Journal of Universal Computer Science, vol. 12, no. 6 (2006), 654-671
submitted: 31/5/06, accepted: 23/6/06, appeared: 28/6/06

Randomness and Secrecy - A Brief Introduction

Johannes Blömer
(Paderborn University, Germany
bloemer@uni-paderborn.de)

Abstract: We give a brief introduction to probabilistic encryptions. This serves as an example how randomness plays a pivotal role in cryptographic systems that satisfy advanced security concepts.
Key Words: security, cryptography, randomness
Category: E.3, F.1.2

1 Introduction

The goal of cryptography is to provide techniques for keeping information secret, for determining that information has not been tampered with, and for determining who authored pieces of information. To achieve these goals cryptographers develop methods to encrypt information and to sign information digitally, i.e. to provide an electronic equivalent for a handwritten signature. For a long time cryptography was of interest mainly for spies, military leaders, and diplomats. In recent decades, the application of cryptography has expanded because cryptographic technology is used extensively in today's computing and telecommunications infrastructure. Moreover, cryptography became an exact science on the border between mathematics and computer science. Among the main achievements of cryptography as an exact science are precise security definitions. Once exact security definitions were given it soon turned out that randomness is a necessary ingredient to satisfy strong security notions. For example, if we encrypt information to keep it secret, a weak notion of security says, that given the encrypted information an eavesdropper should not be able to deduce the information. This security can be achieved without randomness. On the other hand, a strong notion of security will require that given only the encrypted information an eavesdropper cannot deduce even partial knowledge about the information. To achieve this kind of security randomness is necessary.

The first who clearly stated that randomness is a key ingredient in a secure cryptographic system was C. Shannon in 1949 in his paper *Communication theory of secrecy systems* [Shannon], in which he introduced among other things the notion of perfect secrecy. Cryptographic system with perfect secrecy provide, well, perfect security, even against an adversary that has unlimited computing power but is not omniscient. However, to achieve perfect secrecy one has to pay a huge price, e.g. in any system with perfect secrecy the secret key has to be as long as the message to be encrypted and a key can only be used to encrypt a single message. It was not until 1984 in a paper entitled *Probabilistic encryption* [Goldwasser/Micali] that Goldwasser and Micali introduced notions of security that achieve the same goals as perfect secrecy if only adversaries with limited computing powers are considered.[1] They also constructed encryption schemes that satisfy these security notions, but that do not suffer from the drawbacks of systems having perfect secrecy. Nevertheless, just like Shannon showed that perfect secrecy is impossible without randomness, Goldwasser and Micali showed that to meet their security requirements an encryption must use randomness. Since the work of Goldwasser and Micali randomness has been recognized as a key component in almost any cryptographic system that provides more than just the most basic kind of security.

In this brief overview we demonstrate the close connection between security and randomness by reviewing the original security definitions of Goldwasser and Micali. We also sketch how to construct encryption schemes that satisfy these security definitions. Even though we cannot provide all the details we try to keep our treatment mathematically rigorous, clearly stating what assumptions we need to show that our constructions satisfy the security definitions of Goldwasser and Micali.

2 Notation

In this section we introduce the basic notation that will be used throughout the rest of the paper. For any two bit strings $x, y \in \{0,1\}^*$ we denote the concatenation of x and y by $x\|y$. Given some finite set S, we denote the uniform distribution on S by $U(S)$. If $S = \{0,1\}^s$ we write U_s for $U(\{0,1\}^s)$. Let D be a distribution on some set S, we write $x \leftarrow D$ if an element $x \in S$ is chosen according to distribution D on set S. By slight abuse of notation, given a finite S and a distribution D on S we call *any* function $f : S \to W$ a *random variable.* Traditionally, this term is reserved for functions with $W = \mathbb{R}$.

For a deterministic algorithm A we denote by $A(x)$ the output of A on input x. In this article we mainly deal with probabilistic polynomial-time algorithms (PPTAs) and probabilistic polynomial-size circuit families (PPSCs). If A is a PPTA or a PPSC we denote by $A(x, r)$ the output of A if the input is x and $r = r_1\|\ldots\|r_l \in \{0,1\}^*$ are the internal random bits of A. We assume that for any input x a PPTA or a PPSC generates a fixed number $l(x)$ of random bits which are uniformly distributed, i.e. distributed according to $U_{l(x)}$. A PPTA or PPSC A together with an input x defines a random variable with domain

[1] Note that this happened almost ten years after the invention of public-key cryptography by Diffie and Hellman and the invention of RSA by Rivest, Shamir, and Adleman.

$\{0,1\}^{l(x)}$ whose range is the set of possible outputs of A. For any input x we denote this random variable simply by $A(x)$. Namely,

$$\Pr(A(x) = y) = \frac{|\{r \in \{0,1\}^{l(x)} | A(x,r) = y\}|}{2^{l(x)}}.$$

According to our notation for random variables, for any PPTA or PPSC A and any input x for A we write $y \leftarrow A(x)$, if y is chosen according to the random variable $A(x)$.

Given some PPTA or PPSC A in addition to the internal random bits of A we may have a second source of randomness, i.e. we may also assume that the input to A is distributed according to some distribution D on all possible inputs of some given bit length n. Then we write

$$\Pr(A(x) = y | x \leftarrow D)$$

for the probability that the output of A is y if the input is chosen according to distribution D. Note that the probability here is taken with respect to the internal random bits of A and with respect to the input distribution D. Also note that for fixed y and fixed family of distributions $\{D_n\}_{n \in \mathbb{N}}$, D_n a distribution on $\{0,1\}^n$, the probabilities $\Pr(A(x) = y | x \leftarrow D)$ define a function $\mathbb{N} \to \mathbb{R}$.

