Cryptography and Algorithmic Randomness[*]

Kohtaro Tadaki   Norihisa Doi

Research and Development Initiative, Chuo University
1–13–27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
E-mail: tadaki@kc.chuo-u.ac.jp,   doi@doi.ics.keio.ac.jp
http://www2.odn.ne.jp/tadaki/

Abstract. The secure instantiation of the random oracle is one of the major open problems in modern cryptography. We investigate this problem using concepts and methods of algorithmic randomness.

In modern cryptography, the random oracle model is widely used as an imaginary framework in which the security of a cryptographic scheme is discussed. In the random oracle model, the cryptographic hash function used in a cryptographic scheme is formulated as a random variable uniformly distributed over all possibility of the function, called the random oracle. The main result of this paper is to show that, for any secure signature scheme in the random oracle model, there exists a specific computable function which can instantiate the random oracle while keeping the security originally proved in the random oracle model. In modern cryptography the generic group model is used also for a similar purpose to the random oracle model. We show that the same results hold for the generic group model.

In the process of proving the results, we introduce the notion of effective security, demonstrating the importance of this notion in modern cryptography.

Key words: cryptography, random oracle model, generic group model, provable security, algorithmic randomness, computable analysis

1 Introduction

In modern cryptography, the *random oracle model* [1] is widely used as an *imaginary* framework in which the security of a cryptographic scheme is discussed. In the random oracle model, the cryptographic hash function used in a cryptographic scheme is formulated as a random variable uniformly distributed over all possibility of the function, called the *random oracle*, and the legitimate users and the adversary against the scheme are modeled so as to get the values of the hash function not by evaluating it in their own but by querying the random oracle. Since the random oracle is an

---

[*]A part of this work was presented at the Seventh International Conference on Computability, Complexity and Randomness (CCR 2012), July 2-6, 2012, Cambridge, Great Britain.

imaginary object, even if the security of a cryptographic scheme is proved in the random oracle model, the random oracle has to be instantiated using a concrete cryptographic hash function such as the SHA hash functions if we want to use the scheme in the real world. In fact, the instantiations of the random oracle by concrete cryptographic hash functions are widely used in modern cryptography to produce efficient cryptographic schemes. Once the random oracle is instantiated, however, the original security proof in the random oracle model is spoiled and goes back to square one. Actually, it is not clear how much the instantiation can maintain the security originally proved in the random oracle model, nor is it clear whether the random oracle can be instantiated somehow while keeping the original security.

The question of securely instantiating the random oracle within cryptographic schemes proven secure in the random oracle model is one of the most intriguing problems in modern cryptography. Actually, many researches on the secure instantiation of the random oracle have been done so far, which include Canetti, et al. [5], Bellare, et al. [2], Leurent and Nguyen [17], Fischlin, et al. [11]. These mainly give negative results.

In this paper we investigate the problem of secure instantiation of the random oracle, using concepts and methods of *algorithmic randomness*. Algorithmic randomness, also known as *algorithmic information theory*, enables us to consider the randomness of an *individual* object. It originated in the groundbreaking works of Solomonoff [28], Kolmogorov [16], and Chaitin [6] in the mid-1960s. They independently introduced the notion of *program-size complexity*, also known as *Kolmogorov complexity*, in order to quantify the randomness of an individual object. Around the same time, Martin-Löf [18] introduced a measure theoretic approach to characterize the randomness of an individual infinite binary sequence. This approach, called *Martin-Löf randomness* nowadays, is one of the major notions in algorithmic randomness as well as program-size complexity. Later on, in the 1970s Schnorr [26] and Chaitin [7] showed that Martin-Löf randomness is equivalent to the randomness defined by program-size complexity in characterizing random infinite binary sequences. In the 21st century, algorithmic randomness makes remarkable progress through close interaction with recursion theory [23, 10].

In cryptography, the randomness is just a probability distribution or its sequence. Namely, the *true randomness* in cryptography is a uniform probability distribution such as the random oracle, while the *pseudorandomness* is a sequence of probability distributions which has a certain asymptotic property defined based on computational complexity theory. Thus, cryptology seems to have had no concern with the randomness of an individual object so far. In algorithmic randomness, on the other hand, the notion of a *random real* plays a central role. It is an individual infinite binary sequence which is classified as "random", and not a random variable, unlike in cryptography. Algorithmic randomness enables us to classify an individual infinite binary sequence into random or not.

To summarize our contributions, we first review the security proof in the random oracle model (see e.g. Katz and Lindell [15, Chapter 13] for the detail). In the random oracle model, a cryptographic scheme Π relies on an oracle h which is a certain type of function mapping finite strings to finite strings, depending on a security parameter n. Let Hash_n denote the set of all such functions h on a security parameter n. Then the random oracle is the sequence $\{H_n\}$ of random variables such that each H_n is uniformly distributed over functions in Hash_n. Now, in order to introduce a security notion, such as CCA-security for encryption schemes and EUF-ACMA security for signature schemes, into the scheme Π, we first consider an appropriately designed experiment $\mathsf{Expt}_{\mathcal{A}^{H_n}, \Pi^{H_n}}$ defined for the scheme Π and any adversary $\mathcal{A}$, where Π and $\mathcal{A}$ are both allowed to have an

oracle access to H_n. Then a definition of security for Π in the random oracle model takes the following general form: the scheme Π is *secure in the random oracle model* if, for all probabilistic polynomial-time adversaries $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,

$$\Pr\left[\mathsf{Expt}_{\mathcal{A}^{H_n},\Pi^{H_n}}(n) = 1\right] \leq \gamma + \frac{1}{n^d}, \tag{1}$$

where the probability is taken over the random variable H_n, i.e., the random choice of a function in Hash_n, as well as the random choices of the parties running Π and those of the adversary $\mathcal{A}$. The value γ indicates the maximum desired probability of some "bad" event (e.g., for encryption schemes $\gamma = 1/2$ and for signature schemes $\gamma = 0$). Since the random variable H_n is uniformly distributed over Hash_n for every n, the definition (1) of security in the random oracle model is equivalently rewritten into the following form: for all probabilistic polynomial-time adversaries $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,

$$\frac{1}{\#\mathsf{Hash}_n} \sum_{h \in \mathsf{Hash}_n} \Pr\left[\mathsf{Expt}_{\mathcal{A}^{H_n},\Pi^{H_n}}(n) = 1 \mid H_n = h\right] \leq \gamma + \frac{1}{n^d}, \tag{2}$$

where $\#\mathsf{Hash}_n$ denotes the number of functions in Hash_n, and the probability is now conditioned on that the random variable H_n takes a specific function $h \in \mathsf{Hash}_n$ as its value.

Let $\{h_n\}$ be an arbitrary sequence of functions such that $h_n \in \mathsf{Hash}_n$ for all n. In this paper, we introduce the notion of *security of Π relative to a specific oracle $\{h_n\}$*, which can be formulated as follows: the scheme Π is secure relative to $\{h_n\}$ if, for all probabilistic polynomial-time adversaries $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,

$$\Pr\left[\mathsf{Expt}_{\mathcal{A}^{H_n},\Pi^{H_n}}(n) = 1 \mid H_n = h_n\right] \leq \gamma + \frac{1}{n^d}. \tag{3}$$

The specific sequence $\{h_n\}$ of functions is an *instantiation* of the random oracle $\{H_n\}$. Note that, in the case where $\{h_n\}$ is polynomial-time computable, i.e., there exists a deterministic Turing machine which on every input $(1^n, x)$ operates and outputs $h_n(x)$ within time polynomial in n, the condition (3) implies that the scheme Π is just *secure in the standard model*. Here, the standard model is the normal model of a cryptographic scheme, where no random oracle is present.

In this paper, we investigate the properties of a specific oracle $\{h_n\}$ relative to which Π is secure, under the assumption that Π is secure in the random oracle model. The contributions of the paper to the random oracle methodology are as follows:

(i) We investigate the instantiation of the random oracle by a random real in a signature scheme already proved secure in the random oracle model. We present equivalent conditions for a specific oracle $\{h_n\}$ instantiating the random oracle to keep a signature scheme secure, using a concept of algorithmic randomness, i.e., a variant of Martin-Löf randomness. Based on this, in particular we show that the security proved in the random oracle model is firmly maintained after instantiating the random oracle by a random real.

(ii) We introduce the notion of *effective security*, which is a constructive strengthen of the conventional (non-constructive) notions of security. In terms of the definitions (1) and (3) of security, the "effectiveness" means that the natural number N can be computed from the code of an adversary $\mathcal{A}$ and a natural number d. We consider signature schemes in the random oracle

model, and show that some specific *computable* function $\{h_n\}$ can instantiate the random oracle while keeping the effective security originally proved in the random oracle model. We demonstrate that the effective security notions are a natural alternative to the conventional security notions in modern cryptography by reconsidering the security notions required in modern cryptography.

The results in the contributions (i) and (ii) above are based only on the general form of the definitions of security notions for a signature scheme in modern cryptography, and depend neither on specific schemes nor on specific security notions. On the other hand, our results on the secure instantiation of the random oracle are valid only if the security in the random oracle model is confirmed already. This may imply that the random oracle model is not necessarily an imaginary framework to discuss the security of a cryptographic scheme, but may have substantial implications for the security in the standard model.

In addition to the random oracle model, in modern cryptography the *generic group model* [27] is used also as an imaginary framework in which the security of a cryptographic scheme is discussed. In particular, the generic group model is often used to discuss the *computational hardness* of problems, such as the discrete logarithm problem and the Diffie-Hellman problem in finite cyclic groups, which is used as a computational hardness assumption to prove the security of a cryptographic scheme. In the generic group model, the *generic group*, i.e., a random encoding of the group elements, is an imaginary object, just like the random oracle. Therefore, even if the security of a cryptographic scheme or the hardness of a computational problem is proved in the generic group model, the generic group has to be instantiated using a concrete finite cyclic group whose group operations are efficiently computable, for use of the cryptographic scheme in the real world. Hence, the problem of the secure instantiation of the generic group exists in the generic group model, just like in the random oracle model.

In this paper we introduce the notion of *effective hardness* for computational problems, which corresponds to the effective security for cryptographic schemes. Based on concepts and methods of algorithmic randomness, we then show that, for the discrete logarithm problem and the Diffie-Hellman problem in the generic group model, the generic group can be instantiated by a specific computable function while keeping the effective hardness originally proved in the generic group model. This result corresponds to the contribution (ii) above for the random oracle model. We can show the results for the generic group model which corresponds to the contribution (i) for the random oracle model. However, this task is not difficult and therefore omitted in this paper.

1.1 Organization of the paper

The paper is organized as follows. As preliminaries we first review some definitions and results of algorithmic randomness in Section 2.

We then begin the study of the random oracle model in Section 3, where we present the general form of signature schemes which we consider in this paper, and introduce the (conventional) security notion for the signature schemes. In Section 4 we present the contribution (i) above for the random oracle model. Subsequently we present the contribution (ii) above in Sections 5 and 6, where we introduce the notion of effective security and then show a secure instantiation of the random oracle by a computable function in Sections 5, and we demonstrate the importance of the effective security notions in Section 6.

We then begin the study of the generic group model in Section 7, where we explain the discrete logarithm problem in the generic group model. In Section 8 we develop the Lebesgue outer measure on families of encoding functions. It is needed in Section 9, where we introduce the notion of effective hardness and then show a secure instantiation of the generic group in the discrete logarithm problem by a computable function. In Section 10 we show that the same results hold for the Diffie-Hellman problem. We conclude this paper with the clarification of the notion of effective hardness in Section 11.

2   Preliminaries

We start with some notation about numbers and strings which will be used in this paper. $\#S$ is the cardinality of S for any set S. $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ is the set of natural numbers, and $\mathbb{N}^+$ is the set of positive integers. $\mathbb{Q}$ is the set of rationals, and $\mathbb{R}$ is the set of reals.

$\{0, 1\}^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$ is the set of finite binary strings where λ denotes the *empty string*, and $\{0, 1\}^*$ is ordered as indicated. We identify any string in $\{0, 1\}^*$ with a natural number in this order, i.e., we consider a map $\varphi \colon \{0, 1\}^* \to \mathbb{N}$ such that $\varphi(x) = 1x - 1$ where the concatenation $1x$ of the strings 1 and x is regarded as a dyadic integer, and then we identify x with $\varphi(x)$. For any $x \in \{0, 1\}^*$, $|x|$ is the *length* of x. For any $n \in \mathbb{N}$, we denote by $\{0, 1\}^n$ and $\{0, 1\}^{\leq n}$ the sets $\{x \mid x \in \{0, 1\}^* \ \& \ |x| = n\}$ and $\{x \mid x \in \{0, 1\}^* \ \& \ |x| \leq n\}$, respectively. For any $n, m \in \mathbb{N}$, we denote by Func_n^m and $\mathsf{Func}_{\leq n}^m$ the set of all functions mapping $\{0, 1\}^n$ to $\{0, 1\}^m$ and the set of all functions mapping $\{0, 1\}^{\leq n}$ to $\{0, 1\}^m$, respectively. A subset S of $\{0, 1\}^*$ is called *prefix-free* if no string in S is a prefix of another string in S. We write "r.e." instead of "recursively enumerable."

$\{0, 1\}^\infty$ is the set of infinite binary sequences, where an infinite binary sequence is infinite to the right but finite to the left. For any $\alpha \in \{0, 1\}^\infty$ and any $n \in \mathbb{N}$, we denote by $\alpha{\restriction}_n \in \{0, 1\}^*$ the first n bits of α. For any $S \subset \{0, 1\}^*$, the set $\{\alpha \in \{0, 1\}^\infty \mid \exists n \in \mathbb{N} \ \alpha{\restriction}_n \in S\}$ is denoted by $[S]^\prec$. Note that (i) $[S]^\prec \subset [T]^\prec$ for every $S \subset T \subset \{0, 1\}^*$, and (ii) for every set $S \subset \{0, 1\}^*$ there exists a prefix-free set $P \subset \{0, 1\}^*$ such that $[S]^\prec = [P]^\prec$.

Lebesgue outer measure $\mathcal{L}$ on $\{0, 1\}^\infty$ is a function mapping any subset of $\{0, 1\}^\infty$ to a non-negative real. In this paper, we use the following properties of $\mathcal{L}$.

