and* Game_3 *are identical, and we have*

$$\Pr[\mathsf{AskG}_2] = \Pr[\mathsf{AskG}_3].$$

Proof. Whereas in Game_2 g^+ is randomly chosen and s^* is defined as $s^* = (m_b \parallel 0^{k_1}) \oplus g^+$, in Game_3 s^+ is randomly chosen, and g^+ and s^* are defined as $g^+ = (m_b \parallel 0^{k_1}) \oplus s^+$ and $s^* = s^+$. Therefore the distributions of the variables are identical in both games, and hence Game_2 and Game_3 are identical. Then we have

$$\Pr[\mathsf{AskG}_2] = \Pr[\mathsf{AskG}_3].$$

□

Lemma F.5. Game_3 *and* Game_4 *are statistically close if the hash value of* s^+ *is not determined, and we have*

$$|\Pr[\mathsf{AskG}_3] - \Pr[\mathsf{AskG}_4]| \leq \Pr[\mathsf{AskH}_4] + \mathsf{negl}_H,$$
$$\mathsf{negl}_H = \frac{1}{2^{k_0}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80q_H\frac{k_0}{\log k_0}\right),$$

where $q_H = q_D + q_H$ *and* AskH_4 *is the event that the hash value of* s^+ *is determined in* Game_4*. Furthermore, in* Game_4 *we have*

$$\Pr[\mathsf{AskG}_4] \leq \frac{q_G + q_D}{2^{k_0-3}} + \Pr[\mathsf{AskG}_4^{\diamond}]$$

Proof. By Lemma 2.4, Game_3 and Game_4 are statistically close, if the hash value of s^+ is not determined, i.e., s^+ is not queried to RO_H and s^+ is not replied from FPO_H. More precisely, let AskH_4^+ denote the event that s^+ is queried to RO_H in the Game_4 by the adversary $\mathcal{A}$ or the decryption algorithm D, and let AskH_4^- denote the event that $\mathsf{RO}_H(s^+)$ or h^+ is queried from FPO_H by the adversary $\mathcal{A}$ and the reply is s^+. Furthermore, we denote $\mathsf{AskH}_4 = \mathsf{AskH}_4^+ \vee \mathsf{AskH}_4^-$. We use an similar notation $\mathsf{AskH}_i^-, \mathsf{AskH}_i^+$, and AskH_i for any subsequent game. Then, from the fact that $k_0 < k - k_0$, we have

$$|\Pr[\mathsf{AskG}_3|\neg\mathsf{AskH}_3] - \Pr[\mathsf{AskG}_4|\neg\mathsf{AskH}_4]| \leq \mathsf{negl}_H,$$

where

$$\mathsf{negl}_H = \frac{1}{2^{k_0}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80q_H\frac{k_0}{\log k_0}\right),$$

and hence

$$|\Pr[\mathsf{AskG}_3] - \Pr[\mathsf{AskG}_4]| \leq \Pr[\mathsf{AskH}_4] + \mathsf{negl}_H.$$

Furthermore, we have

$$\begin{aligned}\Pr[\mathsf{AskG}_4] &\leq \Pr[\mathsf{AskG}_4^+] + \Pr[\mathsf{AskG}_4^-] + \Pr[\mathsf{AskG}_4^\diamond] \\ &\leq \Pr[\mathsf{AskG}_4^+ | \neg\mathsf{AskG}_4^-] + 2\Pr[\mathsf{AskG}_4^-] + \Pr[\mathsf{AskG}_4^\diamond].\end{aligned}$$

Since h^+ is uniformly distributed and never revealed, $r^+ = t^* \oplus h^+$ is uniformly distributed and independent of the adversary's view. Therefore we have

$$\Pr[\mathsf{AskG}_4^+ | \neg\mathsf{AskG}_4^-] \leq \frac{q_G + q_D}{2^{k_0}}.$$

Moreover, since r^+ is uniformly distributed and independent of g^+, we have

$$\Pr[\mathsf{AskG}_4^-] \leq \frac{q_G + q_D}{2^{k_0 - 1}}.$$

Therefore we can conclude

$$\Pr[\mathsf{AskG}_4] \leq \frac{q_G + q_D}{2^{k_0 - 3}} + \Pr[\mathsf{AskG}_4^\diamond].$$

□

Lemma F.6. Game_4 *and* Game_5 *are identical, and we have*

$$\Pr[\mathsf{AskH}_4] = \Pr[\mathsf{AskH}_5] \text{ and } \Pr[\mathsf{AskG}_4^\diamond] = \Pr[\mathsf{AskG}_5^\diamond].$$

Proof. Since h^+ and r^+ are uniformly distributed and never revealed, replacing $t^* = h^+ \oplus r^+$ by t^+ does not change the game, and hence we have

$$\Pr[\mathsf{AskH}_4] = \Pr[\mathsf{AskH}_5] \text{ and } \Pr[\mathsf{AskG}_4^\diamond] = \Pr[\mathsf{AskG}_5^\diamond].$$

□

Lemma F.7. Game_5 *and* Game_6 *are statistically close, and we have*

$$\begin{aligned}|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_6]| &\leq q_D \left(\frac{1}{2^{k_1}} + \mathsf{negl}'_G \right), \\ \left|\Pr[\mathsf{AskG}_5^\diamond] - \Pr[\mathsf{AskG}_6^\diamond]\right| &\leq q_D \left(\frac{1}{2^{k_1}} + \mathsf{negl}'_G \right), \\ \mathsf{negl}'_G &= \frac{1}{2^{k_0}} \left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80 q_G \frac{k - k_0}{\log(k - k_0)} \right).\end{aligned}$$

Proof. Game_5 and Game_6 only differ if y is a valid ciphertext, and the hash value of the corresponding r is not determined. More precisely, let ValidG_6 denote the event that at the decryption query in Game_6, y is a valid ciphertext (i.e., $s \oplus \mathsf{RO}_G(r)$ has the form $m \,\|\, 0^{k_1}$), and r is not queried to RO_G and r is not replied from FPO_G. There being at most q_D decryption queries, we have

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_6]| \leq q_D \Pr[\mathsf{ValidG}_6].$$

In order to bound this probability, we consider another game $\mathsf{Game}_{6.5}$ where we change the decryption algorithm D in Game_6. In $\mathsf{Game}_{6.5}$ we replace the image $\mathsf{RO}_G(r)$ by randomly chosen $g \leftarrow \{0,1\}^{k_1}$.

By Lemma 2.4, Game_6 and $\mathsf{Game}_{6.5}$ are statistically close, if the hash value of r is not determined, i.e., r is not queried to RO_G and r is not replied from FPO_G. More precisely, let $\mathsf{askG}^+_{6.5}$ denote the event that r is queried to RO_G in the $\mathsf{Game}_{6.5}$ by the adversary $\mathcal{A}$, and let $\mathsf{askG}^-_{6.5}$ denote the event that $\mathsf{RO}_G(r)$ or g is queried from FPO_G by the adversary $\mathcal{A}$ and the reply is r. Furthermore, we denote $\mathsf{askG}_{6.5} = \mathsf{askG}^+_{6.5} \vee \mathsf{askG}^-_{6.5}$. Here we note that

$$\begin{aligned}
\Pr[\mathsf{ValidG}_6] &= \Pr[\neg\mathsf{askG}_6]\Pr[\mathsf{ValidG}_6|\neg\mathsf{askG}_6],\\
\Pr[\mathsf{ValidG}_{6.5}] &= \Pr[\neg\mathsf{askG}_{6.5}]\Pr[\mathsf{ValidG}_{6.5}|\neg\mathsf{askG}_{6.5}],\\
\Pr[\mathsf{askG}_6] &= \Pr[\mathsf{askG}_{6.5}],
\end{aligned}$$

and we have

$$\begin{aligned}
&|\Pr[\mathsf{ValidG}_6] - \Pr[\mathsf{ValidG}_{6.5}]|\\
&= \Pr[\neg\mathsf{askG}_{6.5}]\,|\Pr[\mathsf{ValidG}_6|\neg\mathsf{askG}_6] - \Pr[\mathsf{ValidG}_{6.5}|\neg\mathsf{askG}_{6.5}]|\\
&\le |\Pr[\mathsf{ValidG}_6|\neg\mathsf{askG}_6] - \Pr[\mathsf{ValidG}_{6.5}|\neg\mathsf{askG}_{6.5}]|\,.
\end{aligned}$$

Then by Lemma 2.4, we have

$$|\Pr[\mathsf{ValidG}_6] - \Pr[\mathsf{ValidG}_{6.5}]| \le \mathsf{negl}'_G,$$

where

$$\mathsf{negl}'_G = \frac{1}{2^{k_0}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80q_G\frac{k-k_0}{\log(k-k_0)}\right).$$

Furthermore, since g is uniformly distributed and never revealed, it only occurs with probability 2^{-k_1} that $s \oplus g$ has the form $m \,\|\, 0^{k_1}$. Then we have

$$\Pr[\mathsf{ValidG}_{6.5}] \le \frac{1}{2^{k_1}}.$$

Therefore we can conclude

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_6]| \le q_D\left(\frac{1}{2^{k_1}} + \mathsf{negl}'_G\right).$$

By using the same argument as in the above, we have also

$$\left|\Pr[\mathsf{AskG}^\diamond_5] - \Pr[\mathsf{AskG}^\diamond_6]\right| \le q_D\left(\frac{1}{2^{k_1}} + \mathsf{negl}'_G\right).$$

□

Lemma F.8. Game_6 *and* Game_7 *are statistically close, and we have*

$$\begin{aligned}
|\Pr[\mathsf{AskH}_6] - \Pr[\mathsf{AskH}_7]| &\le q_D\left(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H\right),\\
\left|\Pr[\mathsf{AskG}^\diamond_6] - \Pr[\mathsf{AskG}^\diamond_7]\right| &\le q_D\left(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H\right),\\
\mathsf{negl}'_H &= \frac{1}{2^{k_0}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80q_H\frac{k_0}{\log k_0}\right).
\end{aligned}$$

Proof. Game_6 and Game_7 only differ if y is a valid ciphertext, and the hash value of the corresponding r is determined, while the hash value of the corresponding s is not determined. More precisely, let ValidH_7 denote the event that at the decryption query in Game_7, y is a valid ciphertext (i.e., $s \oplus \mathsf{RO}_G(r)$ has the form $m \,\|\, 0^{k_1}$), and r is queried to RO_G or r is replied from FPO_G, while s is not queried to RO_H and s is not replied from FPO_H. There being at most q_D decryption queries, we have

$$|\Pr[\mathsf{AskH}_6] - \Pr[\mathsf{AskH}_7]| \leq q_D \Pr[\mathsf{ValidH}_7].$$

In order to bound this probability, we consider another game $\mathsf{Game}_{7.5}$ where we change the decryption algorithm D in Game_7. In $\mathsf{Game}_{7.5}$ we replace the image $\mathsf{RO}_H(s)$ by randomly chosen $s \leftarrow \{0,1\}^{k_1}$. By Lemma 2.4, Game_7 and $\mathsf{Game}_{7.5}$ are statistically close, if the hash value of s is not determined, i.e., s is not queried to RO_H and s is not replied from FPO_H. More precisely, let $\mathsf{askH}^+_{7.5}$ denote the event that s is queried to RO_H in the $\mathsf{Game}_{7.5}$ by the adversary $\mathcal{A}$, and let $\mathsf{askH}^-_{7.5}$ denote the event that $\mathsf{RO}_H(s)$ or h is queried from FPO_H by the adversary $\mathcal{A}$ and the reply is s. Furthermore, we denote $\mathsf{askH}_{7.5} = \mathsf{askH}^+_{7.5} \vee \mathsf{askH}^-_{7.5}$. Here we note that

$$\begin{aligned}
\Pr[\mathsf{ValidH}_7] &= \Pr[\neg\mathsf{askH}_7] \Pr[\mathsf{ValidH}_7|\neg\mathsf{askH}_7], \\
\Pr[\mathsf{ValidH}_{7.5}] &= \Pr[\neg\mathsf{askH}_{7.5}] \Pr[\mathsf{ValidH}_{7.5}|\neg\mathsf{askH}_{7.5}], \\
\Pr[\mathsf{askH}_7] &= \Pr[\mathsf{askH}_{7.5}],
\end{aligned}$$

and we have

$$\begin{aligned}
&|\Pr[\mathsf{ValidH}_7] - \Pr[\mathsf{ValidH}_{7.5}]| \\
&= \Pr[\mathsf{askH}_{7.5}] \, |\Pr[\mathsf{ValidH}_7|\neg\mathsf{askH}_7] - \Pr[\mathsf{ValidH}_{7.5}|\neg\mathsf{askH}_{7.5}]| \\
&\leq |\Pr[\mathsf{ValidH}_7|\neg\mathsf{askH}_7] - \Pr[\mathsf{ValidH}_{7.5}|\neg\mathsf{askH}_{7.5}]| \,.
\end{aligned}$$

Then by Lemma 2.4, we have

$$|\Pr[\mathsf{ValidH}_7] - \Pr[\mathsf{ValidH}_{7.5}]| \leq \mathsf{negl}'_H,$$

where

$$\mathsf{negl}'_H = \frac{1}{2^{k_0}} \left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80 q_H \frac{k_0}{\log k_0} \right).$$

Furthermore, since h is uniformly distributed and so is $r = h \oplus t$. Therefore the probability that r has been queried to RO_G is at most $q_G \cdot 2^{-k_0}$. Then we have

$$\Pr[\mathsf{ValidH}_{7.5}] \leq \frac{q_G}{2^{k_0}}.$$

Therefore we can conclude

$$|\Pr[\mathsf{AskH}_6] - \Pr[\mathsf{AskH}_7]| \leq q_D \left(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H \right).$$

By the similar argument to the above, we have also

$$\left|\Pr[\mathsf{AskG}^{\diamond}_6] - \Pr[\mathsf{AskG}^{\diamond}_7]\right| \leq q_D \left(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H \right).$$

□

Lemma F.9. Game_7 *and* Game_8 *are identical, and we have*

$$\Pr[\mathsf{AskH}_7] = \Pr[\mathsf{AskH}_8] \text{ and } \Pr[\mathsf{AskG}_7^{\diamond}] = \Pr[\mathsf{AskG}_8^{\diamond}].$$

Furthermore, in Game_8 *we have*

$$\Pr[\mathsf{AskH}_8], \Pr[\mathsf{AskG}_8^{\diamond}] \leq \Pr[\mathsf{AskH}_8 \vee \mathsf{AskG}_8^{\diamond}] \leq (q_H + 2q_G)\boldsymbol{Adv}^{\mathsf{PD\text{-}OW}},$$

where $\boldsymbol{Adv}^{\mathsf{PD\text{-}OW}}$ *is the success probability of the partial-domain one-wayness of the underlying trapdoor permutation* f.