We call a function $g : \mathbb{N} \to \mathbb{R}$ *negligible* if for every polynomial $p : \mathbb{N} \to \mathbb{R}$ there exists some n_0 such that for $n > n_0$,

$$g(n) < \frac{1}{p(n)}.$$

If some function g (e.g. a density function $\Pr(A(x) = y | x \leftarrow D)$) is negligible we simply write $g(n) < \mathrm{negl}(n)$. More generally, if two functions g and h differ only by some negligible function, we write $g(n) = h(n) + \mathrm{negl}(n)$.

3 Encryption Schemes and One-Way Security

Suppose two parties, Alice and Bob, want to communicate over an insecure channel without an eavesdropper Eve being able to understand the messages that Alice and Bob exchange. Because the channel is insecure, Eve can read all messages that Alice and Bob exchange. Hence Alice and Bob need a means to transform their messages such that Eve cannot comprehend the original messages even though she has full access to the transformed messages. Moreover, the transformation must be done in such a way that the legitimate receiver of a message (Alice or Bob) can easily recover the original message. The task of transforming messages from the original format into a format that is incomprehensible to everyone except the intended receiver is achieved by using *encryption schemes*. We distinguish two types of encryption schemes: *symmetric or private-key schemes* and *asymmetric or public-key schemes*.

Definition 1. A symmetric or private-key encryption scheme is a triple (G, E, D) of PPTAs such that:

1. G is called the *key generator*. On input $1^s, s \in \mathbb{N}$, the algorithm G outputs a *key* from $\{0,1\}^*$.

2. The *encryption algorithm* E and the *decryption algorithm* D take as inputs elements from $\{0,1\}^* \times \{0,1\}^*$ and output elements from $\{0,1\}^*$.

3. For every key $k \in \{0,1\}^*$ generated by G and every $m \in \{0,1\}^*$, we have

$$D\big(k, (E(k,m)\big) = m. \tag{1}$$

The parameter s is called the *security parameter* of the scheme.

To communicate with the aid of a symmetric encryption scheme, Alice or Bob first have to run the key generator G to obtain a common key k. They have to share the key k but keep it secret from other persons. Then they exchange messages $m \in \{0,1\}^*$ as follows. The sender, say Alice, encrypts m as $c = E(k,m)$ and sends c to Bob. Upon receiving c, Bob decrypts c as $m' = D(k,c)$. From (1) we see that $m = m'$.

Note that the definition of a symmetric encryption scheme does not say anything about the security of an encryption scheme (G,E,D), i.e. it says nothing about the chances of an eavesdropper Eve to understand the messages that Alice and Bob exchange. For example, a triple of algorithms (G,E,D) with $E(k,m) = m$ and $D(k,m) = m$ for every $k \in \{0,1\}^*$ satisfies Definition 1, although it certainly can not be called a good or secure encryption scheme. Security issues and the role that the security parameter s plays will be dealt with later.

Definition 2. An asymmetric or public-key encryption scheme is a triple of PPTAs (G,E,D) such that:

1. G is called the *key generator*. On input $1^s, s \in \mathbb{N}$, the algorithm G outputs a *key pair* (pk, sk) from $\{0,1\}^* \times \{0,1\}^*$. The first coordinate pk of each pair is called the *public key*, the second coordinate sk is called the *private key*.

2. The *encryption algorithm* E and the *decryption algorithm* D take as inputs elements from $\{0,1\}^* \times \{0,1\}^*$ and output elements from $\{0,1\}^*$.

3. For every output pair $(pk, sk) \in \{0,1\}^* \times \{0,1\}^*$ of G and every $m \in \{0,1\}^*$ we have

$$D\big(sk, (E(pk,m)\big) = m. \tag{2}$$

If Bob wants to send messages to Alice with the aid of an asymmetric encryption scheme, Alice first has to run the key generator G to obtain a pair of keys (pk_A, sk_A). She publishes the key pk_A (hence the name public-key encryption scheme) but keeps sk_A secret. Now Bob encrypts messages m he wants to send to Alice by setting $c = E(pk_A, m)$. Finally, Bob sends c to Alice. To recover the message m from c Alice sets $m' = D(sk_A, c)$. From (2) it is clear that $m' = m$. Note that in general the key pair (pk_A, pk_A) cannot be used by Alice to encrypt messages she wants to send to Bob. If Alice wants to encrypt messages she sends to Bob, Bob first has to run the key generator G to obtain his own key pair (pk_B, sk_B). This explains the name asymmetric encryption schemes. Again, security issues will be dealt with below.

In the sequel we call messages m *plaintexts* and encrypted messages (i.e. bit strings produced by an encryption algorithm E) *ciphertexts*.

Before we consider security issues, let us look at some examples.

Example 1. The symmetric encryption scheme *xor-encryption* is defined as follows:

- On input 1^s, the key generator G outputs a key k chosen uniformly at random from $\{0,1\}^s$, i.e.

$$\Pr\big(G(1^s) = k\big) = 2^{-s} \quad \text{for all } k \in \{0,1\}^s.$$

- On input $(k, m) \in \{0,1\}^s \times \{0,1\}^*$, the encryption algorithm first partitions m into bit strings $m_1, \ldots, m_{l-1} \in \{0,1\}^s$ and $m_l \in \{0,1\}^t$ with $t \leq s$. Then E sets $c_i = m_i \oplus k, i = 1, \ldots, l-1$ and $c_l = m_l \oplus k[1..t]$, where $k[1..t]$ denotes the t-bit prefix of k. The output c of E is the concatenation $c = c_1 \| \cdots \| c_l$ of the bit strings $c_1, \ldots, c_l$.
- The decryption algorithm D works exactly as the encryption algorithm E.

It is clear that the xor-encryption satisfies (1). It is also not hard to see that the three algorithms G, E, D are PPTAs.