Proposition 2.1 (Properties of Lebesgue outer measure on $\{0, 1\}^\infty$)**.**

(i) $\mathcal{L}\left([P]^\prec\right) = \sum_{x \in P} 2^{-|x|}$ *for every prefix-free set* $P \subset \{0, 1\}^*$. *Therefore* $\mathcal{L}(\emptyset) = \mathcal{L}\left([\emptyset]^\prec\right) = 0$ *and* $\mathcal{L}\left(\{0, 1\}^\infty\right) = \mathcal{L}\left([\{\lambda\}]^\prec\right) = 1$.

(ii) $\mathcal{L}(\mathcal{C}) \leq \mathcal{L}(\mathcal{D})$ *for every* $\mathcal{C} \subset \mathcal{D} \subset \{0, 1\}^\infty$.

(iii) $\mathcal{L}\left(\bigcup_i \mathcal{C}_i\right) \leq \sum_i \mathcal{L}(\mathcal{C}_i)$ *for every sequence* $\{\mathcal{C}_i\}_{i \in \mathbb{N}}$ *of subsets of* $\{0, 1\}^\infty$. $\qquad\qquad\square$

A function $f \colon \mathbb{N} \to \{0, 1\}^*$ or $f \colon \mathbb{N} \to \mathbb{Q}$ is called *computable* if there exists a deterministic Turing machine which on every input $n \in \mathbb{N}$ halts and outputs $f(n)$. A computable function is also called a *total recursive function*. A real a is called *computable* if there exists a computable function $g \colon \mathbb{N} \to \mathbb{Q}$ such that $|a - g(k)| < 2^{-k}$ for all $k \in \mathbb{N}$. We say that $\alpha \in \{0, 1\}^\infty$ is *computable* if the mapping $\mathbb{N} \ni n \mapsto \alpha{\restriction}_n$ is a computable function, which is equivalent to that the real $0.\alpha$ in base-two notation is computable.

2.1 Algorithmic randomness

In the following we concisely review some definitions and results of algorithmic randomness [7, 8, 23, 10]. The idea in algorithmic randomness is to think of a real, i.e., an infinite binary sequence, as random if it is in no *effective null set*. An effective null set is a subset $\mathcal{S}$ of $\{0,1\}^\infty$ such that $\mathcal{L}(\mathcal{S}) = 0$ and $\mathcal{S}$ has some type of effective property. To specify an algorithmic randomness notion, one has to specify a type of effective null set, which is usually done by introducing a test concept. Failing the test is the same as being in the null set. In this manner, various randomness notions, such as 2-randomness, weak 2-randomness, Demuth randomness, Martin-Löf randomness, Schnorr randomness, Kurtz randomness, have been introduced so far, and a hierarchy of algorithmic randomness notions has been developed (see [23, 10] for the detail).

Among all randomness notions, *Martin-Löf randomness* is a central one. This is because in many respects, Martin-Löf randomness is well-behaved, in that the many properties of Martin-Löf random infinite sequences do match our intuition of what random infinite sequence should look like. Moreover, the concept of Martin-Löf randomness is robust in the sense that it admits various equivalent definitions that are all natural and intuitively meaningful, as we will see in Theorem 2.4. Martin-Löf randomness is defined as follows based on the notion of *Martin-Löf test*.

Definition 2.2 (Martin-Löf randomness, Martin-Löf [18]). *A subset $\mathcal{C}$ of $\mathbb{N}^+ \times \{0,1\}^*$ is called a* Martin-Löf test *if $\mathcal{C}$ is an r.e. set and there exists a total recursive function $f \colon \mathbb{N}^+ \to \mathbb{Q} \cap (0,\infty)$ such that $\lim_{n\to\infty} f(n) = 0$ and for every $n \in \mathbb{N}^+$,*

$$\mathcal{L}\left([\mathcal{C}_n]^{\prec}\right) \le f(n),$$

where $\mathcal{C}_n = \left\{\, x \mid (n,x) \in \mathcal{C} \,\right\}$.

For any $\alpha \in \{0,1\}^\infty$, we say that α is Martin-Löf random *if for every Martin-Löf test $\mathcal{C}$ there exists $n \in \mathbb{N}^+$ such that $\alpha \notin [\mathcal{C}_n]^{\prec}$.*[1] □

Let $\mathcal{C}$ be a Martin-Löf test. Then, for each $k \in \mathbb{N}^+$, using (ii) of Proposition 2.1 we see that $\mathcal{L}\left(\bigcap_{n=1}^\infty [\mathcal{C}_n]^{\prec}\right) \le \mathcal{L}\left([\mathcal{C}_k]^{\prec}\right) \le f(k)$. On letting $k \to \infty$, we have $\mathcal{L}\left(\bigcap_{n=1}^\infty [\mathcal{C}_n]^{\prec}\right) = 0$. Thus, the set $\bigcap_{n=1}^\infty [\mathcal{C}_n]^{\prec}$ forms an effective null set in the notion of Martin-Löf randomness. Definition 2.2 says that an infinite binary sequence α is Martin-Löf random if α is not in the effective null set $\bigcap_{n=1}^\infty [\mathcal{C}_n]^{\prec}$ for any Martin-Löf test $\mathcal{C}$.

One of the equivalent variants of Martin-Löf randomness is Solovay randomness, which plays an important role in this paper, as well as Martin-Löf randomness.

Definition 2.3 (Solovay randomness, Solovay [29]). *A subset $\mathcal{C}$ of $\mathbb{N}^+ \times \{0,1\}^*$ is called a* Solovay test *if $\mathcal{C}$ is an r.e. set and*

$$\sum_{n=1}^\infty \mathcal{L}\left([\mathcal{C}_n]^{\prec}\right) < \infty.$$

For any $\alpha \in \{0,1\}^\infty$, we say that α is Solovay random *if for every Solovay test $\mathcal{C}$, there exists $N \in \mathbb{N}^+$ such that, for every $n \ge N$, $\alpha \notin [\mathcal{C}_n]^{\prec}$.* □

---

[1]Normally, Martin-Löf random is defined with fixing the total recursive function $f \colon \mathbb{N}^+ \to \mathbb{Q} \cap (0,\infty)$ to the form $f(n) = 2^{-n}$. However, the relaxation of the function f as in Definition 2.2 does not alter the class of Martin-Löf random infinite binary sequences.

For each Solovay test $\mathcal{C}$, we can show that $\mathcal{L}\left(\bigcap_{n=1}^{\infty}\bigcup_{k=n}^{\infty}[\mathcal{C}_k]^{\prec}\right) = 0$. The set $\bigcap_{n=1}^{\infty}\bigcup_{k=n}^{\infty}[\mathcal{C}_k]^{\prec}$ forms an effective null set in the notion of Solovay randomness.

The robustness of Martin-Löf randomness is mainly due to the fact that it admits characterizations based on the notion of program-size complexity, as shown in Theorem 2.4. The *program-size complexity* (or *Kolmogorov complexity*) $K(x)$ of a finite binary string x is defined as the length of the shortest binary input for a universal decoding algorithm U, called an *optimal prefix-free machine*, to output x (see Chaitin [7] for the detail). By the definition, $K(x)$ can be thought of as the randomness contained in the individual finite binary string x.

Theorem 2.4 (Schnorr [26], Chaitin [7], and Solovay [29], and Miller and Yu [21])**.** *For every $\alpha \in \{0,1\}^{\infty}$, the following conditions are equivalent:*

(i) α is Martin-Löf random.

(ii) α is Solovay random.

(iii) There exists $c \in \mathbb{N}$ such that, for all $n \in \mathbb{N}^+$, $n - c \leq K(\alpha{\restriction}_n)$.

(iv) $\sum_{n=1}^{\infty} 2^{n-K(\alpha{\restriction}_n)} < \infty$. $\qquad\square$

In particular, the condition (iii) means that the infinite binary sequence α is incompressible.

We denote by MLR the set of all infinite binary sequences which are Martin-Löf random. Since there are only countably infinitely many algorithms and every Martin-Löf test induces an effective null set, it is easy to show the following theorem.

Theorem 2.5 (Martin-Löf [18])**.** $\mathcal{L}(\mathsf{MLR}) = 1$. $\qquad\square$

3   Signature schemes and their security

We begin by presenting the general form of signature scheme whose security we consider in this paper. For modern cryptography in general, we refer the reader to Katz and Lindell [15].

In 1993 Bellare and Rogaway proposed the notion of *full-domain hash* (FDH) signature scheme in their original paper on the random oracle model [1]. They showed that the RSA-FDH signature scheme, which is an instantiation of the FDH signature scheme with the RSA function as a trapdoor permutation, is effectively existentially unforgeable under an adaptive chosen-message attack (EUF-ACMA secure) in the random oracle model under the RSA assumption (see Theorem 6.1; for the detail of RSA-FDH see also [15, Chapter 13]). In the first half of this paper, we consider a general form of the FDH signature scheme and give our results about the secure instantiation of the random oracle for that general scheme.

Let $\ell(n)$ be a polynomial with integer coefficients such that $\ell(n) > 0$ for all $n \in \mathbb{N}^+$. An *ℓ-function* is a function $H\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ such that $|H(n,x)| = \ell(n)$ for all $n \in \mathbb{N}$ and $x \in \{0,1\}^*$. For each ℓ-function H and $n \in \mathbb{N}$, we define a function $H_n\colon \{0,1\}^* \to \{0,1\}^{\ell(n)}$ by $H_n(x) = H(n,x)$. An ℓ-function serves as an instantiation of the random oracle, such as a cryptographic hash function.

Definition 3.1. *Let $\ell(n)$ be a polynomial. A* signature scheme relative to ℓ-functions *is a tuple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *of three polynomial-time algorithms such that, for every ℓ-function H,*

1. *The* key generation algorithm Gen *is a probabilistic algorithm which takes as input a security parameter* 1^n *and outputs a pair of keys* (pk, sk). *These are called the* public key *and the* private key, *respectively. We assume that n can be determined from each of pk and sk.*

2. *The* signing algorithm Sign *is a probabilistic algorithm which takes as input a private key sk and a* message $m \in \{0, 1\}^*$. *It is given oracle access to $H_n(\cdot)$, and then outputs a* signature σ, *denoted as* $\sigma \leftarrow \mathsf{Sign}_{sk}^{H_n(\cdot)}(m)$.

3. *The* verification algorithm Vrfy *is a deterministic algorithm which takes as input a public key pk, a massage m, and a signature σ. It is given oracle access to $H_n(\cdot)$, and then outputs a bit b, with $b = 1$ meaning* valid *and $b = 0$ meaning* invalid. *We write this as* $b := \mathsf{Vrfy}_{pk}^{H_n(\cdot)}(m, \sigma)$.

It is required that, for every ℓ-function H, for every $n \in \mathbb{N}^+$, for every (pk, sk) output by Gen(1^n), *and for every $m \in \{0, 1\}^*$,*

$$\mathsf{Vrfy}_{pk}^{H_n(\cdot)}(m, \mathsf{Sign}_{sk}^{H_n(\cdot)}(m)) = 1. \tag{4}$$

□

In general, a signature scheme is used in the following way. One party S, who acts as the *signer*, runs Gen(1^n) to obtain keys (pk, sk). The public key pk is then publicized as belonging to S; e.g., S can put the public key on its webpage or place it in some public directory. We assume that any other party is able to obtain a legitimate copy of S's public key. When S wants to transmit a message m, it computes $\sigma \leftarrow \mathsf{Sign}_{sk}^{H_n(\cdot)}(m)$ and sends (m, σ). Upon receipt of (m, σ), a receiver who knows pk can verify the authenticity of m by checking whether $\mathsf{Vrfy}_{pk}^{H_n(\cdot)}(m, \sigma) = 1$, or not. This establishes both that S sent m, and also that m was not modified in transmit. Note here that Definition 3.1 only defines the syntax of signature schemes and does not define the security of them at all, which is defined in what follows.

As the security notion of signature schemes, in this paper we consider *the existential unforgeability under adaptive chosen-message attacks* (*EUF-ACMA security*) as an example. We can show the same results for other security notions, such as the existential unforgeability against key only attacks (EUF-KOA security), the existential unforgeability against known-message attacks (EUF-KMA security), and the existential unforgeability against generic chosen-massage attacks (EUF-GCMA security), which are all weaker than the EUF-ACMA security.

Given a public key pk generated by a signer S to an adversary, we say that the adversary outputs a *forgery* if it outputs a message m along with a valid signature σ on m, and furthermore m was not previously signed by S using the private key sk which corresponds to pk. The EUF-ACMA security of a signature scheme means that an adversary cannot output a forgery even if it is allowed to obtain signatures on many other messages of its choice. The formal definition is given as follows.

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme relative to ℓ-functions, and consider the following experiment for a probabilistic polynomial-time adversary $\mathcal{A}$,[2] a parameter n, and a function G mapping a superset of $\{0, 1\}^{\leq q(n)}$ to $\{0, 1\}^{\ell(n)}$ where $q(n)$ is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}$ on the parameter n:

The signature experiment $\mathsf{Sig\text{-}forge}_{\mathcal{A}, \Pi}(n, G)$**:**

---

[2]Normally, a probabilistic (uniform) polynomial-time Turing machine is called a *probabilistic polynomial-time adversary* when it is used as an adversary against a cryptographic scheme.

1. $\mathsf{Gen}(1^n)$ *is run to obtain keys* (pk, sk).

2. *Adversary* $\mathcal{A}$ *is given pk and oracle access to both* $\mathsf{Sign}_{sk}^{G(\cdot)}(\cdot)$ *and* $G(\cdot)$. *(The first oracle returns a signature* $\mathsf{Sign}_{sk}^{G(\cdot)}(m')$ *for any message* m' *of the adversary's choice while having oracle access to* $G(\cdot)$ *of itself.) The adversary then outputs* (m, σ). *Let* $\mathcal{Q}$ *denotes the set of messages whose signatures were requested by* $\mathcal{A}$ *during its execution.*

3. *The output of the experiment is defined to be* 1 *if both* $m \notin \mathcal{Q}$ *and* $\mathsf{Vrfy}_{pk}^{G(\cdot)}(m, \sigma) = 1$ *hold true, and* 0 *otherwise.*

Here the function G serves as an instantiation of the random oracle. Since the running time of each of Sign, Vrfy, and $\mathcal{A}$ on the parameter n is at most $q(n)$, the lengths of the strings queried to the oracle $G(\cdot)$ by these three algorithms during their computations are at most $q(n)$. Thus the function G only have to be defined on the set $\{0, 1\}^{\leq q(n)}$.