Proof. In Game_7 and Game_8, the decryption algorithm D only decrypts ciphertext y such that corresponding r and s have been already determined, respectively, and hence we can replace the decryption algorithm D by the plaintext extractor which perfectly simulates the decryption algorithm D. Therefore Game_7 and Game_8 are identical, and we have

$$\Pr[\mathsf{AskH}_7] = \Pr[\mathsf{AskH}_8].$$

Furthermore, in Game_8 we do not use the secret key sk any more. By using the adversary $\mathcal{A}$ in Game_8, we can output an s such that $(s,t) = g(y^*)$ with probability at least $\Pr[\mathsf{AskH}_8 \vee \mathsf{AskG}_8^{\diamond}]/(q_H + 2q_G)$. This is done by making a list which contains the queries to RO_H, and $(m_0\|0^{k_1})\oplus g$ and $(m_1\|0^{k_1})\oplus g$, where g denotes each hash value in $\mathbb{T}_G$, and choose one element in list at random. This probability is bounded by the partial-domain one-wayness of the underlying trapdoor permutation f, and hence we have

$$\Pr[\mathsf{AskH}_8 \vee \mathsf{AskG}_8^{\diamond}] \leq (q_H + 2q_G)\boldsymbol{Adv}^{\mathsf{PD\text{-}OW}}.$$

□

Summarizing the above bounds we can conclude the theorem.

$$
\begin{aligned}
\frac{1}{2}\boldsymbol{Adv}^{\mathsf{IND}}_{\mathsf{CCA2}} &= \left|\Pr[S_0] - \frac{1}{2}\right| \leq \Pr[\mathsf{AskG}_2] + \mathsf{negl}_G,\\
\mathsf{negl}_G &= \frac{1}{2^{k_0}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80q_G\frac{k-k_0}{\log(k-k_0)}\right),\\
\Pr[\mathsf{AskG}_2] &\leq \Pr[\mathsf{AskG}_4] + \Pr[\mathsf{AskH}_4] + \mathsf{negl}_H\\
&\leq \frac{q_D+q_G}{2^{k_0-3}} + \Pr[\mathsf{AskG}_4^{\diamond}] + \Pr[\mathsf{AskH}_4] + \mathsf{negl}_H,\\
\mathsf{negl}_H &= \frac{1}{2^{k_0}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80q_H\frac{k_0}{\log k_0}\right),\\
\Pr[\mathsf{AskH}_4] &\leq \Pr[\mathsf{AskH}_6] + q_D(\frac{1}{2^{k_1}} + \mathsf{negl}'_G),\\
\Pr[\mathsf{AskG}_4^{\diamond}] &\leq \Pr[\mathsf{AskG}_6^{\diamond}] + q_D(\frac{1}{2^{k_1}} + \mathsf{negl}'_G),\\
\mathsf{negl}'_G &= \frac{1}{2^{k_0}}\left(5q_G + 1 + \frac{4q_G^2}{2^{k_0}} + 80q_G\frac{k-k_0}{\log(k-k_0)}\right),\\
\Pr[\mathsf{AskH}_6] &\leq \Pr[\mathsf{AskH}_7] + q_D(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H)\\
&\leq (q_H + 2q_G)\boldsymbol{Adv}^{\mathsf{PD\text{-}OW}} + q_D(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H),\\
\Pr[\mathsf{AskG}_6^{\diamond}] &\leq \Pr[\mathsf{AskG}_7^{\diamond}] + q_D(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H)\\
&\leq (q_H + 2q_G)\boldsymbol{Adv}^{\mathsf{PD\text{-}OW}} + q_D(\frac{q_G}{2^{k_0}} + \mathsf{negl}'_H),\\
\mathsf{negl}'_H &= \frac{1}{2^{k_0}}\left(5q_H + 1 + \frac{4q_H^2}{2^{k_0}} + 80q_H\frac{k_0}{\log k_0}\right).
\end{aligned}
$$

G Overview of Approximation Sampling Algorithms

Sampling from the binomial distribution: Let us denote the binomial distribution with parameters N and p by $\mathrm{BN}(N, p)$.

The simplest method for sampling from $\mathrm{BN}(N, p)$ can be constructed by simulating a toss of a biased coin that faces up the head with probability p as follows: (1) pick a sample u from $[0, 1)$ uniformly at random and (2) if $u \leq p$ output H, otherwise output T. Then, we toss the biased coin N times by this method, and output the number of H, which distributed according to the target distribution. However, it requires N tosses.

Relles proposed a smart idea to sample from $\mathrm{BN}(N, p)$ only with $O(\log N)$ samples [Rel72] from some other distribution. Instead of tossing N biased coins, we sample a median s of N uniform variables over $[0, 1)$. The outcome $s \leq p$ implies that at least $N/2$ coins face up the heads and the outcome $s > p$ implies that at least $N/2$ coins face up the tails. Thus, the outcomes of $N/2$ tosses can be determined by one median of N uniform variables over $[0, 1)$. Recursively performing this procedure, sampling medians in $O(\log N)$ times decides the number of the coins facing up the heads.

Let us assume that $N = 2K - 1$. It is well-known that the beta distribution with the parameter K, denoted by $\mathrm{BE}(K, K)$, coincides with the distribution of the median of N uniform variables over $[0, 1)$. Therefore, if we have a sampling algorithm for the beta distribution, we can sample from the binomial distribution efficiently.

Sampling from the beta distribution: The sampling from the beta distribution is a relatively easy task. Assume that random variables X and Y are distributed according to the gamma distributions with the parameters a and b, denoted by $\mathrm{G}_{\mathrm{A}}(a)$ and $\mathrm{G}_{\mathrm{A}}(b)$, respectively. Then, the ratio $X/(X+Y)$ has the beta distribution $\mathrm{B}_{\mathrm{E}}(a,b)$. Hence, if we have a sampling algorithm for the gamma distribution, it is sufficient to sample from the beta distribution.

Sampling from the gamma distribution: Ahrens and Dieter proposed a sampling algorithm for the gamma distribution using the Cauchy distribution in [AD74]. The algorithm is designed with the acceptance–rejection principle, which appeared in the paper by von Neumann [vN51].

As an example of the acceptance–rejection method, we see the algorithm of Ahrens and Dieter [AD74]. Let us assume that we have a sampling algorithm for the Cauchy distribution with some parameters conditioned on that the output of the sampling algorithm is positive. Let $f(x;a)$ and $g(x;a)$ denote the probability density functions of the gamma distribution with the parameter a and the conditional Cauchy distributions, respectively. Let us further suppose that there exists a good function $C(a)$ such that $f(x;a) < C(a) \cdot g(x;a)$ for any $x > 0$ and $a > 1$. The main algorithm is summarized as follows:

1. Sample x from the conditional Cauchy distribution.
2. Sample u from the uniform distribution over $[0,1)$.
3. If $u < f(x;a)/(C(a)g(x;a))$ output x. Otherwise output $\perp$.

It is easy to verify that the gamma distribution coincides with the output distribution conditioned on that the above algorithm does not output $\perp$. (For the details, see Appendix K.1.) Ahrens and Dieter studied the above algorithm and explicitly showed $C(a)$ such that for any $a > 1$ and $x > 0$, $f(x;a) < C(a)g(x;a)$.

Sampling from the Cauchy distribution: Let us denote the Cauchy distribution with the parameters (m,s) by $\mathrm{C}_{\mathrm{A}}(m,s)$. We note that if X has the distribution $\mathrm{C}_{\mathrm{A}}(0,1)$ then $sX+m$ has the distrbution $\mathrm{C}_{\mathrm{A}}(m,s)$. Therefore, we only consider $\mathrm{C}_{\mathrm{A}}(0,1)$.

We note the fact that if (X,Y) is uniformly distributed in the 2-dimensional unit disc, the distribution of the ratio X/Y coincides with $\mathrm{C}_{\mathrm{A}}(0,1)$. In order to sample a point from the uniform distribution over the disc, we use a simple rejection method: (1) sample x and y from the uniform distribution over $[-1,1)$ and (2) if $x^2+y^2 \leq 1$ output (x,y), otherwise output $\perp$.

Discretizing the distributions and the computations: All of the above algorithms are analyzed in an idealized computation model, which allows us to store and manipulate directly the real numbers. Turning to a standard computation model with bounded precisions, we must discretize the distributions and the parameters appeared in the algorithms appropriciately. In this process, we have to esitimate precisely the statistical distances from the target distribution in addition to the compuational costs and spaces. In order to estimate the statistical distance, we carefully define the sequence of the algorithms whose outputs are close to the target distributions, as frequently done in security proofs in cryptography.

Our main theorem is stated as follows.

Theorem G.1 (Informal)**.** *There is a sampling algorithm such that for any positive integer N, any $0 \leq p \leq 1$ represented by ℓ bits, and any positive real ϵ, the following properties are satisfied:*

- *the distribution of the output of the algorithm is ϵ-close to the binomial distribution* $\mathrm{B}_{\mathrm{N}}(N,p)$ *and*
- *the running time of the algorithm is a polynomial in* $\log N$, ℓ, *and* $\log(1/\epsilon)$.

We again stress that the statistical distance can be controlled by the distance parameter ϵ independently of the other parameters N and p, which is significant for cryptographic applications.

Note on the Poisson distribution: By a similar technique, we can construct an efficient algorithm of sampling the Poisson distribution. In the case of the Poisson distribution, we make use of the sampling algorithms for the binomial and the gamma distributions.

H Preliminaries for Approximation Sampling Algorithms

Let X and Y be two random variables over a set S. Let D_X and D_Y be the distributions of X and Y, respectively. We often abuse the notation of D_X, which will stand for the probability density function or probability function. The statistical distance of D_X and D_Y, denoted by $\Delta(D_X, D_Y)$, is defined to be

$$\Delta(D_X, D_Y) = \frac{1}{2} \int_{w \in S} |D_X(w) - D_Y(w)| \, dw.$$

We say D_X is ϵ-close to D_Y if $\Delta(D_X, D_Y) \leq \epsilon$.

If D is a distribution, $x \leftarrow D$ denotes that x is sampled according to D. Let S be a finite set. Let $s \leftarrow S$ denote that s is sampled from the uniform distribution on S.

For any real number x and any small positive real number $0 < \epsilon < 1$, let $R_\epsilon(x)$ be the truncating function with precision ϵ. That is, $R_\epsilon(\cdot)$ truncates an input to $\lceil -\log \epsilon \rceil$ binary places. We note that, if $x < 2^k$, the integral part of $R_\epsilon(x)$ is of k-bit length and the decimal part of it is of $\lceil -\log \epsilon \rceil$ bits. We also note that $|R_\epsilon(x) - x| < \epsilon$. In particular if $\epsilon = 2^{-k}$, $R_\epsilon(x) = 2^{-k} \lfloor 2^k x \rfloor$. For any real numbers a and $\widetilde{a}$ and any small positive real $0 < \epsilon < 1$, we say that $\widetilde{a}$ is an approximation of a with precision ϵ if $|a - \widetilde{a}| < \epsilon$.

Assume that X is a random variable over $\mathbb{R}$ and has a distribution D_X. $\widetilde{D_X}$ with precision ϵ denotes the distribution of $R_\epsilon(X)$.

$X \sim D_X$ stands for that X is a random variable according to the distribution D_X.

In the following we recall the basic facts for the computations of numbers and define the notation for the costs of calculations. We denote by $T_{\mathcal{U}}$ the computational cost to toss a fair coin.

H.1 Calculations

Arithmetic operations: We first review the precision of arithmetic operations, addition, multiplication, and division.

Theorem H.1. *For any real numbers a and b, let $\widetilde{a}$ and $\widetilde{b}$ be the approximations for a and b with precisions ϵ, respectively. We assume that $2^{-l_1} < a, \widetilde{a}, b, \widetilde{b} < 2^{l_2}$. Thus, $\widetilde{a}$ and $\widetilde{b}$ are represented by $l_2 + \log(1/\epsilon)$ bit. Then we can bound the precisions of four arithmetic operations with $\widetilde{a}$ and $\widetilde{b}$:*

- *For addition and subtraction, $\widetilde{a} + \widetilde{b}$ and $\widetilde{a} - \widetilde{b}$ are approximations of $a + b$ and $a - b$ with precision 2ϵ, respectively,*
- *for multiplication, $\widetilde{a}\widetilde{b}$ is an approximation of ab with precistion $2^{l_2+1}\epsilon$, and*
- *for division, $\widetilde{a}/\widetilde{b}$ is an approximation of a/b with precision $2^{2l_1+l_2+1}\epsilon$.*

The running times of all operations are at most $O((l_1 + l_2 + \log(1/\epsilon))^3)$, that is, $\mathrm{poly}(l_1, l_2, \log(1/\epsilon))$.