Example 2. To define the RSA system we denote by $\mathbb{Z}_N$ the ring of integers modulo $N \in \mathbb{N}$ and by $\mathbb{Z}_N^*$ we denote the multiplicative group of integers relatively prime to $N \in \mathbb{N}$. Furthermore $\phi(N)$ is Euler's totient function. In particular, if $N = pq$ is the product of two distinct primes, then $\phi(N) = (p-1)(q-1)$. The asymmetric encryption or public-key scheme *RSA* is defined as follows:

- On input 1^s, the key generation algorithm G first chooses two s-bit primes p, q uniformly at random and sets $N = p \cdot q$. Then G chooses an element $e \in \mathbb{Z}^*_{\phi(N)}$ also uniformly at random. Finally it determines $d \in Z^*_{\phi(N)}$ such that $e \cdot d = 1 \bmod \phi(N)$. The output of G is the pair (pk, sk) with $pk = (N, e)$ and $sk = d$. Here we write N, e, d as $2s$-bit strings.

- On input $(N, e, m) \in \{0,1\}^{4s} \times \{0,1\}^*$, the encryption algorithm E first partitions m into bit strings $m_1, \ldots, m_{l-1} \in \{0,1\}^{2s-1}$ and $m_l \in \{0,1\}^t$ with $t \leq 2s-1$. Then E interprets each m_i as an element of $\mathbb{Z}_N$ and sets $c_i = m_i^e \bmod N, i = 1, \ldots, l$. This is possible since $N \geq 2^{2s-2}$. The algorithm E interprets the elements c_i as bit strings of length $2s$ (possible since $N < 2^{2s}$) and outputs the concatenation $c = c_1 \| \cdots \| c_l$ of the bit strings $c_1, \ldots, c_l$.
- On input $(d, c) \in \{0,1\}^{2s} \times \{0,1\}^*$, the decryption algorithm D first partitions c into bit strings $c_1, \ldots, c_l \in \{0,1\}^{2s}$. Then D interprets each c_i as an element of $\mathbb{Z}_N$ and sets $m_i = c_i^d \bmod N, i = 1, \ldots, l$. The algorithm D interprets the elements m_i as strings with $2s-1$ bits[2] and outputs the concatenation $m = m_1 \| \cdots \| m_l$.

It follows from elementary number theory that RSA satisfies (2) (see for example [Stinson]). Using algorithms like the extended euclidean algorithm, the square-and-multiply algorithm and prime generating algorithms it can also be shown that the algorithms G, E, D are PPTAs.

Now let us turn to the question what it means for an encryption scheme to be secure. The first idea that comes to mind is the following informal definition of security.

Definition 3 (informal). 1. A symmetric encryption scheme (G, E, D) is secure iff given as input a ciphertext $c = E(k, m)$ no efficient algorithm A can compute the plaintext m.

2. An asymmetric encryption scheme $G, E, D)$ is secure iff given as input a public-key pk and a ciphertext $c = E(pk, m)$ no efficient algorithm A can compute the plaintext m.

However, in several ways this definition requires clarification. What exactly do we mean by saying that an *efficient algorithm* cannot compute plaintexts from ciphertexts (and public keys)? Are we considering worst-case or average-case complexity? It seems that average-case complexity is the right measure. Next, does the algorithms have to be successful for any ciphertext? Clearly, it is bad enough if there is an algorithm that for a non-negligible number of ciphertexts c succeeds in computing the corresponding plaintext. Again, we should be considering probabilistic algorithms, more precisely probabilistic algorithms with a non-negligible success probability in determining the plaintext from a ciphertext (and a public key). These considerations lead to the following definition of *one-way security*.

[2] For the last bit string we have to ignore leading zeros.

Definition 4. A symmetric encryption scheme is one-way secure if for every PPTA A (whose running time is polynomial in the security parameter s) and every polynomial p, the following property holds:

$$\Pr\left(A(1^s, c) = m | c \leftarrow E(k, m), k \leftarrow G(1^s), m \leftarrow U_{p(s)}\right) < \mathrm{negl}(s).$$

That is, no eavesdropper modeled as a PPTA A can with some non-negligible probability compute the correct plaintext from a ciphertext. Here the success probability is with respect to the internal coin tosses of A, a key k chosen according to the distribution defined by the key generator G, and the encryption c of a plaintext m chosen from the uniform distribution of plaintexts whose length is bounded by the polynomial in the security parameter s.[3]

This definition also explains the use of the security parameter s: Given a one-way secure symmetric encryption scheme (G, E, D), the security parameter s can be used to control the security of the scheme: Larger security parameter means better one-way security.

The definition of one-way security is easily extended to asymmetric encryptions schemes, the only difference being that the PPTA A in addition to a ciphertext c also gets the public key being used as an input.

Definition 5. An asymmetric encryption scheme is one-way secure if for every PPTA A (whose running time is polynomial is s) and every polynomial p, the following property holds:

$$\Pr\left(A(1^s, pk, c) = m | c \leftarrow E(pk, m), (pk, sk) \leftarrow G(1^s), m \leftarrow U_{p(s)}\right) < \mathrm{negl}(s).$$

Let us consider whether the xor-encryption and RSA are one-way secure.

Theorem 6. *The xor-encryption (Example 1) is one-way secure.*

Proof. In order to compute the plaintext m correctly from a ciphertext c, any algorithm A that does not get the key k as an input must guess this key correctly. However, for any ciphertext c and any key k there is a plaintext m such that $c = E(k, m)$. From this and the fact that k is chosen uniformly at random from $\{0, 1\}^s$ it easily follows that for all polynomials p and all PPTAs A, we get

$$\Pr\left(A(1^s, c) = m | c \leftarrow E(k, m), k \leftarrow G(1^s), m \leftarrow U_{p(s)}\right) = 2^{-s}.$$

Whether RSA is one-way secure is not known. However, the result immediately follows from the following widely believed assumption(s).