On the one hand, the EUF-ACMA security of signature schemes relative to a specific ℓ-function is defined as follows. This form of the definition corresponds to the condition (3) with $\gamma = 0$ for the security of a signature scheme relative to a specific oracle $\{h_n\}$ considered in the introduction.

Definition 3.2. *Let* H *be an* ℓ-*function. A signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *relative to* ℓ-*functions is* existentially unforgeable under an adaptive chosen-message attack *(or* EUF-ACMA secure*) relative to* H *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and all* $d \in \mathbb{N}^+$ *there exists* $N \in \mathbb{N}^+$ *such that, for all* $n \geq N$,

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, H_n) = 1] \leq \frac{1}{n^d}.$$

$\square$

On the other hand, the EUF-ACMA security of signature schemes in the random oracle model is formulated as follows. This form of the definition corresponds to the condition (2), and is justified based on the consideration in the introduction.

Definition 3.3. *A signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *relative to* ℓ-*functions is* existentially unforgeable under an adaptive chosen-message attack *(or* EUF-ACMA secure*) in the random oracle model if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *and all* $d \in \mathbb{N}^+$ *there exists* $N \in \mathbb{N}^+$ *such that, for all* $n \geq N$,

$$\frac{1}{\#\mathsf{Func}_{\leq q(n)}^{\ell(n)}} \sum_{G \in \mathsf{Func}_{\leq q(n)}^{\ell(n)}} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, G) = 1] \leq \frac{1}{n^d},$$

where $q(n)$ *is the maximum value among the running time of* Sign, *the running time of* Vrfy, *and the running time of* $\mathcal{A}$ *on the parameter* n. $\square$

4 Conditions for secure instantiation of the random oracle

In this section, we present *equivalent* conditions for a specific oracle instantiating the random oracle to keep a signature scheme secure, using a concept of algorithmic randomness.

In order to apply the method of algorithmic randomness to the random oracle methodology, we identify an ℓ-function with an infinite binary sequence in the following manner: We first choose a particular bijective total recursive function $b\colon \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ with $b(k) = (b_1(k), b_2(k))$ as the standard one for use throughout the rest of this paper. We assume for convenience that, for every $k, l \in \mathbb{N}$, if $b_1(k) = b_1(l)$ and $k < l$ then $b_2(k) < b_2(l)$. For example, the inverse function of a function $c\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with $c(m, n) = (m + n)(m + n + 1)/2 + n$ can serve as such a function b. Then each ℓ-function $H\colon \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ is identified with the infinite binary sequence

$$H(b(0))H(b(1))H(b(2))H(b(3)) \cdots \cdots , \tag{5}$$

where the countably infinite finite binary strings $H(b(0)), H(b(1)), H(b(2)), H(b(3)), \dots$ are concatenated. Recall that we identify $\{0,1\}^*$ with $\mathbb{N}$, as explained in Section 2, and therefore each $b_2(k)$ is regarded as a finite binary string in (5). In what follows, we work with this intuition of the identification.

We will give the main result of this section, i.e., Theorem 4.10, in terms of Solovay randomness and Martin-Löf randomness. For that purpose we generalize these two randomness notions in Definitions 4.1 and 4.6, respectively.

Definition 4.1 (Solovay randomness with respect to an arbitrary set of Solovay tests)**.** *Let S be a set of Solovay tests. For any $\alpha \in \{0,1\}^\infty$, we say that α is Solovay random with respect to S if for every Solovay test $\mathcal{C} \in S$, there exists $N \in \mathbb{N}^+$ such that, for every $n \geq N$, $\alpha \notin [\mathcal{C}_n]^\prec$.* $\square$

Definition 4.2. *Let $\ell(n)$ be a polynomial, and let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme relative to ℓ-functions.*

For each probabilistic polynomial-time adversary $\mathcal{A}$ and each $d, n \in \mathbb{N}^+$ we define a subset $[C_{\mathcal{A},d,n}]^\prec$ of $\{0,1\}^\infty$ as the set of all ℓ-functions H such that

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, H_n) = 1] > \frac{1}{n^d}.$$

To be precise, we define a subset $C_{\mathcal{A},d,n}$ of $\{0,1\}^*$ as the set of all finite binary strings of the form $x_0 G(\lambda) x_1 G(0) x_2 G(1) x_3 \cdots x_L G(1^q)$ such that the following properties (i), (ii), (iii), and (iv) hold for q, L, $x_0, x_1, x_2, x_3, \dots, x_L$, and G:

(i) *q is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}$ on the parameter n.*

(ii) *$L + 1 = \#\{0,1\}^{\leq q}$ (i.e., $L = 2^{q+1} - 2$).*

(iii) *For each $j \in \{0, \dots, L\}$, $x_j \in \{0,1\}^*$ and*

$$|x_0 G(\lambda) x_1 G(0) x_2 G(1) x_3 \cdots x_j| = \sum_{k=0}^{k_j - 1} \ell(b_1(k))$$

where k_j is a natural number such that $b(k_j) = (n, j)$.

(iv) *$G \in \mathsf{Func}_{\leq q}^{\ell(n)}$ and*

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, G) = 1] > \frac{1}{n^d}.$$

10

We then define $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ *as the class of all subsets* $\mathcal{C}$ *of* $\mathbb{N}^+ \times \{0,1\}^*$ *for which there exist a probabilistic polynomial-time adversary* $\mathcal{A}$ *and* $d \geq 2$ *such that* $\mathcal{C} = \{(n, y) \mid n \in \mathbb{N}^+ \ \& \ y \in C_{\mathcal{A},d,n}\}$. $\quad\square$

Theorem 4.3. *Let* $\ell(n)$ *be a polynomial. Suppose that a signature scheme* $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *relative to* ℓ-functions is EUF-ACMA secure in the random oracle model. Then $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ *contains only Solovay tests.* $\quad\square$

In order to prove Theorem 4.3, we need the following two lemmas.

Lemma 4.4. *Let* $f_1, \ldots, f_N$ *be reals. Suppose that* $\frac{1}{N} \sum_{i=1}^{N} f_i \leq \varepsilon$. *Then, for every* $\alpha > 0$, *the number of* i *for which* $\alpha\varepsilon < f_i$ *is less than* N/α.

Proof. We prove the contraposition of Lemma 4.4. Assume that the number of i for which $\alpha\varepsilon < f_i$ is at least N/α. Then $\sum_{i=1}^{N} f_i > \alpha\varepsilon N/\alpha = \varepsilon N$ and therefore $\frac{1}{N} \sum_{i=1}^{N} f_i > \varepsilon$. $\quad\square$

Lemma 4.5. *Let* $d \geq 2$. *Then* $\sum_{k=n}^{\infty} 1/k^d \leq 2/n$ *for every* $n \in \mathbb{N}^+$.

Proof. In the case of $n \geq 2$, we have

$$\sum_{k=n}^{\infty} \frac{1}{k^d} \leq \sum_{k=n}^{\infty} \int_{k-1}^{k} \frac{1}{k^d} = \int_{n-1}^{\infty} \frac{1}{x^d} dx = \frac{1}{(d-1)(n-1)^{d-1}} \leq \frac{1}{n-1} \leq \frac{1}{n-n/2} = \frac{2}{n}. \qquad (6)$$

On the other hand, in the case of $n = 1$, using (6) we have

$$\sum_{k=n}^{\infty} \frac{1}{k^d} = 1 + \sum_{k=2}^{\infty} \frac{1}{k^d} \leq 1 + \frac{2}{2} = 2 = \frac{2}{n}.$$

Thus $\sum_{k=n}^{\infty} 1/k^d \leq 2/n$ holds in any case. $\quad\square$

Proof of Theorem 4.3. Let $\mathcal{C} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$. Then there exist a probabilistic polynomial-time adversary $\mathcal{A}$ and $d \geq 2$ such that, for every $n \in \mathbb{N}^+$, $\mathcal{C}_n = C_{\mathcal{A},d,n}$. Suppose that Π is EUF-ACMA secure in the random oracle model. Then it follows from Definition 3.3 that there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,

$$\frac{1}{\#\mathsf{Func}_{\leq q(n)}^{\ell(n)}} \sum_{G \in \mathsf{Func}_{\leq q(n)}^{\ell(n)}} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, G) = 1] \leq \frac{1}{n^{2d}}, \qquad (7)$$

where $q(n)$ is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}$ on the parameter n.

On the one hand, it follows from Definition 4.2 that $\mathcal{C}$ is an r.e. set, since the dyadic rational

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, G) = 1]$$

is computable, given n and $G \in \mathsf{Func}_{\leq q(n)}^{\ell(n)}$.

On the other hand, using (7) and Lemma 4.4 with $\varepsilon = 1/n^{2d}$ and $\alpha = n^d$, we see that, for every $n \geq N$,

$$\# \left\{ G \in \mathsf{Func}_{\leq q(n)}^{\ell(n)} \;\middle|\; \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, G) = 1] > \frac{1}{n^d} \right\} < \frac{\#\mathsf{Func}_{\leq q(n)}^{\ell(n)}}{n^d}.$$

11

Since
$$\#\mathsf{Func}_{\leq q(n)}^{\ell(n)} = 2^{\ell(n)\#\{0,1\}^{\leq q(n)}},$$
it follows from Definition 4.2 and (i) of Proposition 2.1 that

$$\sum_{n=N}^{\infty} \mathcal{L}\left([\mathcal{C}_n]^{\prec}\right) = \sum_{n=N}^{\infty} \sum_{y \in \mathcal{C}_n} 2^{-|y|} < \sum_{n=N}^{\infty} \frac{\#\mathsf{Func}_{\leq q(n)}^{\ell(n)}}{n^d} 2^{-\ell(n)\#\{0,1\}^{\leq q(n)}} = \sum_{n=N}^{\infty} \frac{1}{n^d} < \infty,$$

where the last inequality follows from Lemma 4.5. Thus $\mathcal{C}$ is a Solovay test. $\square$

Definition 4.6 (Martin-Löf randomness with respect to an arbitrary set of Martin-Löf tests). *Let S be a set of Martin-Löf tests. For any $\alpha \in \{0,1\}^{\infty}$, we say that α is Martin-Löf random with respect to S if for every Martin-Löf test $\mathcal{C} \in S$, there exists $n \in \mathbb{N}^+$ such that $\alpha \notin [\mathcal{C}_n]^{\prec}$.* $\square$

Definition 4.7. *Let $\ell(n)$ be a polynomial, and let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme relative to ℓ-functions. We define $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ as the class of all subsets $\mathcal{C}$ of $\mathbb{N}^+ \times \{0,1\}^*$ for which there exists $\mathcal{D} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ such that, for every $n \in \mathbb{N}^+$, $\mathcal{C}_n = \bigcup_{k=n}^{\infty} \mathcal{D}_k$.* $\square$

Theorem 4.8. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. Then $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ contains only Martin-Löf tests.[3]*

Proof. Let $\mathcal{C} \in \mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$. Then there exists $\mathcal{D} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ such that, for every $n \in \mathbb{N}^+$, $\mathcal{C}_n = \bigcup_{k=n}^{\infty} \mathcal{D}_k$. Suppose that Π is EUF-ACMA secure in the random oracle model. It follows from Theorem 4.3 that $\mathcal{D}$ is a Solovay test. It is then easy to see that $\mathcal{C}$ is an r.e. set, since $\mathcal{D}$ is an r.e. set. On the other hand, since $\mathcal{D} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$, there exist a probabilistic polynomial-time adversary $\mathcal{A}$ and $d \geq 2$ such that, for every $n \in \mathbb{N}^+$, $\mathcal{D}_n = C_{\mathcal{A},d,n}$. Then, in the same manner as the proof of Theorem 4.3 we can show that there exists $N \in \mathbb{N}^+$ such that, for every $n \geq N$,

$$\#\left\{ G \in \mathsf{Func}_{\leq q(n)}^{\ell(n)} \,\middle|\, \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n,G) = 1] > \frac{1}{n^d} \right\} < \frac{\#\mathsf{Func}_{\leq q(n)}^{\ell(n)}}{n^d},$$

where $q(n)$ is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}$ on the parameter n. It follows from (i) and (iii) of Proposition 2.1 and Definition 4.2 that, for each $n \geq N$,

$$\mathcal{L}\left([\mathcal{C}_n]^{\prec}\right) \leq \sum_{k=n}^{\infty} \mathcal{L}\left([\mathcal{D}_k]^{\prec}\right) = \sum_{k=n}^{\infty} \sum_{y \in \mathcal{D}_k} 2^{-|y|} < \sum_{k=n}^{\infty} \frac{\#\mathsf{Func}_{\leq q(k)}^{\ell(k)}}{k^d} 2^{-\ell(k)\#\{0,1\}^{\leq q(k)}} = \sum_{k=n}^{\infty} \frac{1}{k^d} \leq \frac{2}{n},$$

where the last inequality follows from Lemma 4.5. Thus $\mathcal{C}$ is a Martin-Löf test. $\square$

Obviously, the following proposition holds.

Proposition 4.9. *Let $\alpha \in \{0,1\}^{\infty}$.*

---

[3]In fact, $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ contains only Schnorr tests, where a Schnorr test is defined as a Martin-Löf test $\mathcal{C} \subset \mathbb{N}^+ \times \{0,1\}^*$ such that $\mathcal{L}\left([\mathcal{C}_n]^{\prec}\right)$ is computable uniformly in n. For the detail of Schnorr tests, see e.g. Section 3.5 of Nies [23].

(i) *For every set S of Martin-Löf tests, if α is Martin-Löf random then α is Martin-Löf random with respect to S.*

(ii) *For every set S of Solovay tests, if α is Solovay random then α is Solovay random with respect to S.* □

The following theorem gives equivalent conditions for a specific oracle instantiating the random oracle to keep the EUF-ACMA security of a signature scheme originally proved in the random oracle model, in terms of algorithmic randomness.