We denote by $T_{\mathcal{A}}(n), T_{\mathcal{M}}(n), T_{\mathcal{D}}(n)$ the computation costs of addition, multiplication, and division with n-bit numbers, respectively.

Square root: Let x be a positive square root of a real number a, i.e., $x^2 = a$. We assume that $0 \leq a < 2^l$ has an n-bit representation. Then we can compute an approximation $\widetilde{x}$ of x with precision $\epsilon = 2^{-l_{\mathrm{ROOT}}}$ by using a binary search with $O(l + l_{\mathrm{ROOT}})$ comparisons. We will denote by $T_{\mathcal{S}}(n, \epsilon)$ the computation cost of taking square root of an n-bit number with precision ϵ.

Logarithm: Let x be the logarithm of a real number a to the base e, i.e., $\ln a = x$. Furthermore, we assume that $2^{-l} < a < 2^l$ has a n-bit representation. By using the Taylor series of the logarihtm, we can compute an approximation $\widetilde{x}$ of x with precision ϵ_{LOG} which depends l and n.

We denote by $T_{\mathcal{L}}(n, \epsilon)$ the computation cost of taking the logarithm of an n-bit number with precision ϵ to the base e. We note the following theorem used in the analyses.

Lemma H.2. *For any positive real number a, let $\widetilde{a}$ be the approximation for a with precision ϵ, and let x and $\widetilde{x}$ be $x = \ln a$ and $\widetilde{x} = \ln \widetilde{a}$, respectively. Furthermore we assume $a, \widetilde{a} > 2^{-l}$. Then, we have the bound that $|\widetilde{x} - x| < 2^l \epsilon$.*

I Definitions of the Distributions

In this section we review the definitions of the uniform, Cauchy, gamma, beta, binomial, and Poisson distributions.

The uniform distribution: For any real numbers a and b ($a < b$), let $\mathrm{Un}(a, b)$ be the uniform distribution over the interval $[a, b)$. The probability density function of the uniform distribution $\mathrm{Un}(a, b)$ is defined as follows:

$$f_{\mathrm{Un}}(x \mid a, b) = \begin{cases} \frac{1}{b-a} & (a \leq x < b) \\ 0 & (x < a \text{ or } b \leq x) \end{cases}.$$

The Cauchy distribution: The Cauchy distribution $\mathrm{Ca}(x_0, \gamma)$ is a continuous probability distribution. It has a location parameter x_0, specifying the location of the peak of the distribution, and a scale parameter γ.

The probability density function of the Cauchy distribution is defined as follows:

$$f_{\mathrm{Ca}}(x \mid x_0, \gamma) = \frac{1}{\pi}\left(\frac{\gamma}{(x - x_0)^2 + \gamma^2}\right), \qquad (-\infty < x < \infty).$$

A Cauchy distribution with $x_0 = 0$ and $\gamma = 1$ is called the standard Cauchy distribution.

The gamma distribution: The gamma distribution $\mathrm{Ga}(\alpha, \beta)$ is a continuous probability distribution. It has a shape parameter α and a scale parameter β.

The probability density function of the gamma distribution can be expressed in terms of the gamma function as follows:

$$f_{\mathrm{Ga}}(x \mid \alpha, \beta) = \frac{1}{\Gamma(\alpha) \cdot \beta^{\alpha}} x^{\alpha - 1} e^{-x/\beta}, \qquad (0 < x < \infty).$$

where

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha - 1} e^{-x} dx, \qquad (\alpha > 0).$$

A gamma distribution with $\beta = 1$ is known as the standard gamma distribution. In this paper we simply denote standard gamma distribution with a shape parameter $a > 0$ by $\mathrm{Ga}(a)$.

The beta distribution: The beta distribution $\mathrm{BE}(\alpha,\beta)$ is a continuous probability distribution defined on the interval $[0,1]$. It has two positive shape parameters, denoted by α and β.

The probability density function of the beta distribution is defined as follows:

$$f_{\mathrm{BE}}(x \mid \alpha,\beta) = B(\alpha,\beta)^{-1}x^{\alpha-1}(1-x)^{\beta-1}, \qquad (0 < x < 1).$$

The beta function $B(\alpha,\beta)$ is defined as follows:

$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx, \qquad (\alpha,\beta > 0).$$

The binomial distribution: The binomial distribution $\mathrm{BN}(N,p)$ is the discrete probability distribution of the number of successes in a sequence of N independent 0/1 experiments, each of which yields success with probability p. Such a success/failure experiment is also called a Bernoulli experiment.

The probability function of the binomial distribution is defined as follows:

$$f_{\mathrm{BN}}(x \mid N,p) = \binom{N}{x}p^x(1-p)^{N-x}, \qquad (x = 0,1,2,\cdots,N).$$

The Poisson distribution: The Poisson distribution $\mathrm{Po}(\lambda)$ is a discrete probability distribution that expresses the probability of a number of events occurring in a fixed period of time if these events occur with a known average rate and independently of the time since the last event.

The probability function of the Poisson distribution is defined as follows:

$$f_{\mathrm{Po}}(x \mid \lambda) = \frac{\lambda^x}{x!}e^{-\lambda}, \qquad (x = 0,1,2,\cdots).$$

J Inequalities for the Distributions

J.1 Inequalities for the Cauchy Distribution

Recall that the probability density function is

$$f_{\mathrm{CA}}(x \mid x_0,\gamma) = \frac{1}{\pi}\cdot\frac{\gamma}{(x-x_0)^2+\gamma^2}, \qquad (-\infty < x < \infty).$$

We often use the integration $\int \frac{dx}{x^2+b^2} = \frac{\arctan(x/b)}{b} + C$ for any $b \neq 0$, where C denotes an integral constant.

Lemma J.1.

$$\Pr_{X\sim\mathrm{CA}(0,1)}\left[0 \le X \le 2^{-l_1}\right] \le 2^{-l_1}.$$

Proof. It follows from the fact that $f_{\mathrm{CA}}(x \mid 0,1) = \frac{1}{\pi}\frac{1}{1+x^2} < 1$ for any x. □

Lemma J.2.

$$\Pr_{X\sim\mathrm{CA}(a-1,\sqrt{2a-1})}[X < 0] \le \frac{1}{2}.$$

Proof. We perform integration.

$$\begin{aligned}\Pr_{X\sim \mathrm{CA}(a-1,\sqrt{2a-1})}[X<0] &= \int_{-\infty}^{0} f_{\mathrm{CA}}(x, a-1, \sqrt{2a-1})dx \\ &= \frac{1}{\pi}\left[\arctan\left(\frac{x-(a-1)}{\sqrt{2a-1}}\right)\right]_{-\infty}^{0} \\ &= \frac{1}{\pi}\left(\arctan\left(-\frac{a-1}{\sqrt{2a-1}}\right)+\frac{\pi}{2}\right) \\ &\le \frac{1}{\pi}(0+\pi/2) = \frac{1}{2}.\end{aligned}$$

□

Lemma J.3. *For any* $0 < \epsilon < \pi/6$,

$$\Pr_{X\sim \mathrm{CA}(0,1)}\left[X \ge \frac{2\pi^3}{9\epsilon^4}\right] \le \epsilon.$$

Proof. Let us consider the following probability:

$$\Pr_{X\sim \mathrm{CA}(0,1)}[X \ge K] = \int_{K}^{\infty} f_{\mathrm{CA}}(x \mid 0, 1)dx = \frac{1}{\pi}\left(\frac{\pi}{2} - \arctan K\right).$$

Thus, we show that $\frac{\pi}{2} - \arctan K < \epsilon$ for any $0 < \epsilon < \pi/6$ and $K > 2\pi^3/(9\epsilon^4)$; that is $K > \tan(\pi/2 - \epsilon)$.

Consider a tangent line of a function $y = \tan x$ at a point $(\theta, \tan\theta)$. The line is represented by

$$y = \tan\theta + \frac{1}{\cos^2\theta}(x-\theta).$$

Recall that the function $\tan x$ is convex for $x \in (0, \pi/2)$ and $1/\cos^2\theta > 1$. Thus, we have that $\tan\theta < \theta/\cos^2\theta$ for any $0 < \theta < \pi/2$. Replacing θ by $\pi/2 - \epsilon$, we have

$$\tan(\pi/2-\epsilon) < \frac{1}{\cos^2(\pi/2-\epsilon)}(\pi/2-\epsilon) < \frac{\pi}{2\sin^2\epsilon}.$$

We now prove that $1/\sin^2\epsilon < 4\pi^2/(9\epsilon^4)$, that is, $\sin\epsilon - 3\epsilon^2/(2\pi) > 0$ for $0 < \epsilon < \pi/6$. The derivative of $\sin\epsilon - 3\epsilon^2/(2\pi)$ is $\cos\epsilon - 3\epsilon/\pi$. Hence, for $0 < \epsilon < \pi/6$, the derivative is not negative. Replacing ϵ with 0, we have 0. Thus, we have proved the inequality. This completes the proof. □

Lemma J.4. *For any real values* $s > 1$, b, *and* $\delta > 0$,

$$\Delta(\mathrm{CA}(b, s), \mathrm{CA}(b, s+\delta)) \le 2\delta.$$

Proof. Without loss of generality, we can set $b = 0$. For a real value x, we define

$$g_s(x) = f_{\mathrm{CA}}(x \mid 0, s) - f_{\mathrm{CA}}(x \mid 0, s+\delta).$$

Calculating the function, we have

$$g_s(x) = \frac{\delta}{\pi}\frac{s(s+\delta)-x^2}{(x^2+s^2)(x^2+(s+\delta)^2)}.$$

This function changes the sign at $x = \pm\sqrt{s(s+\delta)}$. To simplify integration, we first consider the following integration:

$$\int_0^{\infty} |g_s(x)|\,dx = \int_0^{\sqrt{s(s+\delta)}} g_s(x)dx - \int_{\sqrt{s(s+\delta)}}^{\infty} g_s(x)dx.$$

The integration is equal to

$$h(s,\delta) = \frac{2}{\pi}\left(\arctan\left(\sqrt{\frac{s+\delta}{s}}\right) - \arctan\left(\sqrt{\frac{s}{s+\delta}}\right)\right).$$

Thus, the statistical distance is

$$\Delta(\mathrm{C_A}(0,s), \mathrm{C_A}(0,s+\delta)) = \frac{1}{2}\cdot 2h(s,\delta) = h(s,\delta).$$

Applying the mean-value theorem, we have

$$h(s,\delta) = \frac{2}{\pi}(\sqrt{(s+\delta)/s} - \sqrt{s/(s+\delta)})\cdot(1/(1+c^2))$$

for some $c \in (\sqrt{s/(s+\delta)}, \sqrt{(s+\delta)/s})$, where we use the fact that $(\arctan x)' = 1/(1+x^2)$. Since $2/\pi$ and $1/(1+c^2)$ are less than 1, we have $h(s,\delta) \le \sqrt{(s+\delta)/s} - \sqrt{s/(s+\delta)}$. Now, we prove that the RHS is at most 2δ. We divide the RHS into two parts:

$$\sqrt{(s+\delta)/s} - \sqrt{s/(s+\delta)} = (\sqrt{(s+\delta)/s} - 1) + (1 - \sqrt{s/(s+\delta)}).$$

The first part is at most δ. To prove this, we consider $\sqrt{(s+\delta)/s} - 1 \le \delta$. Transforming this inequality, we have $\delta(\delta + 2 - 1/s) \ge 0$. The transformed inequality holds if $\delta \ge 0$ and $s \ge 1$.