[3] It is not hard to see that a bound on the length of the plaintext is necessary for a definition of this kind to make sense. Similar remarks apply to many of the definitions we state below.

Assumption 1 (uniform version) *For any probabilistic polynomial-time algorithm (PPTA) A, the following holds:*

$$\Pr\left(A(N,e,c) = m | (N,e,d) \leftarrow G(1^s), m \leftarrow U(\mathbb{Z}_N), c = m^e \bmod N\right) < negl(s).$$

(non-uniform version) *For any polynomial-size circuit family (PPSC)* $\{C_s\}_{s\in\mathbb{N}}$*, the following holds*

$$\Pr\left(C_s(N,e,c) = m | (N,e,d) \leftarrow G(1^s), m \leftarrow U(\mathbb{Z}_N), c = m^e \bmod N\right) < negl(s).$$

That is, there is no probabilistic polynomial-time algorithm or probabilistic polynomial-size circuit family that for a random public RSA key (N,e) and for a random element $c \in \mathbb{Z}_N$ *is able to invert with non-negligible success probability the RSA encryption function restricted to bit strings representing elements in* $\mathbb{Z}_N$ *.*

The non-uniform assumption will only be used when we discuss advanced security concepts. The uniform version of the assumption easily implies the one-way security of RSA.

Theorem 7. *Under Assumption 1 the RSA encryption scheme (Example 2) is one-way secure.*

4 Advanced Security Concepts - Semantic Security and Polynomial Indistinguishability

Although one-way security captures important aspects of security it has several weaknesses. We state just two problems.

1. If an encryption scheme (G, E, D) is one-way secure, then it can still be easy to compute the plaintext m from a ciphertext $c = (E, em)$, if the plaintext m is of a special form or if the plaintext m is chosen not according to the uniform distribution but some other distribution.

2. Even if a plaintext m cannot be computed efficiently from a ciphertext $E(e, m)$, still partial information about m may be efficiently computable given the ciphertext c.

To illustrate both problems we look at so-called *stereotyped messages*. Consider the case of a company that secures access to its sensitive customer data by a daily changing password. To distribute this password among its employees a public-key encryption scheme (G, E, D) is used where each employee A has its own public key pk_A. Every morning a message m of the form *Today's password for our customer data is xxxx.* is send to every employee A, always encrypted with the employee's public key pk_A. Messages like m are called stereotyped since they consist of a never changing part ("Today's password for our customer data is") and a part that is variable (the actual password).

Using a scheme that is only known to be one-way secure this method to distribute a daily password may not be secure. First, one-way security does not guarantee that messages of the form used in this application are difficult to compute given only the ciphertexts $E(pk_A, m)$. Second, even if in the encryption scheme (G, E, D) used by the company it is difficult to compute the complete message m from the ciphertexts $E(pk_A, m)$ this does not rule out the possibility that parts of m given $E(pk_A, m)$ can be computed efficiently. If the part that is easy to determine happens to be the variable part containing the daily password, then the method is completely insecure and useless. This threat is real. For example, Coppersmith [Coppersmith] describes attacks on stereotyped messages in the RSA encryption scheme.[4]

Hence one-way security is only a first step in defining security notions for encryption schemes. It must be superseded by stricter notions of security. In 1984 in a seminal paper Goldwasser and Micali [Goldwasser/Micali] defined two different stricter notions of security and described encryption schemes that meet these security requirements. The first security notion is called *semantic security*. Informally it can be stated as follows.

Definition 8 (informal). An encryption scheme (G, E, D) is semantically secure if for every probability distribution on plaintexts, whatever a PPTA A can compute about a plaintext given a ciphertext, can also be computed by a PPTA A' without the ciphertext. For a public-key encryption scheme, both A and A' also get the public key being used.

It should be clear that semantic security meets both objections raised against one-way security. The second security notion is called *polynomial indistinguishability*. Informally it reads as follows.

Definition 9 (informal). An encryption scheme (G, E, D) is polynomial indistinguishable, if no PPTA A can generate two plaintexts m_0, m_1 and then correctly distinguish between encryptions of m_0 and m_1 with probability significantly greater than $\frac{1}{2}$. For a public-key encryption scheme, A also gets the public key being used.

At first sight it may not be clear that polynomial indistinguishability also meets the two objections against one-way security. However, it turns out that semantic security and polynomial indistinguishability (if appropriately formalized) are in fact equivalent notions of security. Intuitively, this can be seen as follows. Semantic security states that basically nothing about the plaintext can be computed

[4] There are additional security problems with this approach since an adversary gets not just one ciphertext, but many encryptions $c_A = E(pk_A, m)$) of a single plaintext.

from the ciphertext. But then one should not be able to distinguish ciphertexts of two different plaintexts, which is polynomial indistinguishability. If, on the other hand, one is not able to distinguish the ciphertexts of two plaintexts (i.e. polynomial indistinguishability), one cannot learn anything about a plaintext given a ciphertext, which is semantic security. For a formal proof that semantic security and polynomial indistinguishability are equivalent see [Goldwasser/Micali] (polynomial indistinguishability implies semantic security) and [MRS] (for semantic security implies polynomial indistinguishability).

Since polynomial indistinguishability is somewhat easier to deal with, in the sequel we will concentrate on this notion. First we will give a formal definition of polynomial indistinguishability. Then, in the next section, we will see how to construct encryption schemes that are polynomially indistinguishable. As it turns out, neither RSA nor the xor-encryption is polynomially indistinguishable. However, we will see how we can construct a polynomially indistinguishable encryption combining RSA with randomization.