Theorem 4.10 (Main result I). *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. Let H be an ℓ-function. Then the following conditions are equivalent:*

(i) *Π is EUF-ACMA secure relative to H.*

(ii) *H is Solovay random with respect to $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$.*

(iii) *H is Martin-Löf random with respect to $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$.*

Proof. First we show the equivalence between the conditions (i) and (ii). The negation of the condition (i) is that there exist a probabilistic polynomial-time adversary $\mathcal{A}$ and $d \geq 2$ such that, for infinitely many $n \in \mathbb{N}^+$,

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n, H_n) = 1] > \frac{1}{n^d}.$$

However, from Definition 4.2, it is easy to see that this is equivalent to the condition that there exists $\mathcal{C} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ such that, for infinitely many $n \in \mathbb{N}^+$, $H \in [\mathcal{C}_n]^{\prec}$. This is further equivalent to the condition that H is not Solovay random with respect to $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$, since $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ contains only Solovay tests by Theorem 4.3. Thus the conditions (i) and (ii) are equivalent to each other.

Next we show the equivalence between the conditions (ii) and (iii). Suppose that $\mathcal{C} \in \mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ and $\mathcal{D} \in \mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ satisfy that $\mathcal{C}_n = \bigcup_{k=n}^{\infty} \mathcal{D}_k$ for all $n \in \mathbb{N}^+$. Then the condition that $H \in [\mathcal{C}_n]^{\prec}$ for all $n \in \mathbb{N}^+$ is equivalent to the condition that $H \in [\mathcal{D}_n]^{\prec}$ for infinitely many $n \in \mathbb{N}^+$. Note here that $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ contains only Martin-Löf tests by Theorem 4.8, and $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ contains only Solovay tests by Theorem 4.3. Thus, H is not Martin-Löf random with respect to $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$ if and only if H is not Solovay random with respect to $\mathsf{S\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$. This completes the proof. □

As noted in the previous section, Theorem 5.3 holds for other security notions for signature schemes, such as the EUF-GCMA security, in place of the EUF-ACMA security. Thus, given arbitrary security notion UF and signature scheme Π which is UF secure in the random oracle model, one can define a variant of Martin-Löf randomness, i.e., Martin-Löf randomness with respect to $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{UF}}$, which gives a equivalent condition for a specific oracle instantiating the random oracle in Π to keep the UF security. In this manner, given a security notion and a signature scheme satisfying this security notion in the random oracle model, one can define an algorithmic randomness notion which is specified by an appropriate type of effective null sets based on these security notion and scheme, and which corresponds exactly to the secure instantiation of the random oracle with respect to this security notion.

In the next section we show in Theorem 5.3 that a signature scheme Π can be EUF-ACMA secure relative to some *computable* ℓ-function H, in the case where Π satisfies a stronger security notion, called the *effective* EUF-ACMA security, in the random oracle model. Hence, in such a case, it follows from Theorem 4.10 that there exists a *computable* infinite binary sequence H which is Martin-Löf random with respect to $\mathsf{ML\text{-}TEST}_{\Pi}^{\mathsf{EUF\text{-}ACMA}}$.

The following theorem shows that the EUF-ACMA security proved in the random oracle model is firmly maintained after instantiating the random oracle by a random real.

Theorem 4.11. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. For every ℓ-function H, if H is Martin-Löf random then Π is EUF-ACMA secure relative to H.*

Proof. The result follows immediately from (i) of Proposition 4.9 and Theorem 4.10. $\square$

The following theorem shows that a specific oracle instantiating the random oracle almost surely keeps the EUF-ACMA security of a signature scheme originally proved in the random oracle model.

Theorem 4.12. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. Then $\mathcal{L}(\mathsf{EUF}_{\Pi}^{\mathsf{acma}}) = 1$, where $\mathsf{EUF}_{\Pi}^{\mathsf{acma}}$ is the set of all ℓ-functions H such that Π is EUF-ACMA secure relative to H.*

Proof. The result follows immediately from Theorem 2.5, Theorem 4.11, and (i) and (ii) of Proposition 2.1. $\square$

Impagliazzo and Rudich [14] showed a similar result to Theorem 4.12 for a one-way permutation and derived the negative result about the existence of a secure secret key agreement protocol.

5 Secure instantiation of the random oracle by computable function

Let H be an ℓ-function. We say that H is *computable* if there exists a deterministic Turing machine which on every input (n, x) halts and outputs $H(n, x)$. On the other hand, we say that H is *polynomial-time computable* if there exists a deterministic Turing machine which on every input $(1^n, x)$ operates and outputs $H(n, x)$ within time polynomial in n and $|x|$.

Conjecture 1 below means that, in the case where a signature scheme Π satisfies a certain condition $\mathcal{C}$, the EUF-ACMA security of Π originally proved in the random oracle model can be firmly maintained in the standard model after instantiating the random oracle by some polynomial-time computable ℓ-function (or some polynomial-time computable family of ℓ-functions).

Conjecture 1. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. If Π satisfies $\mathcal{C}$, then there exists a polynomial-time computable ℓ-function (or a polynomial-time computable family of ℓ-functions) relative to which Π is EUF-ACMA secure.* $\square$

Note that an appropriate restriction on a signature scheme Π, i.e., the condition $\mathcal{C}$ on Π, might be necessary to prove Conjecture 1, due to the negative results in the secure instantiation of the random oracle by Canetti, Goldreich, and Halevi [5], who show "contrived" signature schemes (and

encryption schemes) that are secure in the random oracle model but are demonstrably insecure for *any* concrete instantiation of the random oracle. At present, however, it would seem very difficult to prove it with identifying an appropriate nontrivial condition $\mathcal{C}$.

The second best thing is to investigate whether Conjecture 2 below holds true or not, where we consider the instantiation of the random oracle by simply a computable ℓ-function, which is not necessarily polynomial-time computable.

Conjecture 2. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is EUF-ACMA secure in the random oracle model. Then there exists a computable ℓ-function H such that Π is EUF-ACMA secure relative to H.* $\square$

In what follows, we show that an "effective" variant of Conjecture 2 holds true. We introduce the notion of *effective EUF-ACMA security*, which is a constructive strengthen of the conventional (non-constructive) notions of EUF-ACMA security. In terms of Definitions 3.2 and 3.3 for the conventional EUF-ACMA security, the "effectiveness" means that the number N in the definitions can be computed, given the code of an adversary $\mathcal{A}$ and a number d. To begin with a formal definition, we choose a particular recursive enumeration $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots$ of all probabilistic polynomial-time adversaries as the standard one for use throughout the rest of this section. It is easy to show that such an enumeration exists. In fact, the kth probabilistic polynomial-time adversary $\mathcal{A}_k$ can be chosen as a probabilistic Turing machine obtained by executing the kth probabilistic Turing machine $\mathcal{M}_k$ in at most $n^k + k$ steps, where n is the length of the input of $\mathcal{M}_k$.

On the one hand, the effective EUF-ACMA security relative to a specific ℓ-function is defined as follows.

Definition 5.1. *Let H be an ℓ-function. A signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is* effectively existentially unforgeable under an adaptive chosen-message attack *(or* effectively EUF-ACMA secure*) relative to H if there exists a computable function $f\colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i,d)$ then*

$$\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i,\Pi}(n, H_n) = 1] \leq \frac{1}{n^d}.$$

$\square$

Obviously, if a signature scheme Π relative to ℓ-functions is effectively EUF-ACMA secure relative to H, then Π is simply EUF-ACMA secure relative to H.

On the other hand, the effective EUF-ACMA security in the random oracle model is defined as follows.

Definition 5.2. *A signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is* effectively existentially unforgeable under an adaptive chosen-message attack *(or* effectively EUF-ACMA secure*) in the random oracle model if there exists a computable function $f\colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i,d)$ then*

$$\frac{1}{\#\mathsf{Func}_{\leq q_i(n)}^{\ell(n)}} \sum_{G \in \mathsf{Func}_{\leq q_i(n)}^{\ell(n)}} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i,\Pi}(n, G) = 1] \leq \frac{1}{n^d},$$

where $q_i(n)$ is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}_i$ on the parameter n. $\square$

Obviously, if a signature scheme Π relative to ℓ-functions is effectively EUF-ACMA secure in the random oracle model, then Π is simply EUF-ACMA secure in the random oracle model.

The effective variant of Conjecture 2 is then presented as follows.

Theorem 5.3 (Main result II). *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is effectively EUF-ACMA secure in the random oracle model. Then there exists a computable ℓ-function H such that Π is effectively EUF-ACMA secure relative to H.* $\qquad\square$

In order to prove Theorem 5.3, we need Lemmas 4.4 and 4.5 in the previous section, and Lemma 5.4 below. The last one is Exercise 1.9.21 of Nies's textbook [23] of algorithmic randomness.[4] In Section 8 we will prove a modification of Lemma 5.4, i.e., Theorem 8.2. The proof of Lemma 5.4 can be obtained by simplifying the proof of Theorem 8.2.

Lemma 5.4. *Let S be an r.e. subset of $\{0,1\}^*$. Suppose that $\mathcal{L}\left([S]^{\prec}\right) < 1$ and $\mathcal{L}\left([S]^{\prec}\right)$ is a computable real. Then there exists $\alpha \in \{0,1\}^{\infty}$ such that α is computable and $\alpha \notin [S]^{\prec}$.* $\qquad\square$

Proof of Theorem 5.3. Suppose that a signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ relative to ℓ-functions is effectively EUF-ACMA secure in the random oracle model. Then there exists a computable function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, d)$ then

$$\frac{1}{\#\mathsf{Func}^{\ell(n)}_{\leq q_i(n)}} \sum_{G \in \mathsf{Func}^{\ell(n)}_{\leq q_i(n)}} \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i, \Pi}(n, G) = 1] \leq \frac{1}{n^d},$$

where $q_i(n)$ is the maximum value among the running time of Sign, the running time of Vrfy, and the running time of $\mathcal{A}_i$ on the parameter n. Note that the value $q_i(n)$ can be computed, given i and n. It follows from Lemma 4.4 that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, 2d)$ then

$$\#\left\{ G \in \mathsf{Func}^{\ell(n)}_{\leq q_i(n)} \,\middle|\, \Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i, \Pi}(n, G) = 1] > \frac{1}{n^d} \right\} < \frac{\#\mathsf{Func}^{\ell(n)}_{\leq q_i(n)}}{n^d}. \tag{8}$$

For each $i, d, n \in \mathbb{N}^+$ we define a subset $C_{i,d,n}$ of $\{0,1\}^*$ as $C_{\mathcal{A}_i, d, n}$ (see Definition 4.2). Since $\#\mathsf{Func}^{\ell(n)}_{\leq q_i(n)} = 2^{\ell(n)\#\{0,1\}^{\leq q_i(n)}}$, it follows from Definition 4.2, (i) of Proposition 2.1, and (8) that, for each $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, 2d)$ then

$$\mathcal{L}\left([C_{i,d,n}]^{\prec}\right) = \sum_{s \in C_{i,d,n}} 2^{-|s|} < \frac{\#\mathsf{Func}^{\ell(n)}_{\leq q_i(n)}}{n^d} 2^{-\ell(n)\#\{0,1\}^{\leq q_i(n)}} = \frac{1}{n^d}. \tag{9}$$

We choose a particular computable bijection

$$\varphi \colon \mathbb{N}^+ \to \{ (i, d) \mid i \in \mathbb{N}^+ \ \& \ d \geq 2 \},$$

and define $(\varphi_1(m), \varphi_2(m)) = \varphi(m)$. We then define a computable function $g \colon \mathbb{N}^+ \to \mathbb{N}^+$ by $g(m) = \{f(\varphi_1(m), 2\varphi_2(m)) + 1\}^{m+1}$. For each $m \in \mathbb{N}^+$, we define a subset C_m of $\{0,1\}^*$ by

$$C_m = \bigcup_{n=g(m)}^{\infty} C_{\varphi_1(m), \varphi_2(m), n}. \tag{10}$$

---

[4]Lemma 5.4 can be used to prove the non-existence of universal Schnorr test for the notion of Schnorr randomness for an infinite binary sequence. See Fact 3.5.9 of [23] for the detail.

It follows from (iii) of Proposition 2.1, (9), and Lemma 4.5 that, for each $m \in \mathbb{N}^+$,

$$\mathcal{L}\left([C_m]^{\prec}\right) \leq \sum_{n=g(m)}^{\infty} \mathcal{L}\left(\left[C_{\varphi_1(m),\varphi_2(m),n}\right]^{\prec}\right) < \sum_{n=g(m)}^{\infty} \frac{1}{n^{\varphi_2(m)}} \leq \frac{2}{g(m)} \leq \frac{1}{2^m}. \tag{11}$$

We then define C by

$$C = \bigcup_{m=1}^{\infty} C_m. \tag{12}$$

Therefore, using (iii) of Proposition 2.1,

$$\mathcal{L}\left([C]^{\prec}\right) \leq \sum_{m=1}^{\infty} \mathcal{L}\left([C_m]^{\prec}\right) < \sum_{m=1}^{\infty} \frac{1}{2^m} = 1. \tag{13}$$

Next we show that C is an r.e. subset of $\{0,1\}^*$. It follows from Definition 4.2 that, given i, d, and n, one can decide the finite subset $C_{i,d,n}$ of $\{0,1\}^*$, since the dyadic rational $\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i,\Pi}(n,G) = 1]$ is computable, given i, n, and $G \in \mathsf{Func}_{\leq q_i(n)}^{\ell(n)}$. Thus, since φ and g are computable functions, it follows from (10) and (12) that C is an r.e. subset of $\{0,1\}^*$.