The second part is also at most δ. In order to show this, we check that $1 - \sqrt{s/(s+\delta)} \le \delta$. Transforming this, we have $s \ge (1-\delta)^2/(2-\delta)$. Thus, it holds for $0 < \delta < 1$ and $s \ge 1$. This completes the proof. □

J.2 Inequalities for the Gamma Distribution

Before stating the lemmas, we first recall that the definitions of incomplete gamma functions. (See Abramowitz and Stegun [AS72, 6.5.1].) We first review the lower and upper incomplete gamma functions: For any $a > 0$ and $K > 0$,

$$\gamma(K;a) = \int_0^K e^{-t}t^{a-1}dt,$$
$$\Gamma(K;a) = \int_K^\infty e^{-t}t^{a-1}dt.$$

We next recall the regularized lower incomplete gamma function

$$P(K;a) = \frac{1}{\Gamma(a)}\int_0^K e^{-t}t^{a-1}dt, \qquad (K>0, a>0).$$

The recurrence formula of this function is in [AS72, 6.5.21]:

$$P(K;a+1) = P(K;a) - \frac{K^a e^{-K}}{\Gamma(a+1)},$$
$$P(K;1) = 1 - e^{-K}.$$

We next define the regularized upper incomplete gamma function

$$Q(K;a) = \frac{1}{\Gamma(a)}\int_K^\infty e^{-t}t^{a-1}dt, \qquad (K>0, a>0).$$

By the recurrence formula of $P(K;a)$, we have that

$$Q(K;a+1) = Q(K;a) + \frac{K^a e^{-K}}{\Gamma(a+1)},$$
$$Q(K;1) = e^{-K}.$$

Lemma J.5. *For any positive integer a and any positive real t,*

$$\Pr_{X\sim \mathrm{GA}(a)}\left[X \le 2^{-t}\right] \le 2^{-a(t+1)}.$$

Proof. We perform an integration.

$$\begin{aligned}
\Pr_{X\sim \mathrm{GA}(a)}\left[X \le 2^{-t}\right] &= \int_0^{2^{-t}} f_{\mathrm{GA}}(x \mid a, 1)dx \\
&= \int_0^{2^{-t}} \frac{1}{\Gamma(a)} x^{a-1}e^{-x}dx \\
&\le \int_0^{2^{-t}} \frac{1}{\Gamma(a)} x^{a-1}dx \\
&= \frac{1}{\Gamma(a)} \cdot \frac{1}{a} \cdot (2^{-t})^a \\
&= \frac{2^{-at}}{a!} \\
&\le 2^{-a(t+1)}.
\end{aligned}$$

□

Lemma J.6. *For any positive integer $a > 1$ and any real $K > a$, we have that*

$$\Pr_{X\sim \mathrm{GA}(a)}[X \ge K] \le 2^{-K+1+(a-1)\log K}.$$

Proof. We have that

$$\begin{aligned}
\Pr_{X\sim \mathrm{GA}(a)}[X \ge K] &= Q(K; a) \\
&= \frac{K^{a-1}e^{-K}}{(a-1)!} + Q(K; a-1), \\
&= \frac{K^{a-1}e^{-K}}{(a-1)!} + \frac{K^{a-2}e^{-K}}{(a-2)!} + Q(K; a-2), \\
&= \dots \\
&= \sum_{i=0}^{a-1} \frac{K^i e^{-K}}{i!}.
\end{aligned}$$

Since $K > a > 1$, we have that

$$\begin{aligned}
\sum_{i=0}^{a-1} \frac{K^i e^{-K}}{i!} &\le K^{a-1}e^{-K} \sum_{i=0}^{a-1} \frac{1}{i!} \\
&\le K^{a-1}e^{-K} \sum_{i=0}^{\infty} \frac{1}{i!} \\
&\le K^{a-1}e^{-K+1},
\end{aligned}$$

where we use $\sum_{i=0}^{\infty} 1/i! = e$. Hence, we have that

$$\Pr_{X\sim \mathrm{GA}(a)}[X \ge K] \le K^{a-1}e^{-K+1} \le 2^{-(K-1)+(a-1)\log K}.$$

□

J.3 Inequality for the Beta Distribution

Lemma J.7. *For any integer $a > 0$ and any reals $0 \le p \le 1$ and $\epsilon > 0$, we have*

$$\Pr_{X \sim \mathrm{BE}(a,a)}[p \le X \le p + \epsilon] \le 4a(2a-1)^3\epsilon.$$

Proof. Let us recall that the incomplete beta function $B(t;\alpha,\beta)$ and the regularized beta function $I(t;\alpha,\beta)$, which are defined as follows:

$$B(t;\alpha,\beta) = \int_t^1 x^{\alpha-1}(1-x)^{\beta-1}dx, \qquad 0 < t < 1,$$
$$I(t;\alpha,\beta) = \frac{B(t;\alpha,\beta)}{B(\alpha,\beta)}, \qquad 0 < t < 1.$$

If α and β are integers, we have that

$$I(t;\alpha,\beta) = \sum_{i=\alpha}^{\alpha+\beta-1} \binom{\alpha+\beta-1}{i} t^i (1-t)^{\alpha+\beta-1-i}.$$

(See Abramowitz and Stegun [AS72].)

Let us estimate the target probability. We define $A = 2a - 1$ for the clarity.

$$\begin{aligned}
\Pr_{X \sim \mathrm{BE}(a,a)}[p \le X \le p + \epsilon] &= I(p+\epsilon;a,a) - I(p;a,a) \\
&= \sum_{i=a}^{A} \binom{A}{i} \left((p+\epsilon)^i(1-(p+\epsilon))^{A-i} - p^i(1-p)^{A-i}\right) \\
&\le \sum_{i=a}^{A} \binom{A}{i} \left|(p+\epsilon)^i(1-(p+\epsilon))^{A-i} - p^i(1-p)^{A-i}\right| \\
&\le \sum_{i=0}^{A} \binom{A}{i} \left|(p+\epsilon)^i(1-(p+\epsilon))^{A-i} - p^i(1-p)^{A-i}\right| \\
&= 2\Delta(\mathrm{BN}(A, p+\epsilon), \mathrm{BN}(A,p)) \\
&\le 2A^3(A+1)\epsilon,
\end{aligned}$$

where in the last inequality, we use Lemma J.9 in Appendix J.4. □

J.4 Inequalities for the Binomial Distribution

Lemma J.8. *Let p be a non-negative real value. Let N be a positive integer such that $Np \le \epsilon$. Then,*

$$\Pr_{X \sim \mathrm{BN}(N,p)}[X \ne 0] \le 2\epsilon.$$

Proof. Simply, we have that

$$\Pr_X[X = 0] = (1-p)^N \ge 1 - 2Np.$$

Hence, we obtain that

$$\Pr_X[X \ne 0] \le 2Np \le 2\epsilon.$$

□

Lemma J.9. *Let $0 \leq p \leq 1$ be a real value. Let $p_\epsilon = p + \delta$, where $|\delta| < \epsilon$. The statistical distance of two binomial distributions $\mathrm{BN}(N, p_\epsilon)$ and $\mathrm{BN}(N, p)$ is bounded by $N^3(N+1)\epsilon/2$. That is*

$$\Delta\left(\mathrm{BN}(N, p_\epsilon), \mathrm{BN}(N, p)\right) < \frac{1}{2}N^3(N+1)\epsilon.$$

Proof. Let δ be the error i.e., $p_\epsilon = p + \delta$ where $|\delta| < \epsilon$. For $0 \leq x \leq n$, we define

$$d_{N,p}(x) = \left| f_{\mathrm{BN}}(x \mid N, p+\delta) - f_{\mathrm{BN}}(x \mid N, p) \right|.$$

In order to prove the theorem, we have to show

$$\sum_{x=0}^{N} d_{N,p}(x) < N^3(N+1)\epsilon.$$

Therefore it is sufficient if $d_{N,p}(x) < N^3\epsilon$ for all x.

Case $x = 0$: We define

$$g(p) = f_{\mathrm{BN}}(0 \mid N, p) = (1-p)^N.$$

The derivative of $g(p)$ is $g'(p) = -N(1-p)^{N-1}$ and $|g'(p)| \leq N$. According to the mean value theorem, there is ξ between p and $p+\delta$ such that $g(p+\delta) - g(p) = \delta \cdot g'(\xi)$. We have

$$d_{N,p}(0) = |g(p+\delta) - g(p)| \leq \left|\delta \cdot g'(\xi)\right| \leq N\epsilon.$$

Case $x = N$: In the similar way as above, we have

$$d_{N,p}(n) \leq N\epsilon.$$

Case $x \neq 0, N$: For the fixed value x, we define

$$g(p) = f_{\mathrm{BN}}(x \mid N, p) = \binom{N}{x} p^x (1-p)^{N-x}.$$

Then the derivative of $g(p)$ is

$$\begin{aligned} g'(p) &= \binom{N}{x} p^{x-1}(1-p)^{N-x-1}(x - Np) \\ &= \frac{N}{x}\binom{N-1}{x-1} p^{x-1}(1-p)^{N-x-1}(x - Np) \\ &= N\binom{N-1}{x-1} p^{x-1}(1-p)^{N-x-1}\left(1 - \frac{Np}{x}\right). \end{aligned}$$

According to the mean value theorem, there is ξ between p and $p+\delta$ such that

$$g(p+\delta) - g(p) = \delta \cdot g'(\xi).$$

We have

$$d_{N,p}(x) = |g(p+\delta) - g(p)| \leq \left|\delta \cdot g'(\xi)\right| \leq N\epsilon\binom{N-1}{x-1}\xi^{x-1}(1-\xi)^{N-x-1}\left|1 - \frac{N\xi}{x}\right|.$$

We notice that

$$\max\left|1-\frac{N\xi}{x}\right| = \begin{cases} 1-\frac{N\xi}{N-1} < 1-\xi & \text{(case 1)} \\ -(1-N\xi) < N\xi & \text{(case 2)} \end{cases}.$$

If case 1 occurs,

$$d_{N,p}(x) \le N\epsilon\binom{N-1}{x-1}\xi^{x-1}(1-\xi)^{N-x} \le \sum_{x=1}^{N-1} N\epsilon\binom{N-1}{x-1}\xi^{x-1}(1-\xi)^{N-x} = N\epsilon.$$

If case 2 occurs,

$$\begin{aligned} d_{N,p}(x) &\le N^2\epsilon\binom{N-1}{x-1}\xi^{x}(1-\xi)^{N-x-1} \\ &= N^2\epsilon\frac{x}{N-x}\binom{N-1}{x}\xi^{x}(1-\xi)^{N-x-1} \\ &\le N^3\epsilon\binom{N-1}{x}\xi^{x}(1-\xi)^{N-x-1} \\ &\le \sum_{x=0}^{N-1} N^3\epsilon\binom{N-1}{x}\xi^{x}(1-\xi)^{N-x-1} \\ &= N^3\epsilon. \end{aligned}$$

In both cases, we have

$$d_{N,p}(x) \le N^3\epsilon$$

as required. □

K Approximation Sampling Algorithms

In this section we show the algorithms that the distributions of the outputs of these algorithms are statistically close to the discretized versions of the Cauchy, gamma, and beta distributions and the binomial distribution, respectively.

We first recall the well-known acceptance–rejection method. Next, we review the existing algorithms and analyze them.

Preliminaries: For $\epsilon = 2^{-i}$ where i is some positive integer, let $R_\epsilon(x)$ denote $2^{-i}\cdot\lfloor 2^i x\rfloor$. For a continuous distribution D, the discretized distribution $\widetilde{D}$ with precision ϵ means $R_\epsilon(D)$.

K.1 The Acceptance–Rejection Method

We recall the acceptance–rejection method, the one of the basic methodologies for sampling from non-uniform distributions. This technique is formalized by von Neumann [vN51].

Suppose that we want to sample values according to a distribution D_f over S which is defined by a probability density function $f(x)$. Assume that we can sample values according to another distribution D_g over S which is defined by a probability density function $g(x)$. If, for any $x \in S$, we have $f(x) < cg(x)$ for some $c > 1$, we can use the acceptance–rejection method in order to sample from D_f. The algorithm is as follows:

1. Sample $x \leftarrow D_g$ and $u \leftarrow \mathrm{U}_N(0,1)$.
2. If $u < f(x)/cg(x)$, output x. Otherwise output $\perp$.

In order to simplify the notation, we define $h(x) = f(x)/(cg(x))$ in this subsection. Let D^h denote the distribution of the output of the above algorithm using $h(x)$. $D^h(x)$ denotes the probability density function of the distribution D^h.

For a random variable $U \sim U(0,1)$ and $x \in S$,

$$\Pr\left[U \leq \frac{f(x)}{cg(x)}\right] = \Pr\left[U \leq \frac{f(X)}{cg(X)} \mid X = x\right] = \frac{f(x)}{cg(x)} = h(x).$$

Thus,

$$D^h(x) = \begin{cases} \frac{f(x)}{cg(x)} \cdot g(x) = \frac{f(x)}{c} & (x \in S) \\ 1 - 1/c & (x = \perp) \end{cases}.$$

Therefore, D_f coincides with the distribution of the output conditioned on that the output is not $\perp$.

The following lemmas are used in later.