We now formalize the notion of polynomial indistinguishability. To simplify our exposition we will work with families of probabilistic polynomial-size circuits (PPSCs) rather than with algorithms, i.e. we switch to a non-uniform model of computation. Using circuits rather than algorithms in our definition of security actually strengthens the security requirements (every algorithm can be simulated by circuits but not vice versa). However, we will also be forced to use stronger complexity-theoretic assumptions like the non-uniform version of Assumption 1 to show that the constructions presented in the next section satisfy these security requirements.

Definition 10. 1. A symmetric encryption scheme is called polynomially indistinguishable if for every PPSC $\{C_s\}_{s\in\mathbb{N}}$, every polynomial p , every $m_0, m_1 \in \{0,1\}^{p(s)}$, $|m_0| = |m_1|$, we have

$$\Pr\big(C_s(E(k, m_b)) = b | k \leftarrow G(1^s), b \leftarrow U_1\big) < \frac{1}{2} + \mathrm{negl}(s).$$

2. An asymmetric encryption scheme is called polynomially indistinguishable if for every PPSC $\{C_s\}_{s\in\mathbb{N}}$, every polynomial p, and every $m_0, m_1 \in \{0,1\}^{p(s)}$, $|m_0| = |m_1|$, we have

$$\Pr\big(C_s(pk, E(pk, m_b)) = b | (pk, sk) \leftarrow G(1^s), b \leftarrow U_1\big) < \frac{1}{2} + \mathrm{negl}(s).$$

Remark. 1. Given an encryption of either m_0 or m_1 an algorithm or circuit that simply guesses which message was encrypted has success probability $\frac{1}{2}$. Hence, the definition states that no polynomial-size circuit can do significantly better than just guessing.

2. In the informal definition of polynomial indistinguishability we only required that no algorithm can distinguish the encryptions of two messages which it must be able to generate in polynomial time. In our formal definition we require that this is true for any two messages. This only strengthens the definition. It also simplifies the exposition.

3. The restriction to messages m_0, m_1 whose length is bounded by some polynomial has technical reasons which the reader may ignore.

It is not hard to see that neither the xor-encryption nor RSA is polynomially indistinguishable. For RSA this is easily seen as follows. Let m_0, m_1 be arbitrary plaintexts and let $c = E((N, e), m_b)$ be an encryption of one of these plaintexts. Since the circuit C_s in addition to c also gets the public key as input, C_s may simply compute the encryptions of m_0 and m_1 and compare them with c. This way the circuit C_s can distinguish between encryptions of m_0 and m_1 with probability 1.

Actually, the same argument can be used to show that any asymmetric encryption scheme (G, E, D) with a *deterministic* encryption algorithm E cannot be polynomially indistinguishable. Hence, E must be randomized, i.e. for any plaintext m there exists a whole set or cloud of possible encryptions. On the other hand, the set of possible encryptions for two distinct plaintexts m_0, m_1 must be disjoint. Otherwise the decryption algorithm D cannot decrypt correctly. At first it seems impossible to meet these requirements simultaneously. For adversaries whose resources are unbounded, it is in fact impossible to do so. However, in the next section we will see that in the world of polynomially bounded adversaries one can reconcile these requirements.

A polynomially indistinguishable symmetric encryption scheme need not employ a randomized encryption function. Nevertheless, the xor-encryption as defined in Example 1 is not polynomially indistinguishable. This is easily seen as follows. Let s be the security parameter. Set $m_0 = 0^{2s}$ and $m_1 = 0^s \| 1^s$. Finally let $c = E(k, m_b)$ be the encryption of m_0 or m_1. To determine whether $b = 0$ or $b = 1$, the circuit writes $c = c_0 \| c_1$ with $c_0, c_1 \in \{0, 1\}^s$. If $c_0 = c_1$ then c is the encryption of m_0, otherwise c is the encryption of m_1. Again, we have a circuit that distinguishes encryptions of two different messages with probability 1.

In the case of the xor-encryption the problem is easily albeit expensively fixed. If the key length is s, then we use it only to encrypt plaintexts of length s. To encrypt longer plaintexts m, we partition the plaintext into strings $m_1, \ldots, m_l$ of length at most s. For each string we choose a separate key k_i in $\{0, 1\}^s$ uniformly and independently from all other keys. Then m is encrypted to $m_1 \oplus k_1 \| \ldots \| m_l \oplus k_l$. This scheme is the so-called *one-time-pad*. It is polynomially indistinguishable. This follows immediately from the fact that the modified scheme has *perfect secrecy* as defined originally by C. Shannon. I.e., the a priori probability of a plaintext m (according to some distribution D) is the same as the a posteriori probability of plaintext m given a ciphertext c (see for example [Stinson]).

In many ways, the notions of semantic security and polynomial indistinguishability can be seen as analogs of perfect secrecy when the resources of an adversary are limited to polynomial time.

5 Probabilistic Encryption Schemes

In this section we show how to construct an encryption scheme that is polynomially indistinguishable. As we have just seen, the RSA encryption scheme does not provide polynomial indistinguishability. Nevertheless RSA will play an important role in our construction. However, as explained above RSA must be combined with techniques to randomize encryptions. The security of our construction is based on Assumption 1 in its non-uniform version.

For every positive integer $m \in \mathbb{N}$ or any element $m \in \mathbb{Z}_N$ for some $N \in \mathbb{N}$, we denote by $\mathrm{LSB}_l(m)$ the l least significant bits of m. Furthermore, $\mathrm{LSB}(m) := \mathrm{LSB}_1(m)$. The following theorem was shown by Alexis, Chor, Goldreich, and Schnorr [ACGS].