We then show that $\mathcal{L}\left([C]^{\prec}\right)$ is a computable real. For each $k \in \mathbb{N}$, we define a finite subset D_k of C by

$$D_k = \bigcup_{m=1}^{k} \bigcup_{n=g(m)}^{g(m)2^k - 1} C_{\varphi_1(m),\varphi_2(m),n}.$$

Given $k \in \mathbb{N}$, one can decides the finite set D_k, since φ and g are computable functions and moreover one can decide the finite set $C_{i,d,n}$, given i, d, and n. Therefore, given $k \in \mathbb{N}$, one can calculate the dyadic rational $\mathcal{L}\left([D_k]^{\prec}\right)$ based on (i) of Proposition 2.1. On the other hand, note that

$$C \setminus D_k \subset \left(\bigcup_{m=1}^{k} \bigcup_{n=g(m)2^k}^{\infty} C_{\varphi_1(m),\varphi_2(m),n} \right) \cup \bigcup_{m=k+1}^{\infty} C_m.$$

Thus, using (ii) and (iii) of Proposition 2.1, (9), Lemma 4.5, and (11) we see that, for each $k \in \mathbb{N}$,

$$\mathcal{L}\left([C \setminus D_k]^{\prec}\right) \leq \sum_{m=1}^{k} \sum_{n=g(m)2^k}^{\infty} \mathcal{L}\left(\left[C_{\varphi_1(m),\varphi_2(m),n}\right]^{\prec}\right) + \sum_{m=k+1}^{\infty} \mathcal{L}\left([C_m]^{\prec}\right)$$

$$< \sum_{m=1}^{k} \frac{2}{g(m)2^k} + \sum_{m=k+1}^{\infty} \frac{1}{2^m} \leq \sum_{m=1}^{k} \frac{1}{2^{m+k}} + \frac{1}{2^k} < \frac{1}{2^{k-1}}.$$

Therefore, since $[C]^{\prec} = [D_{k+1}]^{\prec} \cup [C \setminus D_{k+1}]^{\prec}$, using (ii) and (iii) of Proposition 2.1 we have

$$\left|\mathcal{L}\left([C]^{\prec}\right) - \mathcal{L}\left([D_{k+1}]^{\prec}\right)\right| \leq \mathcal{L}\left([C \setminus D_{k+1}]^{\prec}\right) \leq 2^{-k}$$

for each $k \in \mathbb{N}$. Hence, $\mathcal{L}\left([C]^{\prec}\right)$ is a computable real.

Now, it follows from Lemma 5.4 that there exists $H \in \{0,1\}^{\infty}$ such that H is computable and $H \notin [C]^{\prec}$. Since H is computable as an infinite binary sequence, it is easy to see that H is also

17

computable as an ℓ-function. On the other hand, let $i, d, n \in \mathbb{N}^+$ with $n \geq g(\varphi^{-1}(i, d+1))$. We then define $m = \varphi^{-1}(i, d+1)$, i.e., $\varphi(m) = (i, d+1)$. Since $H \notin [C]^{\prec}$ and $n \geq g(m)$, it follows from (12) and (10) that $H \notin [C_{\varphi1(m),\varphi2(m),n}]^{\prec} = [C_{i,d+1,n}]^{\prec}$. Therefore, based on the identification (5) of an ℓ-function with an infinite binary sequence, we see that the function $H_n \colon \{0, 1\}^* \to \{0, 1\}^{\ell(n)}$ satisfies that $\Pr[\mathsf{Sig\text{-}forge}_{\mathcal{A}_i, \Pi}(n, H_n) = 1] \leq 1/n^{d+1} < 1/n^d$. Thus, since the mapping $\mathbb{N}^+ \times \mathbb{N}^+ \ni (i, d) \mapsto g(\varphi^{-1}(i, d+1))$ is a computable function, it follows from Definition 5.1 that Π is effectively EUF-ACMA secure relative to H. $\quad\square$

6   Computable analysis on cryptography

In this section, we show that the effective security notions introduced in the previous section are a natural alternative to the conventional security notions in modern cryptography.

In Definitions 3.2 and 3.3 for the conventional EUF-ACMA security, the number N is only required to exist, depending on the adversary $\mathcal{A}$ and the number d, that is, the success probability of the attack by an adversary $\mathcal{A}$ on a security parameter n is required to be less than $1/n^d$ for all sufficiently large n, where the lower bound of such n is not required to be computable from $\mathcal{A}$ and d. On the other hand, in Definitions 5.1 and 5.2 for the effective EUF-ACMA security, it is required that the lower bound N of such n can be computed from the code of $\mathcal{A}$ and d.

In modern cryptography based on computational security, it is important to choose the security parameter n of a cryptographic scheme as small as possible to the extent that the *security requirements* are satisfied, in order to make the efficiency of the scheme as high as possible. For that purpose, it is desirable to be able to calculate a concrete value of N, given the code of $\mathcal{A}$ and d, since N gives a lower bound of the security parameter for which the security requirements specified by $\mathcal{A}$ and d are satisfied. This results in the notion of effective security.

Does the replacement of the conventional security notions by the corresponding effective security notions bring difficulties to modern cryptography over all ? We do not think so. It would seem plausible that all the conventional security notions can be replaced by the corresponding effective security notions in modern cryptography with little cost. As an example, let us consider the EUF-ACMA security of the RSA-FDH signature scheme under the RSA assumption and its effective counterpart. Let $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{RSA}}(n)$ be the success probability of an algorithm $\mathcal{A}$ in solving the RSA problem on a security parameter n. On the one hand, the (conventional) RSA assumption is defined as the condition that, for all probabilistic polynomial-time algorithms $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,

$$\mathsf{Succ}_{\mathcal{A}}^{\mathsf{RSA}}(n) \leq \frac{1}{n^d}.$$

On the other hand, the *effective* RSA assumption is defined as the condition that there exists a computable function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, d)$ then

$$\mathsf{Succ}_{\mathcal{A}_i}^{\mathsf{RSA}}(n) \leq \frac{1}{n^d},$$

where $\mathcal{A}_i$ is the ith algorithm in a particular recursive enumeration of all probabilistic polynomial-time algorithms. Now, recall the following theorem.

Theorem 6.1 (Bellare and Rogaway [1])**.** *RSA-FDH is EUF-ACMA secure in the random oracle model under the RSA assumption.* $\quad\square$

By analyzing the proof of Theorem 6.1 given in [1], we can see that the following effective version of Theorem 6.1 holds. We can do this task very easily, compared with the non-triviality of the original proof itself.

Theorem 6.2. *RSA-FDH is effectively EUF-ACMA secure in the random oracle model under the effective RSA assumption.* □

Note that the effective RSA assumption seems more difficult to prove than the RSA assumption. However, in modern cryptography based on computational security, we must make a computational assumption, such as the RSA assumption, somehow to guarantee the security of a cryptographic scheme. Since making any computational assumption does not cost at all in the development of theory of cryptography, making the effective RSA assumption instead of the RSA assumption would not seem to bring any trouble to modern cryptography. In this manner, we would expect that all the conventional security notions can be replaced by the corresponding effective security notions in modern cryptography with little cost. Thus, it would seem plausible that we can easily reconstruct the theory of cryptography based on the effective security notions instead of the conventional security notions.

In the above, we consider the validity of the effective security notions in modern cryptography. Actually, it would seem more natural to require that the functions $f\colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ in Definitions 5.1 and 5.2 are polynomial-time computable rather than simply computable. We call this type of effective security *polynomial-time effective security*. Conjecture 3 below is a polynomial-time effective version of Conjecture 1, and states that *the security in the random oracle model implies one in the standard model.* In the future, it would be challenging to prove Conjecture 3 (or its appropriate modification) with identifying an appropriate computational assumption COMP and an appropriate nontrivial condition $\mathcal{C}$ on a signature scheme Π.

Conjecture 3. *Let $\ell(n)$ be a polynomial. Suppose that a signature scheme Π relative to ℓ-functions is* polynomial-time *effectively EUF-ACMA secure in the random oracle model. Under the assumption* COMP*, if Π satisfies the condition $\mathcal{C}$, then there exists a* polynomial-time computable *ℓ-function (or a* polynomial-time computable *family of ℓ-functions) relative to which Π is* polynomial-time *effectively EUF-ACMA secure.* □

Note that the computational assumption COMP should be needed in Conjecture 3. Without this assumption, Conjecture 3 implies that the complexity class P is a proper subclass of the class NP, unless no signature scheme Π satisfies the condition $\mathcal{C}$. Thus, Conjecture 3 without the computational assumption COMP would become very difficult to prove. Actually, in the random oracle methodology, the random oracle is instantiated by a concrete cryptographic hash function such as the SHA hash functions (without adequate theoretical reason). Thus, from a theoretical point of view, it would seem reasonable to assume at least the existence of a *collision resistant hash function*[5] as the computational assumption COMP.

Computable analysis [24, 30] is a branch of computation theory which studies the computability and the computational complexity of mathematical notions appearing in analysis. It is closely related to algorithmic randomness. In particular, computable analysis considers the notion of the *effective convergence* of a sequence of reals, where a sequence $\{a_n\}_{n\in\mathbb{N}^+}$ of reals is called *converges effectively* to a real α if there exists a computable function $f\colon \mathbb{N}^+ \to \mathbb{N}^+$ such that, for every $N, n \in$

---

[5]See [15, Chapter 4] for the detail of collision resistant hash function.

$\mathbb{N}^+$, if $n \geq f(N)$ then $|a_n - \alpha| < 1/N$. On the one hand, we can see that the form of the definition of the convergence of a sequence of reals in analysis well corresponds to the definitions of security in modern cryptography based on computational security, such as Definitions 3.2 and 3.3. On the other hand, we can see that the notion of the effective convergence of a sequence of reals in computable analysis well corresponds to the effective security notions introduced in Definitions 5.1 and 5.2. Thus, the replacement of the conventional security notion by the corresponding effective security notion moder cryptography is just regarded as performing computable analysis over cryptography. The results of the previous section shows that *performing computable analysis over cryptography results in the secure instantiation of the random oracle.*

In what follows we continue to perform computable analysis over cryptography by introducing the notion of *effective hardness* for computational problems, whose hardness is used as a computational assumption to prove the security of a cryptographic scheme in moder cryptography. In particular, we consider the *discrete logarithm problem* and the *Diffie-Hellman problem* in the *generic group model*, and investigate the secure instantiation of the generic group, i.e., a random encoding of the group elements, in what follows.

7   The discrete logarithm problem in the generic group model

In this section we review the discrete logarithm problem in the generic group model. For the discrete logarithm problem *in the standard model* and its related problems, such as the Diffie-Hellman problem in the standard model, we refer the reader to Katz and Lindell [15, Chapter 7].

Shoup [27] introduced the notion of *generic algorithm* to study the computational complexity of the discrete logarithm and related problems in the *generic group model*, where the generic algorithm does not exploit any special properties of the encodings of group elements, other than the property that each group element is encoded as a unique binary string. Formally, a generic algorithm is defined as follows.

For any integer $N \geq 2$, we denote by $\mathbb{Z}_N$ the *additive group of integers modulo N* and sometimes the set $\{0, 1, \ldots, N-1\}$. For any $n \in \mathbb{N}^+$, an *encoding function into n bitstrings* is a bijective function mapping $\{0, 1, \ldots, 2^n - 1\}$ to $\{0, 1\}^n$. Let N be a positive integer with $N \leq 2^n$, and let $\mathcal{G}_N$ be the set of all finite cyclic groups G of order N with $G \subset \{0, 1\}^n$. Given an arbitrary finite cyclic group G of order N, we can represent each element of G by a unique n bits string, since $N \leq 2^n$. Thus, every finite cyclic group G of order N is in $\mathcal{G}_N$ in $\binom{2^n}{N}$ distinct representations. Recall that every finite cyclic group G of order N can be isomorphic to the additive group $\mathbb{Z}_N$ based on a generator of G. Thus, we see that, for every pair of a finite cyclic group $G \in \mathcal{G}_N$ and its generator g, there is an encoding function σ into n bitstrings such that $\mathbb{Z}_N$ is isomorphic to G via σ and $\sigma(1) = g$. Conversely, for every encoding function σ into n bitstrings, by defining a binary operation $\circ \colon \sigma(\mathbb{Z}_N) \times \sigma(\mathbb{Z}_N) \to \sigma(\mathbb{Z}_N)$ by

$$\sigma(x) \circ \sigma(y) := \sigma(x + y),$$

the set $\sigma(\mathbb{Z}_N)$ becomes a finite cyclic group in $\mathcal{G}_N$ with the generator $\sigma(1)$ and $\mathbb{Z}_N$ is isomorphic to $\sigma(\mathbb{Z}_N)$ via σ. In this manner, there is a surjective mapping from an encoding function σ into n bitstrings to a pair of a finite cyclic group $G \in \mathcal{G}_N$ and its generator. If we restrict the domain of definition of encoding functions into n bitstrings to $\mathbb{Z}_N$, the mapping becomes bijective.

A *generic algorithm* is a probabilistic oracle Turing machine $\mathcal{A}$ which behaves as follows [27, 19]: Let $n \in \mathbb{N}^+$, and let σ be an encoding function into n bitstrings and N a positive integer with $N \leq 2^n$.

(i) $\mathcal{A}$ takes as input a list $\sigma(x_1), \ldots, \sigma(x_k)$ with $x_1, \ldots, x_k \in \mathbb{Z}_N$, as well as (the binary representations of) N and its prime factorization.

(ii) As $\mathcal{A}$ is executed, it is allowed to make calls to oracles which compute the functions $add \colon \sigma(\mathbb{Z}_N) \times \sigma(\mathbb{Z}_N) \to \sigma(\mathbb{Z}_N)$ and $inv \colon \sigma(\mathbb{Z}_N) \to \sigma(\mathbb{Z}_N)$ with

$$add(\sigma(x), \sigma(y)) = \sigma(x + y) \quad \text{and} \quad inv(\sigma(x)) = \sigma(-x).$$

(iii) Eventually, $\mathcal{A}$ halts and outputs a finite binary string, denoted by

$$\mathcal{A}(N; \sigma(x_1), \ldots, \sigma(x_k)).$$

Consider the following experiment for a polynomial-time generic algorithm $\mathcal{A}$, a parameter n, and a positive integer $N \leq 2^n$:

The discrete logarithm experiment $\mathsf{DLog}_{\mathcal{A}}(n, N)$:

1. *Generate an encoding function σ into n bitstrings uniformly.*
2. *Generate $x \in \mathbb{Z}_N$ uniformly.*
3. *The output of the experiment is defined to be 1 if $\mathcal{A}(N; \sigma(1), \sigma(x)) = x$ and 0 otherwise.*

Note here that $x \in \mathbb{Z}_N$ is the discrete logarithm of $\sigma(x)$ with respect to the generator $\sigma(1)$ in the finite cyclic group $\sigma(\mathbb{Z}_N)$ of order N. Thus in the experiment, given a generator $\sigma(1)$ of a finite cyclic group $\sigma(\mathbb{Z}_N)$ and an element $\sigma(x)$ of $\sigma(\mathbb{Z}_N)$, the generic algorithm $\mathcal{A}$ tries to calculate the discrete logarithm x of $\sigma(x)$ while making calls to oracles which compute the functions add and inv. Shoup [27] showed the following lower bound for the complexity of the discrete logarithm problem in the generic group model.