Lemma K.1. *Consider the following algorithm:*

1. *Initialize $i \leftarrow 0$.*
2. *Sample $x \leftarrow D_g$ and $u \leftarrow \mathrm{U}_N(0,1)$.*
3. *If $u < f(x)/(cg(x))$, output x. If $i \geq r$ output $\perp$. Otherwise go to Step 2.*

Let D denote the output distribution of the above algorithm. Then,

$$\Delta(D, D_f) = \left(1 - \frac{1}{c}\right)^r.$$

Lemma K.2. *Let $\tilde{h}(x)$ be a function such that for any $x \in S$, $\left|h(x) - \tilde{h}(x)\right| < \epsilon$. Then, we have*

$$\Delta(D^h, D^{\tilde{h}}) \leq \epsilon.$$

Lemma K.3. *Let D^{U_N} denote the distribution of the output of the algorithm using $u \leftarrow \mathrm{U}_N(0,1)$. Let $D^{\widetilde{\mathrm{U}_N}}$ denote the distribution of the output of the algorithm using $u \leftarrow \{0,1\}^l$. Then,*

$$\Delta(D^{\mathrm{U}_N}, D^{\widetilde{\mathrm{U}_N}}) \leq 2^{-l}.$$

Proof of Lemma K.1. Since D_f coincides with the conditional distribution given that the output is not $\perp$, we have that

$$D(x) = \begin{cases} (1 - (1 - 1/c)^r) f(x) & (x \in S) \\ (1 - 1/c)^r & (x = \perp) \end{cases}.$$

To ease of notation, let δ denote $(1 - 1/c)^r$. We obtain that

$$\begin{aligned} \Delta(D, D_f) &= \frac{1}{2} \int_{x \in S \cup \{\perp\}} |D(x) - f(x)|\, dx \\ &= \frac{1}{2} \left(D(\perp) + \int_{x \in S} |D(x) - f(x)|\, dx \right) \\ &= \frac{1}{2} \left(\delta + \int_{x \in S} (1 - \delta) f(x) dx \right) \\ &= \frac{1}{2} \cdot 2\delta = \delta, \end{aligned}$$

which completes the proof. □

Proof of Lemma K.2. Let us consider the distribution $D^{\tilde{h}}$. If we use $\tilde{h}$ in the criterion, we have that for a random variable $U \sim U(0,1)$ and $x \in S$

$$\Pr[U \le \tilde{h}(x)] = \Pr[U \le \tilde{h}(X) \mid X = x] = \tilde{h}(x).$$

So,

$$\begin{aligned}
\left|D^{\tilde{h}}(\perp) - D^{h}(\perp)\right| &= \left|\left(1 - \int_{x\in S} \tilde{h}(x)g(x)dx\right) - \left(1 - \int_{x\in S} h(x)g(x)dx\right)\right| \\
&= \left|\int_{x\in S} (\tilde{h}(x) - h(x))g(x)dx\right| \\
&\le \int_{x\in S} \left|\tilde{h}(x) - h(x)\right| g(x)dx \\
&\le \int_{x\in S} \epsilon \cdot g(x)dx \\
&= \epsilon.
\end{aligned}$$

Suppose that $D^{\tilde{h}}(\perp) = D^{h}(\perp) + \delta = 1 - 1/c + \delta$. We note that $|\delta| \le \epsilon$. The probability density function of $D^{\tilde{h}}$ is

$$D^{\tilde{h}}(x) = \begin{cases} \tilde{h}(x) \cdot g(x) & (x \in S) \\ 1 - 1/c + \delta & (x = \perp) \end{cases}.$$

We obtain the inequality as follows:

$$\begin{aligned}
\Delta(D^{\tilde{h}}, D^{h}) &= \frac{1}{2}\int_{x\in S} \left|D^{\tilde{h}}(x) - D^{h}(x)\right| dx + \frac{1}{2}\left|D^{\tilde{h}}(\perp) - D^{h}(\perp)\right| \\
&\le \frac{1}{2}\int_{x\in S} \left|\tilde{h}(x) - h(x)\right| g(x)dx + \frac{\epsilon}{2} \\
&\le \frac{\epsilon}{2} + \frac{\epsilon}{2} \le \epsilon.
\end{aligned}$$

□

Proof of Lemma K.3. Let us define

$$\tilde{h}(x) = \Pr_{u\leftarrow\{0,1\}^l}[u \le h(x)].$$

The probability density function $D^{\widetilde{\mathrm{U_N}}}(x)$ for $x \in S$ is $D^{\widetilde{\mathrm{U_N}}}(x) = \tilde{h}(x)g(x)$. Notice that for any $x \in S$,

$$\left|\Pr_{u\leftarrow \mathrm{U_N}(0,1)}[u \le h(x)] - \Pr_{u\leftarrow\{0,1\}^l}[u \le h(x)]\right| \le 2^{-l}.$$

The remaining part of the proof is the same as the proof of Lemma K.2. □

K.2 Sampling from the Cauchy Distribution based on the Uniform Distribution

We adopt the algorithm CU in [Dev86, Chapter 9.5.3] with a little modification in order to discretize outputs. Let us introduce two parameters, the threshold parameter $r_{\mathsf{CU}} \in \mathbb{N}$ and the precision parameter $\epsilon_{\mathsf{CU}} = 2^{-a}$ for some positive integer a. See Figure 5 for the description of the algorithm CU.

Theorem K.4. *The distribution of the output of the algorithm* CU *is* $2^{-r_{\mathsf{CU}}}$*-close to* $\widetilde{\mathrm{C_A}}(0,1)$ *with precision* ϵ_{CU}*. That is,*

$$\Delta(\mathsf{CU}, \widetilde{\mathrm{C_A}}(0,1)) \le 2^{-r_{\mathsf{CU}}}.$$

Algorithm CU

1. Initialize $i \leftarrow 0$.
2. Take samples $u \leftarrow \mathrm{UN}(-1,1)$ and $v \leftarrow \mathrm{UN}(0,1)$.
3. Set $x \leftarrow u/v$.
4. If $u^2 + v^2 \leq 1$ then output $R_{\epsilon_{\mathsf{CU}}}(x)$. If $i \geq r_{\mathsf{CU}}$ then output $\perp$. Otherwise $i \leftarrow i+1$ and go to Step 2.

Figure 5: Algorithm CU

Algorithm DiscCU

1. Initialize $i \leftarrow 0$.
2. Take samples $u \leftarrow \{0,1\}^{l_u+1}$ and $v \leftarrow \{0,1\}^{l_u}$.
   (a) Consider as $u \in [-1,1)$ and $v \in [0,1)$.
   (b) If $|u| < 2^{-t_u}$ or $v < 2^{-t_u}$,
      i. if $i \geq r_{\mathsf{CU}}$ then output $\perp$.
      ii. Otherwise increase $i \leftarrow i+1$ and go to Step 2.
3. Set $x \leftarrow u/v$.
   (a) If $|x| < 2^{-t_{\mathrm{low}}}$ or $|x| > 2^{t_{\mathrm{high}}}$,
      i. if $i \geq r_{\mathsf{CU}}$ then output $\perp$.
      ii. Otherwise increase $i \leftarrow i+1$ and go to Step 2.
4. If $u^2 + v^2 \leq 1$ then output $R_{\epsilon_{\mathsf{CU}}}(x)$. If $i \geq r_{\mathsf{CU}}$ then output $\perp$. Otherwise increase $i \leftarrow i+1$ and go to Step 2.

Figure 6: Algorithm DiscCU

Proof. Conditioned on that CU does not output $\perp$, the conditional distribution of the output is exactly $\widetilde{\mathrm{CA}}(0,1)$ [Dev86]. Since the area of the half of the unit disk is $\pi/2$, in each iteration, the probability that the output is $\perp$ is $1 - \pi/4 \leq 1/2$. By Lemma K.1, the statistical distance is at most $2^{-r_{\mathsf{CU}}}$. □

We adapt the algorithm CU to discrete samplings. We call our algorithm DiscCU. In the algorithm (Figure 6), we use a lot of (flexible) parameters. The properties of the algorithm (e.g., precision) depend on the flexible parameters.

Theorem K.5. *The sample from the algorithm has following properties:*

- *The output length is* $\log(H_{\mathsf{CU}}) + \log(1/\epsilon_{\mathsf{CU}})$ *bits,*
- *the running time of the algorithm is at most* T_{CU}*,*
- *the absolute value of output is at most* H_{CU} *and at least* L_{CU}*, and*
- *the output distribution is* Δ_{CU}*-close to* $\widetilde{\mathrm{CA}}(0,1)$ *with precision* ϵ_{CU}*,*

where the parameters are as follows:

$$\begin{aligned}
&\epsilon_{\mathsf{CU}} > 2^{-l_u+(2t_u+1)},\\
&T_{\mathsf{CU}} \leq r_{\mathsf{CU}} \cdot O\left((2l_u+1)T_{\mathcal{U}} + T_{\mathcal{D}}(l_u) + T_{\mathcal{M}}(l_u) + T_{\mathcal{A}}(2l_u)\right),\\
&H_{\mathsf{CU}} = 2^{t_{\mathrm{high}}},\\
&L_{\mathsf{CU}} = 2^{-t_{\mathrm{low}}},\\
&\Delta_{\mathsf{CU}} \leq 2^{-r_{\mathsf{CU}}} + r_{\mathsf{CU}}(2^{-t_u+1} + 2^{-t_{\mathrm{low}}+1} + 2^{-t_{\mathrm{high}}/4+1} + 8\pi 2^{-l_u}).
\end{aligned}$$

Proof.

On ϵ_{CU}: We first estimate the precision of the output. At Step 1, we sample u and v in l_u+1 and l_u bits, respectively, which means that each u and v has precision 2^{-l_u}. Since we guarantee $2^{-t_u} \leq |u|, v$, by Theorem 1, x has precision at least $2^{-l_u+(2t_u+1)}$. In the following, we set $\epsilon_{\mathsf{CU}} > 2^{-l_u+2t_u+1}$.

On H_{CU} and L_{CU}: At Step 3-(a), we reject too large or small x. It is obvious that $2^{-t_{\text{low}}} \le |x| \le 2^{t_{\text{high}}}$. Thus, $H_{\mathsf{CU}} = 2^{t_{\text{high}}}$ and $L_{\mathsf{CU}} = 2^{-t_{\text{low}}}$.

On Δ_{CU}: In order to estimate the statistical distance, we consider the sequence of the algorithms.

$\mathsf{CU}_0()$**:** This is the algorithm $\mathsf{CU}()$ with *no repeat*.

$\mathsf{CU}_1()$**:** We add the discarding procedure to Step 3. The algorithm discards x if $|x| < 2^{-t_{\text{low}}}$ or $|x| > 2^{t_{\text{high}}}$.

$\mathsf{CU}_2()$**:** Let us define $\delta = 2^{-l_u}$. We replace the criterion $u^2 + v^2 \le 1$ with $R_\delta(u)^2 + R_\delta(v)^2 \le 1$.

$\mathsf{CU}_3()$**:** We replace the output procedure $x \leftarrow R_{\epsilon_{\mathsf{CU}}}(u/v)$ with $x \leftarrow R_{\epsilon_{\mathsf{CU}}}(R_\delta(u)/R_\delta(v))$.

$\mathsf{CU}_4()$**:** We discard bad u and v. In Step 2, if $|u| < 2^{-t_u}$ or $v < 2^{-t_u}$ then discard them.

$\mathsf{CU}_5()$**:** We revive *go to* in Steps 2, 3, and 4. This is the algorithm $\mathsf{DiscCU}()$.

We first estimate $\Delta(\mathsf{CU}_0, \mathsf{CU}_1)$, the effect of the discarding procedure in Step 3. The two lemmas in Appendix J.1 (Lemma J.1 and Lemma J.3) show that x falls into $[2^{-t_{\text{low}}}, 2^{t_{\text{high}}}]$ with high probability. By setting $2^{t_{\text{high}}} = 2\pi^3/(9\epsilon^4)$ in Lemma J.3, we have that the probability that $x > 2^{t_{\text{high}}}$ is less than $\epsilon \le 2^{-t_{\text{high}}/4}$. Thus, this discarding causes the statistical distance from the target distribution at most $2^{-t_{\text{low}}+1} + 2^{-t_{\text{high}}/4+1}$.

Next, we estimate $\Delta(\mathsf{CU}_1, \mathsf{CU}_2)$. Note that, for small positive δ, $\pi(1+2\delta)^2 - \pi \le 8\pi\delta$. Thus, the change of the criterion induces the statistical distance at most $8\pi 2^{-l_u}$.

We next estimate $\Delta(\mathsf{CU}_2, \mathsf{CU}_3)$. Since we set $\epsilon_{\mathsf{CU}} > 2^{-l_u+(2t_u+1)}$, this causes no error, that is, $\Delta(\mathsf{CU}_2, \mathsf{CU}_3) = 0$.

We also estimate $\Delta(\mathsf{CU}_3, \mathsf{CU}_4)$. If $|u|, v < 2^{-t_u}$, the algorithm CU_4 outputs $\perp$. So, the statistical distance is at most 2^{-t_u+1}.

Summing up the above discussions, the statistical distance $\Delta(\mathsf{CU}_0, \mathsf{CU}_4)$ is at most $2^{-t_u+1} + 2^{-t_{\text{low}}+1} + 2^{-t_{\text{high}}/4+1} + 8\pi 2^{-l_u}$.

Additionally, at Step 4, a rejection occurs with probability $(2-\pi/2)/2 \le 1/2$. Thus, repeating the algorithm r_{CU} times, the probability that the algorithm outputs $\perp$ is at most $2^{-r_{\mathsf{CU}}}$.

Compiling the above arguments, Δ_{CU} is at most $2^{-r_{\mathsf{CU}}} + r_{\mathsf{CU}}(2^{-t_u+1} + 2^{-t_{\text{low}}+1} + 2^{-t_{\text{high}}/4+1} + 8\pi 2^{-l_u})$. □

K.3 Sampling from the Gamma Distribution based on the Cauchy and the Uniform Distribution

We adopt the algorithm in [AD74] with slight modification. We define the functions f, g, and C which appear in the criteria:

$$
\begin{aligned}
f(x,a) &= f_{\text{GA}}(x \mid a, 1) = \frac{e^{-x}x^{a-1}}{\Gamma(a)}, \\
g(x,a) &= f_{\text{CA}}(x \mid a-1, \sqrt{2a-1}) = \frac{1}{\pi} \cdot \frac{\sqrt{2a-1}}{(x-(a-1))^2 + (2a-1)}, \\
C(a) &= \frac{\pi e^{-(a-1)}(a-1)^{a-1}\sqrt{2a-1}}{\Gamma(a)}.
\end{aligned}
$$

Ahrens and Dieter showed that $f(x,a) < C(a)g(x,a)$ for any $x > 0$ and that $\sqrt{\pi} < C(a) < \pi$ for any $a > 1$. The algorithm GC in Figure 7 is a modified version of [AD74]. (In [AD74], they inlined a subroutine sampling from the Cauchy distribution. In the algorithm, we call the subroutine explicitely.) In order to simplify arguments, we only consider the case that the parameter $a > 1$ is an integer.