Theorem 11. *If Assumption 1 is true in its non-uniform version, then for any PPSC $\{C_s\}_{s\in\mathbb{N}}$,*

$$\Pr\big(C_s(N,e,c) = LSB(m) | (N,e,d) \leftarrow G(1^s), m \leftarrow U(\mathbb{Z}_N), c = m^e \bmod N\big) < \frac{1}{2} + negl(s),$$

where G is the key generator of the RSA encryption scheme (Example 2). Hence, in the RSA encryption scheme no PPSC can predict with probability significantly greater than $1/2$ the least significant bit of a plaintext $m \in \mathbb{Z}_N$ given only the public key e, N and the encryption $c = m^e \bmod N$ of m. Here the probability is over the internal random bits of C_s, a random public RSA key (N, e), and the encryption of a random plaintext in $\mathbb{Z}_N$.

Clearly, a uniform version of the above theorem is true as well, i.e. if we restrict ourselves to PPTAs then we only need the uniform version of Assumption 1.

Now we can describe our RSA-based polynomially indistinguishable encryption scheme, called *1-bit-probabilistic-RSA*.

Definition 12. The asymmetric encryption scheme *1-bit-probabilistic-RSA* is defined as follows:

- The key generation algorithm is as in the RSA encryption scheme.
- On input $(N, e, m) \in \{0,1\}^{4s} \times \{0,1\}^*, m = m_1 \dots m_l$ with $m_i \in \{0,1\}$, for each message bit m_i, the encryption algorithm E first chooses an element r_i

uniformly and independently at random from $\mathbb{Z}_N$. Then E computes $c_i = r_i^e \bmod N$ and sets $b_i = \mathrm{LSB}(r_i)$. The encryption of plaintext m is given by $c = c_1\|m_1 \oplus b_1\| \dots \|c_l\|m_l \oplus b_l$. Here we interpret the c_i's as bit strings of length $2s$.

– On input $c = c_1\|b_1'\| \dots \|c_l\|b_l'$ with $b_i' \in \{0,1\}, c_i \in \{0,1\}^{2s}$, the decryption algorithm interprets the bit strings c_i as elements of $\mathbb{Z}_N$ and computes $r_i = c_i^d \bmod N$ using the secret key d. Then D sets $m_i := b_i' \oplus \mathrm{LSB}(r_i), i = 1, \dots, l$. The output of D is $m = m_1\| \dots \|m_l$.

It is easily seen that this describes a correct encryption scheme, i.e. we have $D(d, E(e, N, m)) = m$ for all $m \in \{0,1\}^*$. The following theorem shows that the 1-bit-probabilistic-RSA scheme is polynomially indistinguishable.

Theorem 13. *Under Assumption 1 (non-uniform version) the 1-bit-probabilistic-RSA encryption scheme is polynomially indistinguishable.*

Proof sketch. We show that if the 1-bit-probabilistic-RSA scheme is not polynomially indistinguishable then we can construct a polynomial-size circuit family $\{C_s\}_{s\in\mathbb{N}}$ that violates Theorem 11. This will prove our theorem. The proof proceeds in two stages.

1. It is shown that if the 1-bit-probabilistic-RSA scheme is not polynomially indistinguishable then there is a polynomial-size circuit family $\{C_s'\}_{s\in\mathbb{N}}$ with the following property:

$$\Pr\big(C_s'(N, e, c, b \oplus \mathrm{LSB}(r)) = b | (N, e, d) \leftarrow G(1^s), b \leftarrow U_1, r \leftarrow U(\mathbb{Z}_N), \\ c = r^e \bmod N\big) > \frac{1}{2} + \mathrm{negl}(s), \quad (3)$$

   where G is the RSA key generator (Example 2).

   That is, the circuits C_s' can distinguish the encryptions of 1 bit messages with probability significantly greater than $\frac{1}{2}$.

2. Given circuit family $\{C_s'\}_{s\in\mathbb{N}}$, a circuit family $\{C_s\}_{s\in\mathbb{N}}$ that violates Theorem 11 is constructed.

1. is shown using the so-called *hybrid technique.* We refer to [Goldreich 2004] for a rigorous proof. We only mention that it is this step in the proof that necessitates the use of PPSCs rather than PPTAs. Roughly this can be seen as follows. If the assumptions of the theorem are not correct then for infinitely many $s \in \mathbb{N}$, there exist pairs (m_0, m_1) of plaintexts such that one can distinguish efficiently and with probability significantly greater than $\frac{1}{2}$ between encryptions of m_0, m_1. Then an algorithm satisfying (3) needs as an additional input the two plaintexts

m_0, m_1. However, since these plaintexts depend on s, this raises a serious problem. To overcome this problem we use circuits $\{C'_s\}$ instead of an algorithm. Then for each s, the plaintexts m_0, m_1 are hardwired into the circuit C'_s. This need to switch to a non-uniform model of computations remains even if we only want to show that 1-bit-probabilistic-RSA is polynomially indistinguishable in a uniform sense. The only way to avoid non-uniform models and assumptions is to use a technically more elaborate definition of polynomial indistinguishability.

To prove 2., based on circuits C'_s we construct circuits C_s as follows

input: (N, e, c) where $(N, e, d) \leftarrow G(1^s), c = m^e \bmod N, m \leftarrow U(\mathbb{Z}_N)$.

output: Bit b (a guess for $\mathrm{LSB}(m)$).