Theorem 7.1 (Shoup [27]). *There exists $C \in \mathbb{N}^+$ such that, for every generic algorithm $\mathcal{A}$, $n \in \mathbb{N}^+$, and N with $2 \leq N \leq 2^n - 1$,*

$$\Pr[\mathsf{DLog}_{\mathcal{A}}(n, N) = 1] \leq \frac{Cm^2}{p},$$

where p is the largest prime divisor of N and m is the maximum number of the oracle queries among all the computation paths of $\mathcal{A}$. $\qquad\square$

Theorem 7.1 says that any generic algorithm that solves with nonzero constant probability the discrete logarithm problem in finite cyclic groups of order N must perform at least $\Omega(\sqrt{p})$ group operations (i.e., oracle queries).

In what follows, we show that the generic group, i.e, the random encoding function σ into n bitstrings, used in the discrete logarithm problem can be instantiated by a deterministic and computable one while keeping the computational hardness originally proved in the generic group model, as in Theorem 7.1. Before that, we develop the Lebesgue outer measure on families of encoding functions in the next section.

8 Lebesgue outer measure on families of encoding functions

For each $n \in \mathbb{N}^+$, we denote by Encf_n the set of all encoding functions into n bitstrings. Note that $\#\mathsf{Encf}_n = (2^n)!$. A *family of encoding functions* is an infinite sequence $\{\sigma_n\}_{n \in \mathbb{N}^+}$ such that σ_n is an encoding function into n bitstrings for all $n \in \mathbb{N}^+$. A family of encoding functions serves as an instantiation of an infinite sequence of the generic groups, i.e., random encoding functions, over all security parameters. We denote by Encf^∞ the set of all families of encoding functions. Namely,

$$\mathsf{Encf}^\infty := \prod_{k=1}^{\infty} \mathsf{Encf}_k = \mathsf{Encf}_1 \times \mathsf{Encf}_2 \times \mathsf{Encf}_3 \times \cdots\cdots .$$

On the other hand, a *finite family of encoding functions* is a finite sequence $s = (\sigma_1, \ldots, \sigma_n)$ such that σ_k is an encoding function into k bitstrings for all $k = 1, \ldots, n$. Here, n is called the *length* of s and denoted by $|s|$. A finite family of encoding functions is an initial segment (a prefix) of a family of encoding functions. For each $n \in \mathbb{N}$, we denote by Encf^n the set of all finite families of encoding functions of length n. Namely,

$$\mathsf{Encf}^n := \prod_{k=1}^{n} \mathsf{Encf}_k = \mathsf{Encf}_1 \times \cdots \times \mathsf{Encf}_n.$$

Note that $\mathsf{Encf}^0 = \{\lambda\}$ where $\lambda := ()$ is the *empty sequence*. We denote by Encf^* the set of all finite families of encoding functions, i.e., $\mathsf{Encf}^* := \bigcup_{n=0}^{\infty} \mathsf{Encf}^n$. For any sequences $s = (\sigma_1, \ldots, \sigma_n)$ and $t = (\tau_1, \ldots, \tau_m)$ in Encf^*, we say that s is a *prefix* of t if $n \leq m$ and $\sigma_k = \tau_k$ for all $k \leq n$. A subset P of Encf^* is called *prefix-free* if no sequence in P is a prefix of another sequence in P.

In what follows we use the notion of Lebesgue outer measure on Encf^∞, which is defined as follows. For any sequence $s = (\sigma_1, \ldots, \sigma_n) \in \mathsf{Encf}^*$, $I(s)$ is defined as the set of all families $\{\tau_k\}_{k \in \mathbb{N}^+}$ of encoding functions for which $\sigma_k = \tau_k$ for all $k \leq n$, and $|I(s)|$ is defined by

$$|I(s)| := \prod_{k=1}^{n} \frac{1}{\#\mathsf{Encf}_k} = \frac{1}{\#\mathsf{Encf}_1 \times \cdots \times \#\mathsf{Encf}_n}.$$

Note that $I(\lambda) = \mathsf{Encf}^\infty$ and $|I(\lambda)| = 1$. *Lebesgue outer measure $\mathcal{L}$ on Encf^∞* is a function mapping any subset A of Encf^∞ to a non-negative real, and is defined by

$$\mathcal{L}(A) := \inf \sum_{n=1}^{\infty} |I(s_n)|,$$

where the infimum extends over all infinite sequences $s_1, s_2, \ldots \in \mathsf{Encf}^*$ for which $A \subset \bigcup_{n=1}^{\infty} I(s_n)$.

In what follows, we use the properties of $\mathcal{L}$ presented in Proposition 8.1 below. For any subset T of Encf^*, we denote by $[T]^{\prec}$ the set $\bigcup_{s \in T} I(s)$.

Proposition 8.1.

(i) For every prefix-free set $P \subset \mathsf{Encf}^*$,

$$\mathcal{L}([P]^{\prec}) = \sum_{s \in P} |I(s)|.$$

Therefore $\mathcal{L}(\emptyset) = \mathcal{L}([\emptyset]^{\prec}) = 0$ and $\mathcal{L}(\mathsf{Encf}^\infty) = \mathcal{L}([\{\lambda\}]^{\prec}) = 1$.

(ii) $\mathcal{L}(A) \le \mathcal{L}(B)$ for every sets $A \subset B \subset \mathsf{Encf}^\infty$.

(iii) $\mathcal{L}\left(\bigcup_i A_i\right) \le \sum_i \mathcal{L}(A_i)$ for every sequence $\{A_i\}_{i\in\mathbb{N}}$ of subsets of Encf^∞.

(iv) $\mathcal{L}\left(\bigcup_i [P_i]^\prec\right) = \sum_i \mathcal{L}\left([P_i]^\prec\right)$ for every finite or infinite sequence $\{P_i\}_i$ of subsets of Encf^* such that $[P_i]^\prec \cap [P_j]^\prec = \emptyset$ for every $i \ne j$. $\qquad\square$

For any subset S of Encf^*, we say that S is *recursively enumerable* (*r.e.*, for short) if there exists a deterministic Turing machine which on every input $s \in \mathsf{Encf}^*$ halts if and only if $s \in S$. Note here that any sequence in Encf^* is a finite object, which can be represented as a finite binary string, and thus can be manipulated by a Turing machine. Finally, a family $\{\sigma_n\}_{n\in\mathbb{N}+}$ of encoding functions is called *computable* if there exists a deterministic Turing machine which on every input (n,x) with $x \in \{0,1,\dots,2^n-1\}$ halts and outputs $\sigma_n(x)$.

Theorem 8.2 below plays a crucial role in what follows. It is a modification of Lemma 5.4. We can prove this theorem based on the properties of $\mathcal{L}$ in Proposition 8.1, as well as the computability of the mapping $\mathbb{N}^+ \ni n \mapsto \#\mathsf{Encf}_n$.

Theorem 8.2. Let S be an r.e. subset of Encf^*. Suppose that $\mathcal{L}\left([S]^\prec\right) < 1$ and $\mathcal{L}\left([S]^\prec\right)$ is a computable real. Then there exists a computable family of encoding functions which is not in $[S]^\prec$.

Proof. We define $F\colon \mathsf{Encf}^* \to [0,1]$ by $F(t) = \mathcal{L}\left([S]^\prec \cap I(t)\right)$. First, we show that the real-valued function F is computable, i.e., there exists a computable function $f\colon \mathsf{Encf}^* \times \mathbb{N} \to \mathbb{Q}$ such that

$$|F(t) - f(t,k)| < 2^{-k} \tag{14}$$

for all $t \in \mathsf{Encf}^*$ and $k \in \mathbb{N}$.

Let $n \in \mathbb{N}$. Since $\bigcup_{t\in\mathsf{Encf}^n} I(t) = \mathsf{Encf}^\infty$ we have

$$\bigcup_{t\in\mathsf{Encf}^n} [S]^\prec \cap I(t) = [S]^\prec$$

and $\left([S]^\prec \cap I(t)\right) \cap \left([S]^\prec \cap I(t')\right) = \emptyset$ for any distinct $t, t' \in \mathsf{Encf}^n$. Note that, for every $t \in \mathsf{Encf}^*$, there is $S' \subset \mathsf{Encf}^*$ such that $[S]^\prec \cap I(t) = [S']^\prec$.[6] It follows from (iv) of Proposition 8.1 that

$$\sum_{u\in\mathsf{Encf}^n} F(u) = \mathcal{L}\left([S]^\prec\right) \tag{15}$$

for every $n \in \mathbb{N}$.

Since S is an r.e. set, there is a deterministic Turing machine which enumerates S, i.e., there is a deterministic Turing machine which on every input $m \in \mathbb{N}^+$ outputs a finite subset S_m of S, where $S_m \subset S_{m+1}$ for every $m \in \mathbb{N}^+$ and $\bigcup_{m=1}^\infty S_m = S$. Therefore, for each $t \in \mathsf{Encf}^*$, we have $[S_m]^\prec \cap I(t) \subset [S_{m+1}]^\prec \cap I(t)$ for every $m \in \mathbb{N}^+$ and $\bigcup_{m=1}^\infty \left([S_m]^\prec \cap I(t)\right) = [S]^\prec \cap I(t)$. Using (ii) and (iv) of Proposition 8.1 it is easy to show that, for each $t \in \mathsf{Encf}^*$, $\mathcal{L}\left([S_m]^\prec \cap I(t)\right) \le \mathcal{L}\left([S_{m+1}]^\prec \cap I(t)\right)$ for every $m \in \mathbb{N}^+$ and $\lim_{m\to\infty} \mathcal{L}\left([S_m]^\prec \cap I(t)\right) = F(t)$. It follows from (15) that, for each $n \in \mathbb{N}$,

$$\sum_{u\in\mathsf{Encf}^n} \mathcal{L}\left([S_m]^\prec \cap I(u)\right) \le F(t) + \sum_{u\in\mathsf{Encf}^n \text{ and } u\ne t} \mathcal{L}\left([S_m]^\prec \cap I(u)\right) \le \mathcal{L}\left([S]^\prec\right) \tag{16}$$

---

[6]As such S', the set $T \cup \{s \in S \mid t \text{ is a prefix of } s\}$ suffices, where $T = \{t\}$ if there is a prefix $s \in S$ of t and $T = \emptyset$ otherwise.

23

for every $m \in \mathbb{N}^+$ and $t \in \mathsf{Encf}^n$, and

$$\lim_{m \to \infty} \sum_{u \in \mathsf{Encf}^n} \mathcal{L}\left([S_m]^{\prec} \cap I(u)\right) = \mathcal{L}\left([S]^{\prec}\right). \tag{17}$$

Note that, any given finite set $P \subset \mathsf{Encf}^*$, one can compute a finite prefix-free set $Q \subset \mathsf{Encf}^*$ such that $[Q]^{\prec} = [P]^{\prec}$. It follows from (i) of Proposition 8.1 and the computability of the mapping $\mathbb{N}^+ \ni l \mapsto \#\mathsf{Encf}_l$ that, any given $t \in \mathsf{Encf}^*$ and $m \in \mathbb{N}^+$, one can compute the rational $\mathcal{L}\left([S_m]^{\prec} \cap I(t)\right)$. Therefore, any given $n \in \mathbb{N}$ and $m \in \mathbb{N}^+$, one can compute the rational $\sum_{u \in \mathsf{Encf}^n} \mathcal{L}\left([S_m]^{\prec} \cap I(u)\right)$.

Now, since $\mathcal{L}\left([S]^{\prec}\right)$ is a computable real by the assumption, there exists a computable function $g \colon \mathbb{N} \to \mathbb{Q}$ such that

$$\left|\mathcal{L}\left([S]^{\prec}\right) - g(k)\right| < 2^{-k} \tag{18}$$

for all $k \in \mathbb{N}$. It follows from (17) that there exists a computable function $h \colon \mathsf{Encf}^* \times \mathbb{N} \to \mathbb{N}^+$ such that, for every $t \in \mathsf{Encf}^*$ and $k \in \mathbb{N}$,

$$g(k) - 2^{-k} < \sum_{u \in \mathsf{Encf}^{|t|}} \mathcal{L}\left([S_{h(t,k)}]^{\prec} \cap I(u)\right).$$

But, by (16) and (18), the right-hand side is at most

$$F(t) + \sum_{u \in \mathsf{Encf}^{|t|} \text{ and } u \neq t} \mathcal{L}\left([S_{h(t,k)}]^{\prec} \cap I(u)\right) < g(k) + 2^{-k}.$$

Thus we define a function $f \colon \mathsf{Encf}^* \times \mathbb{N} \to \mathbb{Q}$ by

$$f(t,k) = g(k) - \sum_{u \in \mathsf{Encf}^{|t|} \text{ and } u \neq t} \mathcal{L}\left([S_{h(t,k)}]^{\prec} \cap I(u)\right).$$

We then see that the rational-valued function f is computable and (14) holds, as desired.

Next, we construct a computable family $\{\sigma_n\}_{n \in \mathbb{N}^+}$ of encoding functions such that

$$F((\sigma_1, \ldots, \sigma_m)) < \prod_{k=1}^{m} \frac{1}{\#\mathsf{Encf}_k} \tag{19}$$

holds for all $m \in \mathbb{N}$. We do this by the recursive procedure given below. First, since

$$\bigcup_{\tau \in \mathsf{Encf}_{m+1}} I((\tau_1, \ldots, \tau_m, \tau)) = I((\tau_1, \ldots, \tau_m))$$

holds for every $m \in \mathbb{N}$ and $(\tau_1, \ldots, \tau_m) \in \mathsf{Encf}^m$, we note by (iv) of Proposition 8.1 that

$$\sum_{\tau \in \mathsf{Encf}_{m+1}} F((\tau_1, \ldots, \tau_m, \tau)) = F((\tau_1, \ldots, \tau_m)) \tag{20}$$

for every $m \in \mathbb{N}$ and $(\tau_1, \ldots, \tau_m) \in \mathsf{Encf}^m$. Let $s_n = (\sigma_1, \ldots, \sigma_n) \in \mathsf{Encf}^n$ for each $n \in \mathbb{N}$. Then the recursive procedure is given as follows.