Algorithm GC(a): a is an integer and larger than 1

1. Set $b \leftarrow a-1, A \leftarrow a+b$, and $s \leftarrow A^{1/2}$.
2. Initialize $i \leftarrow 0$.
3. Initialize $j \leftarrow 0$.
4. Generate $x \leftarrow \mathrm{C}_A(b, s)$.
   (a) If $x < 0$,
      i. if $j \geq r_{\mathsf{GC}}$ then output $\perp$.
      ii. Otherwise increase $j \leftarrow j+1$ and go to Step 4.
5. Generate $u \leftarrow \mathrm{U}_N(0,1)$.
6. If $u \leq f(x,a)/(C(a)g(x,a))$ output $R_{\epsilon_{\mathsf{GC}}}(x)$. If $i \geq r_{\mathsf{GC}}$ then output $\perp$ Otherwise increase $i \leftarrow i+1$ and go to Step 3.

Figure 7: Algorithm GC

Algorithm DiscGC(a)

1. Set $b \leftarrow a-1, A \leftarrow a+b$, and $s \leftarrow A^{1/2}$.
2. Initialize $i \leftarrow 0$.
3. Initialize $j \leftarrow 0$.
4. Generate $t \leftarrow$ DiscCU. Compute $x \leftarrow st+b$.
   (a) If $x < 2^{-t_x}$,
      i. if $j \geq r_{\mathsf{GC}}$ then output $\perp$.
      ii. Otherwise increase $j \leftarrow j+1$ and go to Step 4.
5. Generate $u \leftarrow \{0,1\}^{l_{\mathsf{GC}}}$. (Consider u as a real value in $[0,1)$.)
   (a) If $u < 2^{-t_{\mathsf{GC}}}$,
      i. if $i \geq r_{\mathsf{GC}}$ then output $\perp$.
      ii. Otherwise increase $i \leftarrow i+1$ and go to Step 3.
6. If $\ln u \leq b(\ln x - \ln b) - (x-b) + \ln(A + (x-b)^2) - \ln A$ output x. If $i \geq r_{\mathsf{GC}}$ then output $\perp$. Otherwise increment $i \leftarrow i+1$ and go to Step 3.

Figure 8: Algorithm DiscGC

Theorem K.6 ([AD74, p.229])**.** *The output distribution of* GC(a) *is* $(7/10)^{r_{\mathsf{GC}}}$*-close to* $\widetilde{\mathrm{G}_A}(a)$ *with precision* ϵ_{GC}.

We modify the algorithm GC to treat the precision and analyze the statistical distance. We call it DiscGC (see Figure 8). Again, we assume that an input a is an integer.

Theorem K.7. *The sample from the algorithm has following properties: If* $a \leq 2^n$ *then*

- *the output has precision* ϵ_{GC},
- *the running time of the algorithm is at most* T_{GC},
- *the output is at most* H_{GC} *and at least* L_{GC}, *and*
- *the output distribution is* Δ_{GC}*-close to* $\widetilde{\mathrm{G}_A}(a)$ *with precision* ϵ_{GC},

where the parameters are as follows:

$$
\begin{aligned}
&\epsilon_{\mathsf{GC}} > 2^{n+1} L_{\mathsf{CU}},\\
&T_{\mathsf{GC}} \leq T_{\mathcal{S}}(n+1; \epsilon_{\mathrm{ROOT}}) + r_{\mathsf{GC}} \cdot O(T_{\mathcal{L}}(z; \epsilon_{\mathrm{LOG}}) + T_{\mathcal{A}}(z) + r_{\mathsf{GC}}(T_{\mathsf{CU}} + T_{\mathcal{M}}(z) + T_{\mathcal{A}}(z) + l_{\mathsf{GC}} \cdot T_{\mathcal{U}}))\\
&H_{\mathsf{GC}} = 2^n (H_{\mathsf{CU}} + 1),\\
&L_{\mathsf{GC}} = 2^{-t_x},\\
&\Delta_{\mathsf{GC}} \leq 2^{-r_{\mathsf{GC}}/2}\\
&\qquad + r_{\mathsf{GC}}(2^{-t_{\mathsf{GC}}} + 2^{-t_x+1} + 2^{-l_{\mathsf{GC}}} + \epsilon_{\mathrm{SQRT}} + (2^{k+t_x}+1)\epsilon_{\mathsf{GC}} + (2^k + 2^{t_{\mathsf{GC}}} + 3)\epsilon_{\mathrm{LOG}} + 2^{n+1}\epsilon_{\mathsf{CU}} + \Delta_{\mathsf{CU}}),
\end{aligned}
$$

where $z = O(n + l_{\mathsf{GC}} + \log(H_{\mathsf{CU}}) + \log(1/\epsilon_{\mathsf{CU}}) + \log(1/\epsilon_{\mathrm{ROOT}}) + \log(1/\epsilon_{\mathrm{LOG}}))$.

Proof.

On ϵ_{GC}: Recall that $s = \sqrt{2a-1}$ is less than $a \le 2^n$ and so is $b = a - 1$. Additionally, b is an integer. Thus, the precision of $x = st + b$ is at most $2^{n+1}L_{\mathsf{CU}}$.

On H_{GC} and L_{GC}: Since the output of $\mathsf{DiscCU}(0,1)$ is in $[-H_{\mathsf{CU}}, H_{\mathsf{CU}}]$, x is in $[-H_{\mathsf{CU}}\cdot 2^n, (H_{\mathsf{CU}}+1)\cdot 2^n]$. Thus, we have that $H_{\mathsf{GC}} = (H_{\mathsf{CU}} + 1) \cdot 2^n$. We cut off x at Step 3-(a). So, we have that $L_{\mathsf{GC}} = 2^{-t_x}$.

On Δ_{GC}: We start with GC and add procedures to GC sequentially.

$\mathsf{GC}_0(a)$**:** This is the algorithm $\mathsf{GC}(a)$ with *no repeat*.

$\mathsf{GC}_1(a)$**:** We add the discarding procedure to Step 5. The algorithm discards u if $u < 2^{-t_{\mathsf{GC}}}$.

$\mathsf{GC}_2(a)$**:** In Step 4, we replace the criterion $x < 0$ with $x < 2^{-t_x}$.

$\mathsf{GC}_3(a)$**:** We modify the sampling method of u. We replace $u \leftarrow U(0,1)$ with $u \leftarrow \{0,1\}^{l_{\mathsf{GC}}}$ in Step 5.

$\mathsf{GC}_4(a)$**:** We replace the criterion $u \le f(x,a)/(C(a)g(x,a))$ in Step 6 with $\ln u \le \ln(f(x,a)) - \ln(C(a)) - \ln(g(x,a))$.

$\mathsf{GC}_5(a)$**:** Let s' denote the computed value of $A^{1/2}$ with precision ϵ_{SQRT}. Let $\delta \in [-\epsilon_{\mathrm{SQRT}}, \epsilon_{\mathrm{SQRT}}]$ be a real value such that $\delta = A^{1/2} - s'$. We replace the sampling $x \leftarrow \mathrm{C_A}(b,s)$ with $x \leftarrow (Ca)(b, s+\delta)$.

$\mathsf{GC}_6(a)$**:** We again modify the sampling method. The sample x is sampled as follows: $t \leftarrow \mathrm{C_A}(0,1)$ and $x \leftarrow s't + b$.

$\mathsf{GC}_6(a)$**:** We next modify the sampling method of t and computations in the criterion at Step 6. We replace $t \leftarrow \mathrm{C_A}(0,1)$ with $t \leftarrow \widetilde{\mathrm{C_A}}(0,1)$ with precision ϵ_{CU}.

$\mathsf{GC}_7(a)$**:** We replace $\widetilde{\mathrm{C_A}}(0,1)$ with the algorithm DisCU.

$\mathsf{GC}_8(a)$**:** We revive *go to* in Steps 4, 5, and 6. This is the algorithm $\mathsf{DiscGC}(a)$.

We first estimate $\Delta(\mathsf{GC}_0(a), \mathsf{GC}_1(a))$. Clearly, the change induces the distance $2^{-t_{\mathsf{GC}}}$.
Next, we estimate $\Delta(\mathsf{GC}_1(a), \mathsf{GC}_2(a))$. Notice that

$$\Pr_{X\sim \mathrm{C_A}(a-1,\sqrt{2a-1})}[0 \le X \le 2^{-t_x}] = \int_0^{2^{-t_x}} g(x,a)dx \le \int_0^{2^{-t_x}} dx \le 2^{-t_x}.$$

So, the distance is at most 2^{-t_x}.

We next upper bound $\Delta(\mathsf{GC}_2(a), \mathsf{GC}_3(a))$. We have changed $u \leftarrow U(0,1)$ at Step 5 with $u \leftarrow \{0,1\}^{l_{\mathsf{GC}}}$. The effect is at most $2^{-l_{\mathsf{GC}}}$ by Lemma K.3.

The distance $\Delta(\mathsf{GC}_3(a), \mathsf{GC}_4(a))$ is 0. This is because

$$\frac{f(x,a)}{C(a)g(x,a)} = \frac{e^{-x}x^b(A + (x-b)^2)}{e^{-b}b^bA},$$
$$\ln(f(x,a)) - \ln(C(a)) - \ln(g(x,a)) = b(\ln(x) - \ln(b)) - (x-b) + \ln(A + (x-b)^2) - \ln(A).$$

We next estimate $\Delta(\mathsf{GC}_4(a), \mathsf{GC}_5(a))$. We have replaced $x \leftarrow \mathrm{C_A}(b,s)$ with $x \leftarrow \mathrm{C_A}(b,s') = \mathrm{C_A}(b, s+\delta)$, since the square root is not computed precisely. The distance between $\mathrm{C_A}(b,s)$ and $\mathrm{C_A}(b,s+\delta)$ is at most 2δ from Lemma J.4 in Appendix J.1. Since δ is at most ϵ_{SQRT}, so is the distance. This shows that $\Delta(\mathsf{GC}_4(a), \mathsf{GC}_5(a)) \le 2\epsilon_{\mathrm{SQRT}}$.

Clearly the distance $\Delta(\mathsf{GC}_5(a), \mathsf{GC}_6(a))$ is 0, since $s't + b$ is distributed according to $\mathrm{C_A}(b,s')$.

We next estimate $\Delta(\mathsf{GC}_6(a), \mathsf{GC}_7(a))$. Replacing $t \leftarrow \mathrm{C}_{\mathrm{A}}(0,1)$ with $t \leftarrow \widetilde{\mathrm{C}_{\mathrm{A}}}(0,1)$ with precision ϵ_{CU} effects the value of x. This causes the precision of computation $\ln(f(x,a)) - \ln(C(a)) - \ln(g(x,a))$. To determine their effect to the distance, we compute the precision of them. $\ln x$ has precision $\epsilon_{\mathrm{LOG}} + 2^{t_x}\epsilon_{\mathsf{GC}}$, since $x > 2^{-t_x}$. Also, $\ln b$ has precision ϵ_{LOG}. $x - b = s't$ has precision $2^{n+1}\epsilon_{\mathsf{CU}}$. $\ln(A + (x-b)^2)$ has precision $\epsilon_{\mathrm{LOG}} + \epsilon_{\mathsf{GC}}$, since $A + (x-b)^2 > 1$. $\ln A$ has precision ϵ_{LOG}. Thus, $b(\ln x - \ln b) - (x-b) + \ln(A + (x-b)^2) - \ln A$ has precision $2^k(\epsilon_{\mathrm{LOG}} + 2^{t_x}\epsilon_{\mathsf{GC}}) + \epsilon_{\mathsf{GC}} + \epsilon_{\mathrm{LOG}} + \epsilon_{\mathrm{LOG}} + \epsilon_{\mathrm{LOG}} + 2^{n+1}\epsilon_{\mathsf{CU}}$, that is, $\epsilon_{\mathsf{GC}}(2^{k+t_x}+1) + \epsilon_{\mathrm{LOG}}(2^k+3) + 2^{n+1}\epsilon_{\mathsf{CU}}$. We also replace $\ln(u)$ with the approximation of the logarithm. This causes the error at most $2^{l_{\mathsf{GC}}}\epsilon_{\mathrm{LOG}}$.

Finally, we estimate $\Delta(\mathsf{GC}_6(a), \mathsf{GC}_7(a))$. The distance at most Δ_{CU}, since we only replace the sampling algorithms.

Compiling the above arguments, $\Delta(\mathsf{GC}(a), \mathsf{GC}_8(a))$ is upper bounded by

$$2^{-t_{\mathsf{GC}}} + 2^{-t_x+1} + 2^{-l_{\mathsf{GC}}} + \epsilon_{\mathrm{SQRT}} + \epsilon_{\mathsf{GC}}(2^{k+t_x}+1) + \epsilon_{\mathrm{LOG}}(2^k+3) + 2^{n+1}\epsilon_{\mathsf{CU}} + \epsilon_{\mathrm{LOG}}2^{t_{\mathsf{GC}}} + \Delta_{\mathsf{CU}}.$$

Thus, the statistical distance $\Delta_{\mathsf{GC}} = \Delta(\widetilde{\mathrm{G}_{\mathrm{A}}}(a), \mathsf{DiscGC}(a))$ is upper bounded by

$$\begin{aligned}&(7/10)^{r_{\mathsf{GC}}} + r_{\mathsf{CU}}(2^{-t_{\mathsf{GC}}} + 2^{-t_x+1} + 2^{-l_{\mathsf{GC}}} + \epsilon_{\mathrm{SQRT}} + \epsilon_{\mathsf{GC}}(2^{k+t_x}+1) + \epsilon_{\mathrm{LOG}}(2^k+3) + 2^{n+1}\epsilon_{\mathsf{CU}} + \epsilon_{\mathrm{LOG}}2^{t_{\mathsf{GC}}} + \Delta_{\mathsf{CU}})\\ &= 2^{-r_{\mathsf{GC}}/2} + r_{\mathsf{CU}}(2^{-t_{\mathsf{GC}}} + 2^{-t_x+1} + 2^{-l_{\mathsf{GC}}} + \epsilon_{\mathrm{SQRT}} + \epsilon_{\mathsf{GC}}(2^{k+t_x}+1) + \epsilon_{\mathrm{LOG}}(2^k + 2^{t_{\mathsf{GC}}} + 3) + 2^{n+1}\epsilon_{\mathsf{CU}} + \Delta_{\mathsf{CU}}).\end{aligned}$$

□

K.4 Sampling from the Beta Distribution based on the Gamma Distribution

According to the fact that if $X \sim \mathrm{G}_{\mathrm{A}}(a)$ and $Y \sim \mathrm{G}_{\mathrm{A}}(b)$ then $X/(X+Y) \sim \mathrm{B}_{\mathrm{E}}(a,b)$, we can sample from any beta distribution by using random variables sampled from two gamma distributions. Obviously, the algorithm BG in Figure 9 samples from $\widetilde{\mathrm{B}_{\mathrm{E}}}(a,b)$ with precision ϵ_{BG}.