1. Choose $\hat{b} \leftarrow U_1$.

2. Simulate C'_s with input $(N, e, c, \hat{b})$.

3. If $C'_s(N, e, c, \hat{b}) = \hat{b}$, output $b := 0$, otherwise output $b := 1$.

To analyze the success probability of circuits C_s, first note that for any $m \in \mathbb{Z}_N$, if $\hat{b} \leftarrow U_1$ then $\tilde{b} = \hat{b} \oplus \mathrm{LSB}(m) \leftarrow U_1$ as well. Using this observation together with our assumption on the success probability of C'_s, we get

$$
\begin{aligned}
&\Pr(C_s(N, e, c) = \mathrm{LSB}(m)) \\
&\qquad = \Pr\left(C'_s(N, e, c, \hat{b}) = \hat{b} \wedge \mathrm{LSB}(m) = 0 | \hat{b} \leftarrow U_1\right) + \\
&\qquad\quad \Pr\left(C'_s(N, e, c, \hat{b}) = \neg\hat{b} \wedge \mathrm{LSB}(m) = 1 | \hat{b} \leftarrow U_1\right) \\
&\qquad = \Pr\left(C'_s(N, e, c, \tilde{b} \oplus \mathrm{LSB}(m)) = \tilde{b} \wedge \mathrm{LSB}(m) = 0 | \hat{b} \leftarrow U_1\right) + \\
&\qquad\quad \Pr\left(C'_s(N, e, c, \tilde{b} \oplus \mathrm{LSB}(m)) = \tilde{b} \wedge \mathrm{LSB}(m) = 1 | \hat{b} \leftarrow U_1\right) \\
&\qquad = \Pr\left(C'_s(N, e, c, b \oplus \mathrm{LSB}(m)) = b | b \leftarrow U_1\right) \\
&\qquad > \frac{1}{2} + \mathrm{negl}(s).
\end{aligned}
$$

Here, all probabilities are also conditioned on $(N, e, d) \leftarrow G(1^s), m \leftarrow U(\mathbb{Z}_N), c = m^e \bmod N$. This concludes the proof sketch.

Remark. We have based our construction of a polynomially indistinguishable encryption scheme on the assumption that the RSA encryption function is a *trapdoor function* with the LSB as a *hardcore bit*. However, the construction underlying the 1-bit-probabilistic-RSA scheme can be carried out with any trapdoor function f and any hardcore bit $b(x)$ for f. For precise definitions of trapdoor functions and hardcore bits, we refer to [Goldreich 2001].

The 1-bit-probabilistic-RSA scheme achieves polynomial indistinguishability but it does so quite inefficiently in terms of its message expansion, i.e. if security parameter s is used, then every single bit of a message is encrypted by $2s+1$ bits in the ciphertext. Since one has to choose s quite large to guarantee security, in practice this will not be tolerable. Fortunately, there are more efficient techniques to design encryption schemes that are polynomially indistinguishable. Like the 1-bit-probabilistic-RSA encryption scheme these techniques combine one-way secure encryption with randomization.

1. One can use an extension of the 1-bit-probabilistic-RSA scheme in which blocks B of l message bits are encrypted by $(c, B \oplus \mathrm{LSB}_l(r))$, where $r \leftarrow U(\mathbb{Z}_N)$ and $c = r^e \bmod N$. To prove the polynomial indistiguishabilty of this scheme (l-bit-probabilistic-RSA) one needs to assume that there is no algorithm or circuit family that can predict $\mathrm{LSB}_l(r)$ given only the public RSA key (N, e) and the value $c = r^e \bmod N$. If $l = O\big(\log(N)\big)$, this assumption is known to follow from Assumption 1. For larger values of l, this is not known to be true.
2. One can also use cryptographically secure *pseudorandom generators* to design polynomially indistinguishable encryption schemes. One way to construct cryptographically secure pseudorandom generators is based on Assumption 1 and on Theorem 11. But one can use any trapdoor function f and a hardcore bit for f as well. This construction was originally proposed in [Blum/Goldwasser].
3. Finally polynomially indistinguishable encryption schemes can be constructed by combining one-way secure encryption schemes with cryptographically secure *hash functions*.

For an exhaustive treatment of these constructions we refer to [Goldreich 2004].

6 Beyond Polynomial Indistinguishability and Semantic Security

So far we have considered adversaries whose goal is to gain information about a plaintext given only the ciphertext and publicly available information like the public key in an asymmetric encryption scheme. However, adversaries may have different goals and/or may be more powerful.

Consider a scenario in which an adversary may know that a ciphertext c is the encryption of some contract proposal exchanged between some of his business competitors. Then his goal may be to modify the ciphertext in such a way that certain contract clauses are modified significantly. Clearly, one way for the adversary to achieve his goal is to intercept the ciphertext c, to try to decrypt c

to obtain m, to modify m in the desired way and finally to encrypt the modified plaintext. For a semantically secure encryption scheme such an attempt must fail (with overwhelming probability). However, there may be ways to generate ciphertexts of related messages without decrypting a single ciphertext. A scheme in which this is impossible is called *non-malleable*, a notion initially introduced in [DDN].

Definition 14 (informal). An encryption scheme (G, E, D) is called non-malleable if there is no PPTA A that given a ciphertext c can generate a different ciphertext c' such that the respective plaintexts m and m' are related in a known fashion.

It can be shown that non-malleability is a stronger security notion than semantic security or polynomial indistinguishability, i.e. any scheme that is non-malleable is also semantically secure and polynomially indistinguishable. Needless to say that in any construction of non-malleable schemes randomness plays a crucial role. Below we will mention briefly how to construct non-malleable encryption schemes.

So far in our discussion of security concepts we have restricted ourselves to so-called *passive adversaries*, i.e. adversaries that only get a ciphertext (or a pair of ciphertexts) and try to gain information about the corresponding plaintext (or pair of plaintexts). However, sometimes it is more realistic to consider more powerful adversaries. One adversarial model that has proved to be very useful are so-called *adaptively-chosen-ciphertext attacks (CCA)* introduced in [Naor/Yung] and [Rackoff/Simon]. We will explain this notion only in connection with polynomial indistinguishability.