Initially, we set $n := 0$ and $s_n := \lambda$. Then, obviously, the property (19) holds for $m = n$, which is precisely the assumption $\mathcal{L}\left([S]^{\prec}\right) < 1$ of the theorem.

For an arbitrary $n \in \mathbb{N}$, assume that we have constructed $s_n = (\sigma_1, \ldots, \sigma_n)$ and (19) holds for $m = n$. It follows from (20) with $m = n$ that

$$F((\sigma_1, \ldots, \sigma_n, \tau_0)) < \prod_{k=1}^{n+1} \frac{1}{\#\mathsf{Encf}_k} \tag{21}$$

for some $\tau_0 \in \mathsf{Encf}_{n+1}$. Since F is a computable real function, by computing the approximation of $F((\sigma_1, \ldots, \sigma_n, \tau))$ with an arbitrary precision for each $\tau \in \mathsf{Encf}_{n+1}$, one can find τ_0 for which (21) holds, and then set $\sigma_{n+1} := \tau_0$ and $s_{n+1} := (\sigma_1, \ldots, \sigma_n, \tau_0)$. It follows that (19) holds for $m = n+1$.

Thus, any given $n \in \mathbb{N}^+$, one can compute σ_n by the above procedure. This implies that the family $\{\sigma_n\}_{n \in \mathbb{N}^+}$ of encoding functions is computable.

Now, assume contrarily that $\{\sigma_n\}_{n \in \mathbb{N}^+} \in [S]^{\prec}$. Then there is $n \in \mathbb{N}$ such that $(\sigma_1, \ldots, \sigma_n) \in S$. It follows that

$$F((\sigma_1, \ldots, \sigma_n)) = \mathcal{L}\left(I((\sigma_1, \ldots, \sigma_n))\right) = \prod_{k=1}^{n} \frac{1}{\#\mathsf{Encf}_k}.$$

However, this contradicts (19) with $m = n$. Hence we have $\{\sigma_n\}_{n \in \mathbb{N}^+} \notin [S]^{\prec}$, and the proof is completed. □

9 Effective hardness and secure instantiation of the generic group

In this section we introduce the notion of *effective hardness* for the discrete logarithm problem, and then show that the generic group used in the problem can be instantiated by a deterministic and computable one while keeping the computational hardness. For that purpose, we first translate Theorem 7.1 into the form well used as a computational hardness assumption for a cryptographic scheme in cryptography.

Consider the following experiment for a polynomial-time generic algorithm $\mathcal{A}$, a parameter n, and an encoding function σ into n bitstrings:

The discrete logarithm experiment $\mathsf{DLog}_{\mathcal{A}}(n, \sigma)$**:**

1. *Generate an n-bit prime p uniformly.*
2. *Generate $x \in \mathbb{Z}_p$ uniformly.*
3. *The output of the experiment is defined to be 1 if $\mathcal{A}(p; \sigma(1), \sigma(x)) = x$ and 0 otherwise.*

In the experiment above, we consider the discrete logarithm problem in the finite cyclic group $\sigma(\mathbb{Z}_p)$ of a *prime* order p. The reason for choosing a prime order is to minimize the probability of the generic algorithm $\mathcal{A}$ solving the discrete logarithm problem. This can be checked from the form of Theorem 7.1.

The hardness of the discrete logarithm problem in the generic group model is then formulated as follows.

Definition 9.1. *We say that the discrete logarithm problem is hard in the generic group model if for all polynomial-time generic algorithms $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,*

$$\frac{1}{\#\mathsf{Encf}_n} \sum_{\sigma \in \mathsf{Encf}_n} \Pr[\mathsf{DLog}_{\mathcal{A}}(n, \sigma) = 1] \leq \frac{1}{n^d}. \tag{22}$$

$\square$

Note that, in the left-hand side of (22), the probability is averaged over all encoding functions into n bitstrings. This results in a *random* encoding function into n bitstrings, i.e., the *generic group.*

In this paper we consider a stronger notion of the hardness of the discrete logarithm problem than that given by Definition 9.1 above. This stronger notion, called the *effective hardness* of the discrete logarithm problem, is defined as follows: We first choose a particular recursive enumeration $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots$ of all polynomial-time generic algorithms. It is easy to show that such an enumeration exists. In fact, the kth polynomial-time generic algorithm $\mathcal{A}_k$ can be chosen as a generic algorithm obtained by executing the kth generic algorithm $\mathcal{M}_k$ in at most $n^k + k$ steps, where n is the length of the input of $\mathcal{M}_k$. We use this specific enumeration as the standard one throughout the rest of this paper.

Definition 9.2. *We say that the discrete logarithm problem is effectively hard in the generic group model if there exists a computable function $f : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, d)$ then*

$$\frac{1}{\#\mathsf{Encf}_n} \sum_{\sigma \in \mathsf{Encf}_n} \Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma) = 1] \leq \frac{1}{n^d}.$$

$\square$

Theorem 9.3. *The discrete logarithm problem is effectively hard in the generic group model.* $\square$

In order to prove Theorem 9.3, we need the following lemma.[7]

Lemma 9.4. *Let $d \geq 4$. Then $2^n \geq n^d$ for all $n \geq d^2$.*

Proof. We first show that

$$2^d \geq d^2 \tag{23}$$

by induction. Obviously, $2^k \geq k^2$ holds for $k = 4$. For an arbitrary $k \geq 4$, assume that $2^k \geq k^2$ holds. Then $2^{k+1} \geq 2k^2 \geq (k+1)^2$, where the second inequality follows from the inequality $\sqrt{2}x \geq x + 1$ for all $x \geq \sqrt{2} + 1$. Thus (23) holds.

Now, we show that

$$2^n \geq n^d \tag{24}$$

holds for all $n \geq d^2$ by induction. First, it follows from (23) that $2^{d^2} \geq (d^2)^d$, which implies that (24) holds for $n = d^2$. For an arbitrary $k \geq d^2$, assume that (24) holds for $n = k$. We note that

---

[7]In order to prove Theorem 9.3, it is suffice to use the inequality $2^n \geq n^d$ which holds for all $n \geq ((d+1)/\ln 2)^{d+1}$ and not for all $n \geq d^2$ as in Lemma 9.4. The former follows immediately from the inequality $e^x \geq x$ which holds for all $x \in \mathbb{R}$. However, we prefer a more "insightful" polynomial lower bound $n \geq d^2$ than the super-exponential lower bound $n \geq ((d+1)/\ln 2)^{d+1}$. See Section 11 for further remarks.

$2^{1/d} - 1 \geq \ln 2/d > 1/(2d) > 1/d^2$, where the first inequality follows from the mean-value theorem. Then, since $k \geq d^2$, we see that $2^{(k+1)/d} \geq 2^{1/d}k \geq k + k/d^2 \geq k+1$. This implies that (24) holds for $n = k+1$. Thus, (24) holds for all $n \geq d^2$. $\qquad\square$

Proof of Theorem 9.3. Let $k \in \mathbb{N}^+$, and consider the kth generic algorithm $\mathcal{A}_k$. Since the number of oracle queries along any computation path of $\mathcal{A}_k$ is bounded to the above by $n^k + k$, it follows from Theorem 7.1 that there exists $C \in \mathbb{N}^+$ such that, for every $n \in \mathbb{N}^+$ and n-bit prime p,

$$\Pr[\mathsf{DLog}_{\mathcal{A}_k}(n,p) = 1] \leq \frac{C(n^k+k)^2}{p} \leq \frac{C(n^k+k)^2}{2^{n-1}}.$$

Therefore, for every $n \geq \max\{k, 2C\}$,

$$\frac{1}{\#\mathsf{Encf}_n} \sum_{\sigma \in \mathsf{Encf}_n} \Pr[\mathsf{DLog}_{\mathcal{A}_k}(n,\sigma) = 1] \leq \frac{n^{2k+1}}{2^n}. \tag{25}$$

Note by Lemma 9.4 that, for each $d \in \mathbb{N}^+$,

$$\frac{n^{2k+1}}{2^n} \leq \frac{1}{n^d} \tag{26}$$

for every $n \geq (2k+d+1)^2$.

Thus we define a function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ by

$$f(k,d) = \max\{(2k+d+1)^2, 2C\}.$$

Then f is computable, and it follows from (25) and (26) that, for all $k, d, n \in \mathbb{N}^+$, if $n \geq f(k,d)$ then

$$\frac{1}{\#\mathsf{Encf}_n} \sum_{\sigma \in \mathsf{Encf}_n} \Pr[\mathsf{DLog}_{\mathcal{A}_k}(n,\sigma) = 1] \leq \frac{1}{n^d}.$$

This completes the proof. $\qquad\square$

The hardness of the discrete logarithm problem in the generic group model given by Definition 9.1 follows immediately from Theorem 9.3.

Corollary 9.5. *The discrete logarithm problem is hard in the generic group model.* $\qquad\square$

We are interested in the instantiation of the generic group in the discrete logarithm problem. Thus, it is convenient to define the hardness of the discrete logarithm problem relative to a specific family of encoding functions.

Definition 9.6. *Let $\{\sigma_n\}_{n \in \mathbb{N}^+}$ be a family of encoding functions. We say that* the discrete logarithm problem is hard relative to $\{\sigma_n\}_{n \in \mathbb{N}^+}$ *if for all polynomial-time generic algorithms $\mathcal{A}$ and all $d \in \mathbb{N}^+$ there exists $N \in \mathbb{N}^+$ such that, for all $n \geq N$,*

$$\Pr[\mathsf{DLog}_{\mathcal{A}}(n,\sigma) = 1] \leq \frac{1}{n^d}.$$

$\qquad\square$

The corresponding effective hardness notion is defined as follows.

Definition 9.7. *Let $\{\sigma_n\}_{n \in \mathbb{N}^+}$ be a family of encoding functions. We say that the discrete logarithm problem is effectively hard relative to $\{\sigma_n\}_{n \in \mathbb{N}^+}$ if there exists a computable function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, d)$ then*

$$\Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma) = 1] \leq \frac{1}{n^d}.$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

In a similar manner to the proof of Theorem 5.3, we can prove the following theorem.

Theorem 9.8 (Main result III). *There exists a computable family of encoding functions relative to which the discrete logarithm problem is effectively hard.*

Proof. First, by Theorem 9.3 there exists a computable function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, d)$ then

$$\frac{1}{\#\mathsf{Encf}_n} \sum_{\sigma \in \mathsf{Encf}_n} \Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma) = 1] \leq \frac{1}{n^d}.$$

It follows from Lemma 4.4 that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, 2d)$ then

$$\#\left\{ \sigma \in \mathsf{Encf}_n \;\middle|\; \Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma) = 1] > \frac{1}{n^d} \right\} < \frac{\#\mathsf{Encf}_n}{n^d}. \tag{27}$$

In order to apply the method of algorithmic randomness, i.e., Theorem 8.2, for each $i, d, n \in \mathbb{N}^+$ we define a subset $[C_{i,d,n}]^{\prec}$ of Encf^∞ as the set of all families $\{\sigma_n\}_{n \in \mathbb{N}^+}$ of encoding functions such that

$$\Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma_n) = 1] > \frac{1}{n^d}. \tag{28}$$

Namely, we define a subset $C_{i,d,n}$ of Encf^* as the set of all finite families $(\sigma_1, \ldots, \sigma_n)$ of encoding functions where only σ_n is required to satisfy the inequality (28). Since $C_{i,d,n}$ is a prefix-free set for every $i, d, n \in \mathbb{N}^+$, it follows from (i) of Proposition 8.1 and (27) that, for each $i, d, n \in \mathbb{N}^+$, if $n \geq f(i, 2d)$ then

$$\mathcal{L}\left([C_{i,d,n}]^{\prec}\right) = \sum_{s \in C_{i,d,n}} |I(s)| < \frac{1}{n^d}. \tag{29}$$

We choose a particular computable bijection

$$\varphi \colon \mathbb{N}^+ \to \{\, (i, d) \mid i \in \mathbb{N}^+ \ \& \ d \geq 2 \,\},$$

and define $(\varphi_1(m), \varphi_2(m)) = \varphi(m)$. We then define a computable function $g \colon \mathbb{N}^+ \to \mathbb{N}^+$ by $g(m) = \{f(\varphi_1(m), 2\varphi_2(m)) + 1\}^{m+1}$. For each $m \in \mathbb{N}^+$, we define a subset C_m of $\{0, 1\}^*$ by

$$C_m = \bigcup_{n=g(m)}^{\infty} C_{\varphi_1(m), \varphi_2(m), n}. \tag{30}$$

It follows from (iii) of Proposition 8.1, (29), and Lemma 4.5 that, for each $m \in \mathbb{N}^+$,

$$\mathcal{L}\left([C_m]^{\prec}\right) \leq \sum_{n=g(m)}^{\infty} \mathcal{L}\left(\left[C_{\varphi_1(m),\varphi_2(m),n}\right]^{\prec}\right) < \sum_{n=g(m)}^{\infty} \frac{1}{n^{\varphi_2(m)}} \leq \frac{2}{g(m)} \leq \frac{1}{2^m}. \tag{31}$$

We then define C by

$$C = \bigcup_{m=1}^{\infty} C_m. \tag{32}$$

Therefore, using (iii) of Proposition 8.1,

$$\mathcal{L}\left([C]^{\prec}\right) \leq \sum_{m=1}^{\infty} \mathcal{L}\left([C_m]^{\prec}\right) < \sum_{m=1}^{\infty} \frac{1}{2^m} = 1. \tag{33}$$

Next we show that C is an r.e. subset of Encf^*. It is easy to see that, given i, d, and n, one can decide the finite subset $C_{i,d,n}$ of Encf^*, since the dyadic rational $\mathrm{Pr}[\mathsf{DLog}_{\mathcal{A}_i}(n,\sigma) = 1]$ is computable, given i, n, and an encoding function σ into n bitstrings. Thus, since φ and g are computable functions, it follows from (30) and (32) that C is an r.e. subset of Encf^*.