Replacing the sampling algorithm and adding the criterion, we obtain the algorithm $\mathsf{DiscBG}(a,b)$ in Figure 10.

Algorithm $\mathsf{BG}(a,b)$

1. Take samples $x \leftarrow \mathrm{G}_{\mathrm{A}}(a)$ and $y \leftarrow \mathrm{G}_{\mathrm{A}}(b)$.
2. Output $R_{\epsilon_{\mathsf{BG}}}(x/(x+y))$.

Figure 9: Algorithm BG

Algorithm $\mathsf{DiscBG}(a,b)$

1. Take samples $x \leftarrow \mathsf{DiscGC}(a)$ and $y \leftarrow \mathsf{DiscGC}(b)$.
2. If $2^{-t_{\mathrm{low}}} < x, y < 2^{t_{\mathrm{high}}}$ then output $R_{\epsilon_{\mathsf{BG}}}(x/(x+y))$. Otherwise output $\perp$.

Figure 10: Algorithm DiscBG

Theorem K.8. *The sample from the algorithm* $\mathsf{DiscBG}(a,b)$ *has the following properties:*

- *The output has precision* ϵ_{BG},
- *the running time of the algorithm is at most* T_{BG},
- *the output is at most* H_{BG} *and at least* L_{BG}, *and*
- *the output distribution of* $\mathsf{DiscBG}(a,b)$ *is* Δ_{BG}*-close to* $\widetilde{\mathrm{B}_{\mathrm{E}}}(a,b)$ *with precision* ϵ_{BG},

where the parameters are as follows:

$$\begin{aligned}\epsilon_{\mathsf{BG}} &= 2^{t_{\mathrm{high}}+2t_{\mathrm{low}}+2}\epsilon_{\mathsf{GC}},\\ T_{\mathsf{BG}} &\le O(T_{\mathsf{GC}} + T_{\mathcal{A}}(z) + T_{\mathcal{D}}(z))\\ H_{\mathsf{BG}} &= 1,\\ L_{\mathsf{BG}} &= 2^{-(t_{\mathrm{high}}+t_{\mathrm{low}}+1)},\\ \Delta_{\mathsf{BG}} &\le 2\Delta_{\mathsf{GC}} + 2^{-a(t_{\mathrm{low}}+1)} + 2^{-b(t_{\mathrm{low}}+1)} + 2^{-2^{t_{\mathrm{high}}}+(a-1)t_{\mathrm{high}}} + 2^{-2^{t_{\mathrm{high}}}+(b-1)t_{\mathrm{high}}},\end{aligned}$$

where $z = O(\log(H_{\mathsf{GC}}) + \log(1/\epsilon_{\mathsf{GC}}))$.

Proof.

On ϵ_{BG}: It is easy to verify that the precision of $x/(x+y)$ is bounded by $2^{t_{\text{high}}+2t_{\text{low}}+1}\epsilon_{\mathsf{GC}}$.

On H_{BG} and L_{BG}: We have

$$2^{-(t_{\text{high}}+t_{\text{low}}+1)} < \frac{1}{1+2^{t_{\text{high}}+t_{\text{low}}}} < \frac{1}{1+y/x} = \frac{x}{x+y} < \frac{1}{1+2^{-(t_{\text{high}}+t_{\text{low}})}} < 1 - 2^{-(t_{\text{high}}+t_{\text{low}}+1)}.$$

On Δ_{BG}: We denote by $\mathsf{DiscBG}^*(a,b)$ the algorithm $\mathsf{DiscBG}(a,b)$ using $\mathrm{G}_{\mathrm{A}}(\cdot)$ instead of $\mathsf{DiscGC}(\cdot)$.

$$\begin{aligned}
\Delta\left(\widetilde{\mathrm{BE}}(a,b), \mathsf{DiscBG}(a,b)\right) &\le \Delta\left(\widetilde{\mathrm{BE}}(a,b), \mathsf{BG}(a,b)\right) + \Delta\left(\mathsf{BG}(a,b), \mathsf{DiscBG}^*(a,b)\right) \\
&\quad + \Delta\left(\mathsf{DiscBG}^*(a,b), \mathsf{DiscBG}(a,b)\right) \\
&\le 0 + \Pr_{x\sim \mathrm{G}_{\mathrm{A}}(a)}\left[x < 2^{-t_{\text{low}}} \text{ or } x > 2^{t_{\text{high}}}\right] \\
&\quad + \Pr_{x\sim \mathrm{G}_{\mathrm{A}}(b)}\left[x < 2^{-t_{\text{low}}} \text{ or } x > 2^{t_{\text{high}}}\right] + 2\Delta_{\mathsf{GC}} \\
&\le 2\Delta_{\mathsf{GC}} + 2^{-a(t_{\text{low}}+1)} + 2^{-b(t_{\text{low}}+1)} + 2^{-2^{t_{\text{high}}}+(a-1)t_{\text{high}}} + 2^{-2^{t_{\text{high}}}+(b-1)t_{\text{high}}},
\end{aligned}$$

where in the last inequality we use Lemma J.5 and Lemma J.6 in Appendix J.2. □

K.5 Sampling from the Binomial Distribution based on the Bernoulli Experiments

Based on the fact that the binomial distribution describes the number of successes in N independent Bernoulli experiments, we have the following algorithms $\mathsf{BU}(N,p)$ and $\mathsf{DiscBU}(N,\tilde{p})$ in Figures 11 and 12, respectively, where $\tilde{p}$ denotes an approximation of p with precision $\epsilon_{p,\mathsf{BU}}$.

Algorithm $\mathsf{BU}(N,p)$

1. Initialize $i \leftarrow 0$ and $c \leftarrow 0$.
2. Generate $u \leftarrow \mathrm{U}_{\mathrm{N}}(0,1)$. If $u \le p$ score a success: $c \leftarrow c+1$.
3. Increase $i \leftarrow i+1$. If $i < n$ go to Step 2, otherwise output c.

Figure 11: Algorithm BU

Algorithm $\mathsf{DiscBU}(N,\tilde{p})$

1. Initialize $i \leftarrow 0$ and $c \leftarrow 0$.
2. Generate $u \leftarrow \{0,1\}^{l_{\mathsf{BU}}}$. (Consider u as a real value in $[0,1)$.) If $u \le \tilde{p}$ score a success: $c \leftarrow c+1$.
3. Increase $i \leftarrow i+1$. If $i < N$ go to Step 2, otherwise output c.

Figure 12: Algorithm DiscBU

Theorem K.9. *For the inputs N and $\tilde{p}$, the sampling algorithm $\mathsf{DiscBU}(N,\tilde{p})$ has following properties: If the inputs satisfy*

- *N: a positive integer between $0 \le N \le n$,*
- *$\tilde{p}$: a positive real between $0 \le \tilde{p} \le 1$ which is an approximation of $0 \le p \le 1$ with precision $\epsilon_{p,\mathsf{BU}}$,*

then

- *the output distribution of $\mathsf{DiscBU}(N,\tilde{p})$ is Δ_{BU}-close to $\mathrm{B}_{\mathrm{N}}(N,p)$ and*

- *the running time is at most* T_{BU},

where the parameters are as follows:

$$\Delta_{\mathsf{BU}} \leq N(\epsilon_{p,\mathsf{BU}} + 2^{-l_{\mathsf{BU}}}),$$
$$T_{\mathsf{BU}} \leq N \cdot O(T_{\mathcal{A}}(\log N) + l_{\mathsf{BU}} T_{\mathcal{U}}).$$

Proof. By using the triangle inequality, we have that

$$\begin{aligned}
\Delta_{\mathsf{BU}} &= \Delta\left(\mathrm{B}_{\mathrm{N}}(N,p), \mathsf{DiscBU}(N,\tilde{p})\right) \\
&\leq \Delta\left(\mathrm{B}_{\mathrm{N}}(N,p), \mathsf{BU}(N,p)\right) + \Delta\left(\mathsf{BU}(N,p), \mathsf{BU}(N,\tilde{p})\right) + \Delta\left(\mathsf{BU}(N,\tilde{p}), \mathsf{DiscBU}(N,\tilde{p})\right) \\
&\leq 0 + N \cdot \epsilon_{p,\mathsf{BU}} + N \cdot 2^{-l_{\mathsf{BU}}} \\
&\leq N(\epsilon_{p,\mathsf{BU}} + 2^{-l_{\mathsf{BU}}}).
\end{aligned}$$

□

K.6 Sampling from the Binomial Distribution based on the Beta Distribution

The algorithm BU is simple, but has a drawback that the running time grows exponentially in input size of N. In order to overcome this drawback, the following lemma in [Rel72] is useful.

Lemma K.10 ([Rel72])**.** *The following procedure samples correctly from a binomial distribution* $\mathrm{B}_{\mathrm{N}}(n,p)$.

1. Select any two positive integers a and b such that $a + b - 1 = n$.
2. Take a sample s from the beta distribution $\mathrm{B}_{\mathrm{E}}(a,b)$.
3. If $s \leq p$ take a sample x from the binomial distribution $\mathrm{B}_{\mathrm{N}}(b-1,(p-s)/(1-s))$ and output $x \leftarrow a + x$.
4. If $s > p$ take a sample x from the binomial distribution $\mathrm{B}_{\mathrm{N}}(a-1,p/s)$ and output x.

The slightly modified algorithm BB in Figure 13 appeared in [Rel72]. As the consequence of the lemma, the algorithm correctly samples according to $\mathrm{B}_{\mathrm{N}}(n,p)$. We note the running time of this algorithm grows linearly in $\log n$.

We next modify the algorithm BB by replacing subroutines. The obtained algorithm DiscBB is in Figure 14, where $\tilde{p}$ is an approximation of p with precision $\epsilon_{p,\mathsf{BB}}$ and δ is the cut-off parameter such that $N\delta$ and $N^4\epsilon_{p,\mathsf{BB}}\delta^{-n}$ is negligible in n. Note that we add Steps 1-(a) and 1-(b) to DiscBB by the technical reason.

Theorem K.11. *For the inputs* N *and* $\tilde{p}$*, the sampling algorithm* $\mathsf{DiscBB}(N,\tilde{p})$ *has following properties: If the inputs satisfy*

- N*: a positive integer between* $0 \leq N \leq 2^n$.
- $\tilde{p}$*: a positive real between* $\delta < \tilde{p} < 1 - \delta$ *which is an approximation* p *with precision* $\epsilon_{p,\mathsf{BB}}$, [5]

then

- *the output distribution is* Δ_{BB}*-close to* $\mathrm{B}_{\mathrm{N}}(N,p)$
- *and the running time is at most* T_{BB},

[5] By modifying the algorithm using Chebyshev's inequality or Lemma J.8 we can treat in the case $0 \leq p \leq 1$. This is done by Steps 1-(a) and 1-(b).

Algorithm $\mathsf{BB}(N, p)$

1. If $N \le 2$ generate $x \leftarrow \mathrm{B}_{\mathrm{N}}(n, p)$ and output x.
2. If N is even then generate $y \leftarrow \mathrm{B}_{\mathrm{N}}(1, p)$ and $x \leftarrow \mathrm{B}_{\mathrm{N}}(N-1, p)$, and output $x + y$.
3. If N is odd then set $a \leftarrow (N+1)/2$ and generate $s \leftarrow \mathrm{B}_{\mathrm{E}}(a, a)$.
4. If $s \le p$ then $x \leftarrow \mathrm{B}_{\mathrm{N}}(a-1, (p-s)/(1-s))$ and output $x + a$. Otherwise $x \leftarrow \mathrm{B}_{\mathrm{N}}(a-1, p/s)$ and output x.

Figure 13: Algorithm BB

Algorithm $\mathsf{DiscBB}(N, \tilde{p})$

1. If $N \le 2$ generate $x \leftarrow \mathsf{DiscBU}(N, \tilde{p})$ and output x.
   (a) If $\tilde{p} \le \delta$ then output 0.
   (b) If $\tilde{p} \ge 1 - \delta$ then output N.
2. If N is even then generate $y \leftarrow \mathsf{DiscBU}(1, \tilde{p})$ and $x \leftarrow \mathsf{DiscBB}(N-1, \tilde{p})$, and output $x + y$.
3. If N is odd then set $a \leftarrow (N+1)/2$ and generate $s \leftarrow \mathsf{DiscBG}(a, a)$.
4. If $s \le \tilde{p}$ then $x \leftarrow \mathsf{DiscBB}(a-1, (\tilde{p}-s)/(1-s))$ and output $x + a$. Otherwise $x \leftarrow \mathsf{DiscBB}(a-1, \tilde{p}/s)$ and output x.