As before, given the encryption $c = E(pk, m_b), b \in \{0, 1\}$, of one of two plaintexts m_0 and m_1 the adversary has to determine bit b, i.e. has to determine whether c is an encryption of m_0 or of m_1. This time, however, after receiving the challenge c, the adversary is allowed to ask for the decryptions of (polynomially) many ciphertexts $c_1, \ldots, c_l$. The c_i's may depend on c and may be adaptively chosen. That is c_{i+1} is chosen after the adversary has been told the plaintexts corresponding to $c_1, \ldots, c_i$. The only restriction is that the adversary is not allowed to ask for the decryption of c itself. After having been told the plaintexts corresponding to the adaptively chosen ciphertexts c_i the adversary announces his guess $\tilde{b}$ for the bit b. As before the success probability of the adversary is the probability that $\tilde{b} = b$. We call an encryption scheme (G, E, D) *polynomially indistinguishable against CCAs* if no probabilistic polynomial time CCA against the scheme has a success probability that is significantly greater than $1/2$.

One easily sees that the 1-bit-probabilistic-RSA is not polynomial indistinguishable against CCA. In fact, given the encryption $c = c_1 \| m_1 \oplus \mathrm{LSB}(r_1) \| \ldots \| c_l \| m_l \oplus \mathrm{LSB}(r_l)$ of some plaintext $m = m_1 \| \ldots \| m_l$, an adversary mounting an CCA may simply ask for the decryption m' of $c' = c_1 \| m_1 \oplus \mathrm{LSB}(r_1) \oplus$

$z_1 \| \ldots \| c_l \| m_l \oplus \mathrm{LSB}(r_l) \oplus z_l$ for some arbitrary non-zero bit string $z_1 \| \ldots \| z_l$. Given m', it is straightforward to determine the plaintext m itself.

To defend against CCAs, Bellare and Rogoway [Bellare/Rogaway] introduced *plaintext-aware encryption schemes*.

Definition 15 (informal). An encryption scheme (G, E, D) is called plaintext-aware if there is no PPTA A that with non-negligible probability can construct a valid ciphertext without knowing the corresponding plaintext.

In the so-called *random oracle model* it is known how to construct plaintext-aware encryption schemes. We are not going to define the random oracle model or describe the constructions for plaintext-aware encryption schemes. But as the term random oracle model suggests, randomness again plays an important role. One can also show, that an encryption scheme that is semantically secure and is plaintext-aware is non-malleable [BDPP, DDN]. One important practical scheme that combines plaintext-awareness and semantic security is the RSA-OAEP (=RSA-optimal-asymmetric-encryption-padding), employed for example by RSA Inc. in its standard PKCS#1 v2.0.

So far we have seen how randomness plays a vital role in constructing encryption schemes that satisfy advanced security concepts like semantic security and polynomial indistinguishability. However, randomness is equally important in the construction of other cryptographic primitives. *Digital signatures* that satisfy more than the most basic security requirements must use randomness. *Identification protocols* must employ randomness if they want to satisfy even the most basic security requirements about identification. Hence it is safe to say that security without randomness is impossible.

The following is by no means an exhaustive bibliography, for this we refer to [Goldreich 2001, Goldreich 2004, MOV].

References

[ACGS] Alexi, W., Chor, B., Goldreich, O., Schnorr, C.-P.: "RSA and Rabin functions: certain parts are as hard as the whole", SIAM Journal on Computing, Vol. 17(2), 1988, pp. 194–209.

[BDPP] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P., "Relations among notions of security for public-key encryption schemes", Proc. Crypto'98, Lecture Notes in Computer Science, Vol. 1462, 1999, pp. 26–45.

[Bellare/Rogaway] Bellare, M., Rogaway, P.: "Optimal Asymmetric Encryption – How to encrypt with RSA", Proc. Eurocrypt'94, Lecture Notes in Computer Science, Vol. 950, 1995, pp. 92–111.

[Blum/Goldwasser] Blum, M., Goldwasser, S.:"An efficient probabilistic public-key encryption scheme which hides all partial information", Proc. Crypto'84, Lecture Notes in Computer Science, Vol. 196, 1985, pp. 289–302.

[Coppersmith] Coppersmith, D. :"Small solutions to polynomial equations and low exponent RSA vulnerabilities", Journal of Cryptology, Vol. 10(4), 1997, pp. 223–260.

[DDN] Dolev, D., Dwork, C., Naor, M.:"Non-malleable cryptography", Proceedings of STOC'91, 1991, pp. 542–552.

[Goldreich 2001] Goldreich, O.: "Foundations of cryptography – basic tools", Cambridge University Press, 2001.

[Goldreich 2004] Goldreich, O.: "Foundations of cryptography – basic applications", Cambridge University Press, 2004.

[Goldwasser/Micali] Goldwasser, D., Micali, S.: "Probabilistic encryption", Journal of Computer and System Sciences, Vol. 28(2), 1984, pp. 270–299.

[MOV] Menezes, A.J., van Oorschot, P.C., Vanstone,S.A.:"Handbook of applied cryptography", CRC Press, 1996.

[MRS] Micali, S., Rackoff, C., Sloan, B.: "The notion of security for probabilistic cryptosystems", SIAM Journal on Computing, Vol. 17(2), 1988, pp. 412–426.

[Naor/Yung] Naor, M., Yung, M.: "Public-key cryptosystems provably secure against chosen ciphertext attacks", Proceedings of STOC'90, 1990, pp. 427–437.

[Rackoff/Simon] Rackoff, C., Simon, D.R.: "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", Proc. Crypto'91, Lecture Notes in Computer Science, Vol. 576, 1992, pp. 433–444.

[Shannon] Shannon, C.: "Communication theory of secrecy systems", Bell System Technical Journal, Vol. 28, 1949, pp. 656–715.

[Stinson] Stinson, D.R.: "Cryptography - Theory and practice", 2nd edition, Chapman & Hall,2002.