We then show that $\mathcal{L}\left([C]^{\prec}\right)$ is a computable real. For each $k \in \mathbb{N}$, we define a finite subset D_k of C by

$$D_k = \bigcup_{m=1}^{k} \bigcup_{n=g(m)}^{g(m)2^k - 1} C_{\varphi_1(m),\varphi_2(m),n}.$$

Given $k \in \mathbb{N}$, one can decides the finite set D_k, since φ and g are computable functions and moreover one can decide the finite set $C_{i,d,n}$, given i, d, and n. Therefore, given $k \in \mathbb{N}$, one can calculate the dyadic rational $\mathcal{L}\left([D_k]^{\prec}\right)$ based on (i) of Proposition 8.1. On the other hand, note that

$$C \setminus D_k \subset \left(\bigcup_{m=1}^{k} \bigcup_{n=g(m)2^k}^{\infty} C_{\varphi_1(m),\varphi_2(m),n}\right) \cup \bigcup_{m=k+1}^{\infty} C_m.$$

Thus, using (ii) and (iii) of Proposition 8.1, (29), Lemma 4.5, and (31) we see that, for each $k \in \mathbb{N}$,

$$\mathcal{L}\left([C \setminus D_k]^{\prec}\right) \leq \sum_{m=1}^{k} \sum_{n=g(m)2^k}^{\infty} \mathcal{L}\left(\left[C_{\varphi_1(m),\varphi_2(m),n}\right]^{\prec}\right) + \sum_{m=k+1}^{\infty} \mathcal{L}\left([C_m]^{\prec}\right)$$

$$< \sum_{m=1}^{k} \frac{2}{g(m)2^k} + \sum_{m=k+1}^{\infty} \frac{1}{2^m} \leq \sum_{m=1}^{k} \frac{1}{2^{m+k}} + \frac{1}{2^k} < \frac{1}{2^{k-1}}.$$

Therefore, since $[C]^{\prec} = [D_{k+1}]^{\prec} \cup [C \setminus D_{k+1}]^{\prec}$, using (ii) and (iii) of Proposition 8.1 we have $\left|\mathcal{L}\left([C]^{\prec}\right) - \mathcal{L}\left([D_{k+1}]^{\prec}\right)\right| \leq \mathcal{L}\left([C \setminus D_{k+1}]^{\prec}\right) \leq 2^{-k}$ for each $k \in \mathbb{N}$. Hence, $\mathcal{L}\left([C]^{\prec}\right)$ is a computable real.

Now, it follows from Theorem 8.2 that there exists a computable family $\{\sigma_n\}_{n \in \mathbb{N}^+}$ of encoding functions which is not in $[C]^{\prec}$. Let $i,d,n \in \mathbb{N}^+$ with $n \geq g(\varphi^{-1}(i,d+1))$. We then define $m = \varphi^{-1}(i,d+1)$, i.e., $\varphi(m) = (i,d+1)$. Since $\{\sigma_n\}_{n \in \mathbb{N}^+} \notin [C]^{\prec}$ and $n \geq g(m)$, it follows from (32) and (30) that $\{\sigma_n\}_{n \in \mathbb{N}^+} \notin \left[C_{\varphi_1(m),\varphi_2(m),n}\right]^{\prec} = [C_{i,d+1,n}]^{\prec}$. Therefore, we see that the family

29

$\{\sigma_n\}_{n \in \mathbb{N}^+}$ of encoding functions satisfies that $\Pr[\mathsf{DLog}_{\mathcal{A}_i}(n, \sigma_n) = 1] \le 1/n^{d+1} < 1/n^d$ for each $n \in \mathbb{N}^+$. Thus, since the mapping $\mathbb{N}^+ \times \mathbb{N}^+ \ni (i, d) \mapsto g(\varphi^{-1}(i, d+1))$ is a computable function, it follows from Definition 9.7 that the discrete logarithm problem is effectively hard relative to $\{\sigma_n\}_{n \in \mathbb{N}^+}$. □

Corollary 9.9. *There exists a computable family of encoding functions relative to which the discrete logarithm problem is hard.*

Proof. The result follows immediately from Theorem 9.8. □

10   The Diffie-Hellman problem

In this section we consider the hardness of the computational Diffie-Hellman (CDH) problem in the generic group model. For the CDH problem we can show the analogues of all the results about the discrete logarithm problem shown in the preceding sections. In this section, in particular we present the analogue of Theorem 9.8 for the CDH problem without proof.

We first recall the analogue of Theorem 7.1 for the CDH problem. We thus consider the following experiment for a polynomial-time generic algorithm $\mathcal{A}$, a parameter n, and a positive integer $N \le 2^n$:

> **The computational Diffie-Hellman experiment $\mathsf{CDH}_{\mathcal{A}}(n, N)$:**
>
> 1. *Generate an encoding function σ into n bitstrings uniformly.*
> 2. *Generate $x \in \mathbb{Z}_N$ uniformly.*
> 3. *Generate $y \in \mathbb{Z}_N$ uniformly.*
> 4. *The output of the experiment is defined to be 1 if $\mathcal{A}(N; \sigma(1), \sigma(x), \sigma(y)) = \sigma(xy)$ and 0 otherwise.*

Shoup [27] showed the following lower bound for the complexity of the CDH problem in the generic group model, which is the analog of Theorem 7.1.

Theorem 10.1 (Shoup [27])**.** *There exists $C \in \mathbb{N}^+$ such that, for every generic algorithm $\mathcal{A}$, $n \in \mathbb{N}^+$, and N with $2 \le N \le 2^n - 1$,*

$$\Pr[\mathsf{CDH}_{\mathcal{A}}(n, N) = 1] \le \frac{Cm^2}{p},$$

where p is the largest prime divisor of N and m is the maximum number of the oracle queries among all the computation paths of $\mathcal{A}$. □

Now, consider the following experiment for a polynomial-time generic algorithm $\mathcal{A}$, a parameter n, and an encoding function σ into n bitstrings:

> **The computational Diffie-Hellman experiment $\mathsf{CDH}_{\mathcal{A}}(n, \sigma)$:**
>
> 1. *Generate an n-bit prime p uniformly.*
> 2. *Generate $x \in \mathbb{Z}_p$ uniformly.*

3. *Generate $y \in \mathbb{Z}_p$ uniformly.*

4. *The output of the experiment is defined to be 1 if $\mathcal{A}(p; \sigma(1), \sigma(x), \sigma(y)) = \sigma(xy)$ and 0 otherwise.*

Then the effective hardness of the CDH problem relative to a specific family of encoding functions is defined as follows.

Definition 10.2. *Let $\{\sigma_n\}_{n \in \mathbb{N}^+}$ be a family of encoding functions. We say that the CDH problem is effectively hard relative to $\{\sigma_n\}_{n \in \mathbb{N}^+}$ if there exists a computable function $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that, for all $i, d, n \in \mathbb{N}^+$, if $n \geq f(i,d)$ then*

$$\Pr[\mathsf{CDH}_{\mathcal{A}_i}(n, \sigma) = 1] \leq \frac{1}{n^d}.$$

$\square$

Based on Theorem 10.1, we can show the following analogue of Theorem 9.8 in the same manner as the proof of Theorem 9.8.

Theorem 10.3 (Main result IV)**.** *There exists a computable family of encoding functions relative to which the CDH problem is effectively hard.* $\square$

11 Polynomial-time effective hardness

In Section 6 we have demonstrated the importance of the effective security notions in modern cryptography. The replacement of the conventional security notions of cryptographic schemes by the corresponding effective security notions results in the replacement of the conventional hardness notions of computational problems, which are used as computational assumptions to prove the security of the cryptographic schemes, by the corresponding effective hardness notions, as we have seen in Section 6 where the RSA assumption in Theorem 6.1 is replaced by the effective RSA assumption in Theorem 6.2. In addition, we have been able to prove the main results given in the previous two section, Theorems 9.8 and 10.3, by converting Shoup's original results about the lower bounds of the complexity, Theorems 7.1 and 10.1, into the form of effective hardness. Thus, the effective hardness notions introduced in the previous two sections are useful and considered to be a natural alternative to the conventional hardness notions of computational problems in modern cryptography.

Ultimately, it would seem more natural to require that the functions $f \colon \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ in Definitions 9.2, 9.7, and 10.2 are *polynomial-time computable* rather than simply computable. We call this type of effective hardness *polynomial-time effective hardness*. In Theorem 9.3 we have shown that the discrete logarithm problem is effectively hard in the generic group model. In the proof of Theorem 9.3, the function f has the form $f(i, d) = \max\{(2i + d + 1)^2, 2C\}$. This is a polynomial-time computable function. Thus, the proof of Theorem 9.3 actually shows that *the discrete logarithm problem is polynomial-time effectively hard in the generic group model.*

Conjecture 3 below is a polynomial-time effective version of Theorem 9.8, which states that the discrete logarithm problem is effectively hard in the standard model for *some* finite cyclic group such that the group operations are polynomial-time computable. In the future, it would be challenging to determine whether Conjecture 3 (or its appropriate modification) holds for some

computational assumption COMP which seems weaker than the hardness of the discrete logarithm problem itself.

Conjecture 4. *Under the assumption* COMP, *there exists a* polynomial-time computable *family of encoding functions (or a* polynomial-time computable *family of families of encoding functions) relative to which the discrete logarithm problem is* polynomial-time effectively *hard.* □

Acknowledgments

References

[1] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, Proceedings of the 1st ACM Conference on Computer and Communications Security, ACM, pp.62–73, 1993.

[2] M. Bellare, A. Boldyreva, and A. Palacio, An uninstantiable random-oracle-model scheme for a hybrid-encryption problem, *Proc.* EUROCRYPT 2004, Lecture Notes in Computer Science, Springer-Verlag, Vol.3027, pp.171–188, 2004.

[3] L. Bienvenu, W. Merkle, and A. Nies, Solovay functions and K-triviality, Proceedings of the 28th Symposium on Theoretical Aspects of Computer Science (STACS 2011), pp.452–463, 2011.

[4] V. Brattka, J. Miller, and A. Nies, "Randomness and differentiability," preprint, 2012.

[5] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *J. ACM*, vol. 51, pp. 557–594, 2004.

[6] G. J. Chaitin, "On the length of programs for computing finite binary sequences," *J. Assoc. Comput. Mach.*, vol. 13, pp. 547–569, 1966.

[7] G. J. Chaitin, "A theory of program size formally identical to information theory," *J. Assoc. Comput. Mach.*, vol. 22, pp. 329–340, 1975.

[8] G. J. Chaitin, *Algorithmic Information Theory.* Cambridge University Press, Cambridge, 1987.

[9] A. W. Dent, Adapting the weaknesses of the random oracle model to the generic group model, *Proc.* ASIACRYPT 2002, Lecture Notes in Computer Science, Springer-Verlag, Vol.2501, pp.100–109, 2002.

[10] R. G. Downey and D. R. Hirschfeldt, *Algorithmic Randomness and Complexity.* Springer-Verlag, New York, 2010.

[11] M. Fischlin, A Lehmann, T. Ristenpart, T. Shrimpton, M. Stam, and S. Tessaro, Random oracles with(out) programmability, *Proc.* ASIACRYPT 2010, Lecture Notes in Computer Science, Springer-Verlag, Vol.6477, pp.303–320, 2010.

[12] O. Goldreich, *Foundations of Cryptography: Volume 1 – Basic Tools*. Cambridge University Press, New York, 2001.

[13] O. Goldreich, *Foundations of Cryptography: Volume 2 – Basic Applications*. Cambridge University Press, New York, 2004.

[14] R. Impagliazzo and S. Rudich, Limits on the provable consequences of one-way permutations, *Proc.* CRYPTO'88, Lecture Notes in Computer Science, Springer-Verlag, Vol.403, pp.8–26, 1990.

[15] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC Press, 2007.

[16] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *Problems Inform. Transmission*, vol. 1, no. 1, pp. 1–7, 1965.

[17] G. Leurent and P. Q. Nguyen, How risky is the random-oracle model? *Proc.* CRYPTO 2009, Lecture Notes in Computer Science, Springer-Verlag, Vol.5677, pp.445–464, 2009.

[18] P. Martin-Löf, "The definition of random sequences," *Information and Control*, vol. 9, pp. 602–619, 1966.

[19] U. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, *Proc.* EUROCRYPT'98, Lecture Notes in Computer Science, Springer-Verlag, Vol.1403, pp.72–84, 1998.

[20] U. Maurer, Abstract models of computation in cryptography, *Proc.* Cryptography and Coding 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol.3796, pp.1–12, 2005.

[21] J. Miller and L. Yu, "On initial segment complexity and degrees of randomness," *Trans. Amer. Math. Soc.*, vol. 360, pp. 3193–3210, 2008.

[22] D. Moriyama, R. Nishimaki and T. Okamoto, *Theory of Public-Key Cryptography*. Industrial and Applied Mathematics Series Vol.2. JSIAM, Kyoritsu Shuppan Co., Ltd., Tokyo, 2011. In Japanese.

[23] A. Nies, *Computability and Randomness*. Oxford University Press, Inc., New York, 2009.

[24] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics*. Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1989.

[25] C.-P. Schnorr, "A unified approach to the definition of a random sequence," *Mathematical Systems Theory*, vol. 5, pp. 246–258, 1971.

[26] C.-P. Schnorr, "Process complexity and effective random tests," *J. Comput. System Sci.*, vol. 7, pp. 376–388, 1973.

[27] V. Shoup, Lower bounds for discrete logarithms and related problems, *Proc.* EUROCRYPT'97, Lecture Notes in Computer Science, Springer-Verlag, Vol.1233, pp.256–266, 1997.

[28] R. J. Solomonoff, "A formal theory of inductive inference. Part I and Part II," *Inform. and Control*, vol. 7, pp. 1–22, 1964; vol. 7, pp. 224–254, 1964.

[29] R. M. Solovay, "Draft of a paper (or series of papers) on Chaitin's work ... done for the most part during the period of Sept.–Dec. 1974," unpublished manuscript, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, May 1975, 215 pp.

[30] K. Weihrauch, *Computable Analysis.* Springer-Verlag, Berlin, 2000.