Figure 14: Algorithm DiscBB

where the parameters are as follows:

$$\Delta_{\mathsf{BB}} \le \delta^{-n} 2^{4n+1} \epsilon_{p,\mathsf{BB}} + n 2^{4n} \epsilon_{\mathsf{BG}} + 2n \cdot 2^{-l_{\mathsf{BU}}} + n\Delta_{\mathsf{BG}},$$
$$T_{\mathsf{BB}} \le n \cdot O(T_{\mathcal{A}}(z) + T_{\mathcal{D}}(z) + T_{\mathsf{BG}} + T_{\mathsf{BU}})),$$

where $z = O(n + \log(1/\epsilon_{p,\mathsf{BB}}))$.

Proof of Theorem K.11. On Δ_{BB}: Before giving the proof, we introduce the notation for clarity.

We first define $d(\epsilon, N)$ in order to treat the effect of recursive sampling:

$$d(\epsilon, N) = \max_{p} \max_{\tilde{p}:|p-\tilde{p}|<\epsilon} \Delta(\mathrm{B}_{\mathrm{N}}(N, p), \mathsf{DiscBB}(N, \tilde{p})).$$

In order to estimate $d(\epsilon, N)$ above, we define the algorithms BB_i. We upper bound each $\Delta(\mathsf{BB}_i, \mathsf{BB}_{i+1})$ and then upper bound $\Delta(\mathrm{B}_{\mathrm{N}}(N, p), \mathsf{DiscBB}(N, \tilde{p}))$ by the triangle inequality. We define the algorithms BB_i as follows:

$\mathsf{BB}_0(N, p)$**:** This is the algorithm $\mathsf{BB}(N, p)$ which correctly samples from $\mathrm{B}_{\mathrm{N}}(N, p)$.

$\mathsf{BB}_1(N, \tilde{p})$**:** We replace the input p with $\tilde{p}$ in BB_0.

$\mathsf{BB}_2(N, \tilde{p})$**:** We use $\widetilde{\mathrm{B}_{\mathrm{E}}}$ with precision ϵ_{BG} instead of B_{E} in BB_1.

$\mathsf{BB}_3(N, \tilde{p})$**:** We replace $\widetilde{\mathrm{B}_{\mathrm{E}}}$ with DiscBG in the algorithm BB_2.

$\mathsf{BB}_4(N, \tilde{p})$**:** We use DiscBU instead of B_{N} at Steps 1 and 2 in the algorithm BB_3.

$\mathsf{BB}_5(N, \tilde{p})$**:** We finally replace B_{N} with DiscBB at Step 4 in the algorithm BB_4. The obtained algorithm coincides with DiscBB.

From the above definitions,

$$\begin{aligned}\Delta(\mathrm{B}_{\mathrm{N}}(N, p), \mathsf{DiscBB}(N, \tilde{p})) &\le \Delta(\mathrm{B}_{\mathrm{N}}(N, p), \mathrm{B}_{\mathrm{N}}(N, \tilde{p})) + \Delta(\mathrm{B}_{\mathrm{N}}(N, \tilde{p}), \mathsf{DiscBB}(N, \tilde{p})) \\ &\le \Delta(\mathrm{B}_{\mathrm{N}}(N, p), \mathrm{B}_{\mathrm{N}}(N, \tilde{p})) + \sum_{i=1}^{4} \Delta(\mathsf{BB}_i(N, \tilde{p}), \mathsf{BB}_{i+1}(N, \tilde{p})).\end{aligned}$$

By using Lemma J.9, we have that

$$\Delta(\mathrm{B}_N(N,p),\mathrm{B}_N(N,\tilde{p})) \le N^4\epsilon.$$

We next consider $\Delta(\mathsf{BB}_1(N,\tilde{p}),\mathsf{BB}_2(N,\tilde{p}))$. The change of s at Step 4 effects the distance at most $2a(a-1)^3\epsilon_{\mathsf{BG}} \le N^4\epsilon_{\mathsf{BG}}$ by Lemma J.7 in Appendix J.3.

It is obvious that $\Delta(\mathsf{BB}_2(N,\tilde{p}),\mathsf{BB}_3(N,\tilde{p})) \le \Delta_{\mathsf{BG}}$. Is is also obvious that $\Delta(\mathsf{BB}_3(N,\tilde{p}),\mathsf{BB4}(N,\tilde{p})) \le 2(\epsilon + 2^{-l_{\mathsf{BU}}})$. We next consider $\Delta(\mathsf{BB}_4(N,\tilde{p}),\mathsf{BB}_5(N,\tilde{p}))$. Since we change the choosing method for x at Steps 3 and 4, we have that

$$\Delta(\mathsf{BB}_3(N,\tilde{p}),\mathsf{BB}_4(N,\tilde{p})) \le \max\left\{\Delta\left(\mathrm{B}_N\left(a-1,p_1\right),\mathsf{DiscBB}\left(a-1,\tilde{p}_1\right)\right),\Delta\left(\mathrm{B}_N\left(a-1,p_2\right)\mathsf{DiscBB}\left(a-1,\tilde{p}_2\right)\right)\right\},$$

where $p_1 = (\tilde{p}-s)/(1-s)$, $p_2 = \tilde{p}/s$, and each $\tilde{p}_i$ is an approximation of p_i, respectively. By simple calculation, we have that

$$\begin{aligned}
|p_1-\tilde{p}_1| &= \left|\frac{p-s}{1-s}-\frac{\tilde{p}-s}{1-s}\right| = \left|\frac{p-\tilde{p}}{1-s}\right| \le \frac{\epsilon}{1-s} \le \frac{\epsilon}{1-\tilde{p}} \le \frac{\epsilon}{\delta},\\
|p_2-\tilde{p}_2| &= \left|\frac{p}{s}-\frac{\tilde{p}}{s}\right| = \left|\frac{p-\tilde{p}}{s}\right| \le \frac{\epsilon}{s} \le \frac{\epsilon}{\tilde{p}} \le \frac{\epsilon}{\delta}.
\end{aligned}$$

Thus, we have that

$$\Delta(\mathsf{BB}_3(N,\tilde{p}),\mathsf{BB}_4(N,\tilde{p})) \le d(\epsilon/\delta, a-1) \le d(\epsilon/\delta, N/2).$$

Summarizing the above calculation, we have that

$$\begin{aligned}
\Delta(\mathrm{B}_N(N,p),\mathsf{DiscBB}(N,\tilde{p})) &\le \Delta(\mathrm{B}_N(N,p),\mathrm{B}_N(N,\tilde{p})) + \sum_{i=1}^{3}\Delta(\mathsf{BB}_i(N,\tilde{p}),\mathsf{BB}_{i+1}(N,\tilde{p}))\\
&\le N^4\cdot\epsilon + N^4\cdot\epsilon_{\mathsf{BG}} + \Delta_{\mathsf{BG}} + 2(\epsilon + 2^{-l_{\mathsf{BU}}}) + d(\epsilon/\delta, N/2).
\end{aligned}$$

Thus, we have that

$$\begin{aligned}
d(N,\epsilon_{p,\mathsf{BB}}) &\le N^4\cdot\epsilon_{p,\mathsf{BB}} + N^4\cdot\epsilon_{\mathsf{BG}} + 2(\epsilon_{p,\mathsf{BB}} + 2^{-l_{\mathsf{BU}}}) + \Delta_{\mathsf{BG}} + d(\epsilon_{p,\mathsf{BB}}/\delta, N/2)\\
&\le N^4(\epsilon_{p,\mathsf{BB}} + \epsilon_{p,\mathsf{BB}}/\delta) + 2N^4\cdot\epsilon_{\mathsf{BG}}\\
&\qquad + 2(\epsilon_{p,\mathsf{BB}} + \epsilon_{p,\mathsf{BB}}/\delta + 2\cdot 2^{-l_{\mathsf{BU}}}) + \Delta_{\mathsf{BG}} + d(\epsilon_{p,\mathsf{BB}}/\delta^2, N/4)\\
&\le N^4\epsilon_{p,\mathsf{BB}}(1 + 1/\delta + \cdots + 1/\delta^n) + nN^4\cdot\epsilon_{\mathsf{BG}}\\
&\qquad + 2(\epsilon_{p,\mathsf{BB}}(1 + 1/\delta + \cdots + 1/\delta^n) + n\cdot 2^{-l_{\mathsf{BU}}}) + n\Delta_{\mathsf{BG}} + d(\epsilon_{p,\mathsf{BB}}/\delta^n, 1)\\
&\le N^4\epsilon_{p,\mathsf{BB}}\delta^{-n} + nN^4\epsilon_{\mathsf{BG}} + 2\epsilon_{p,\mathsf{BB}}\delta^{-n} + 2n\cdot 2^{-l_{\mathsf{BU}}} + n\Delta_{\mathsf{BG}}) + \delta^{-n}\epsilon_{p,\mathsf{BB}}\\
&= \delta^{-n}(N^4+2)\epsilon_{p,\mathsf{BB}} + nN^4\epsilon_{\mathsf{BG}} + 2n\cdot 2^{-l_{\mathsf{BU}}} + n\Delta_{\mathsf{BG}}.
\end{aligned}$$

where in the last inequality we use $\sum_{i=0}^{n} 1/\delta^i = (1/\delta^{n+1}-1)/(1/\delta-1) = \delta^{-n}(1-\delta^{n+1})(1-\delta) \le \delta^{-n}$. □

References

[AD74] Joachim H. Ahrens and Ulrich Dieter. Computer methods for sampling from Gamma, Beta, Poisson and Binomial distributions. *Computing*, 12(3):223–246, 1974.

[AD80] Joachim H. Ahrens and Ulrich Dieter. Sampling from Binomial and Poisson distributions: A method with bounded computation times. *Computing*, 25(3):193–208, 1980.

[AS72] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of matehmatical functions*. Number 55 in National Bureau of Standards Applied Mathematics Series. United States Government Printing, tenth edition, 1972.

[AS09] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Michael J. Jacobson, Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *SAC 2009*, volume 5867 of *LNCS*, pages 103–119. Springer, Heidelberg, 2009.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 171–188. Springer, Heidelberg, 2004.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracle are practical: A paradigm for designing efficient protocols. In *CCS '93*, pages 62–73. ACM, 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT '94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, 1995.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, 1996.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004. Preliminary version in *STOC '98*, 1998.

[CHJ+02] Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. Gem: A Generic chosen-ciphertext secure Encryption Method. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 175–184. Springer, Heidelberg, 2002.

[Dev86] Luc Devroye Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[FL07] Marc Fischlin and Anja Lehmann. Security-amplifying combiners for collision-resistant hash functions. In Menezes [Men07], pages 224–243.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, 1999.

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, 2004.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 102–113, Cambridge, MA, USA, October 2003. IEEE Computer Society.

[GMMV05] David Galindo, Sebastià Martín, Paz Morillo, and Jorge L. Villar. Fujisaki-Okamoto hybrid encryption revisited. *International Journal of Information Security*, 4(4):228–241, 2005.

[HS08] Jonathan J. Hoch and Adi Shamir. On the strength of the concatenated hash combiner when all the hash functions are weak. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 616–630. Springer, Heidelberg, 2008.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KP09] Eike Kiltz and Krzysztof Pietrzak. On the security of padding-based encryption schemes (or: Why we cannot prove OAEP secure in the standard model). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 389–406. Springer, Heidelberg, 2009.

[Lis07] Moses Liskov. Constructing an ideal hash function from weak ideal compression functions. In Eli Biham and Amr M. Youssef, editors, *SAC 2006*, volume 4356 of *LNCS*, pages 358–375. Springer, Heidelberg, 2007.

[LN09] Gaëtan Leurent and Phong Q. Nguyen. How risky is the random-oracle model? In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 445–464. Springer, Heidelberg, 2009. The full version is available at `http://eprint.iacr.org/2008/441`.

[Men07] Alfred Menezes, editor. *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *LNCS*. Springer, Heidelberg, 2007.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, 2004.

[Nat02] National Institute of Standards and Technology. Secure hash standard. FIPS 180-2, August 2002.

[Nie02] Jesper Buus Nielsen Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, 2002.

[NIT08] Akira Numayama, Toshiyuki Isshiki, and Keisuke Tanaka. Security of digital signature schemes in weakened random oracle models. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 268–287. Springer, Heidelberg, 2008.

[NWO09] Yusuke Naito, Lei Wang, and Kazuo Ohta. How to construct cryptosystems and hash functions in weakenend random oracle models. Cryptology ePrint Archive, Report 2009/550, 2009.

[OP01] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175. Springer, Heidelberg, 2001.

[PV07] Sylvain Pasini and Serge Vaudenay. Hash-and-sign with weak hashing made secure. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 2007*, volume 4586 of *LNCS*, pages 338–354. Springer, Heidelberg, 2007.

[Rel72] Daniel A. Relles. A simple algorithm for generating Binomial random variables when N is large. *American Statistical Association*, 67(339):612–613, 1972.

[Riv92] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Menezes [Men07], pages 205–223.

[vN51] John von Neumann. Various techniques used in connection with random digits. Monte Carlo methods. *Nat. Bureau Standards*, 12:36–38, 1951.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, Heidelberg, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, Heidelberg, 2005.