Part II

Basic Utilities

Chapter 5

Encryption Schemes

Upto the 1970's, Cryptography was understood as the art of building encryption schemes. Since then, other tasks such as message authentication have been recorgnized as at least as central to Cryptography. Yet, the construction of encryption schemes remains, and is likely to remain, a central enterprise of Cryptography.

In this chapter we review the well-known notions of private-key and public-key encryption schemes. More importantly, we define what is meant by saying that such schemes are secure. We then turn to some basic constructions. In particular, we show that the widely used construction of a "stream cipher" yields a secure (private-key) encryption, provided that the "key sequence" is generated using a pseudorandom generator. We actually advocate an alternative construction which uses a pseudorandom function. We then turn to public-key encryption schemes and present constructions based on any trapdoor one-way permutation. Finally, we discuss "dynamic" notions of security such as robustness against chosen ciphertext attacks and non-malleability.

> Author's Note: *Currently the write-up contains only a very rough draft for the first 2.5 sections of this chapter.*

5.1 The Basic Setting

Loosely speaking, encryption schemes are supposed to enable private communication between parties which communicate over an insecure channel. Thus, the basic setting consists of a *sender*, a *receiver*, and an *insecure channel* which may be tapped by an *adversary*. The goal is to allow the sender to transfer information to the receiver, over the insecure channel, without letting the adversary figure out this information. Thus, we distinguish between the actual (secret) information which the receiver wishes to transmit and the messages sent over the insecure communication channel. The former is called the *plaintext*, whereas the latter is called the *ciphertext*. Clearly, the ciphertext must differ from the plaintext or else the adversary can easily obtain the plaintext by tapping the channel. Thus, the sender must transform the plaintext into a ciphertext so that the receiver can retreive the plaintext from the ciphertext, but the adversary cannot do so. Clearly, something must distinguish the receiver (who is able to retreive the plaintext from the corresponding ciphertext) from the adversary (who cannot do so). Specifically, the receiver know something which the adversary does not know. This thing is called a *key*.

An encryption scheme consists of a method of transforming plaintexts to ciphertexts and vice versa, using adequate keys. These keys are essential to the ability to effect these transformations. We stress that the encryption scheme itself (i.e., the encryption/decryption algorithms) may be known to the adversary, and its security relies on the assumption that the adversary does not know the keys. Formally, we need to consider a third algorithm; namely, a probabilistic algorithm used to generate keys. This algorithm must be probabilistic (or else, by invoking it the adversary obtains the very same key used by the receiver).

5.1.1 Overview

In accordance with the above, an encryption scheme consists of three algorithms. These algorithms are public (i.e., known to all parties). The obvious algorithms are the *encryption algorithm*, which transforms plaintexts to ciphertexts, and the *decryption algorithm*, which transforms ciphertexts to plaintexts. By the discussion above, it is clear that the decription algorithm must employ a *key* which is known to the receiver but is not known to the adversary. This key is generated using a third algorithm, called the *key generator*. Furthermore, it is not hard to see that the encryption process must also depend on the key (or else messages sent to one party can be read by a different party who is also a potential receiver). Thus, the key-generation algorithm is used to produce a pair of (related) keys, one for encryption and one for decryption. The encryption algorithm, given an encryption key and a plaintext, produces a plaintext which when fed to the decryption algorithm, with the corresponding decryption key, returns the original plaintext. We stress that knowledge of the decryption key is essential for the latter transformation.

A fundamental distiction between encryption schemes refers to this relation between the two keys. The simpler (and older) notion assumes that the encryption key equals the decryption key. Such schemes are called private-key (or *symmetric*). To use a private-key scheme, the legitimate parties must first agree on the secret key. This can be done by having one party generate the key at random and send it to the other party using a channel which is assumed to be secure. A crucial point is that the key is generated indepdendently of the plaintext, and so it can be generated and exchanged prior to the plaintext even being determined. Thus, private-key encryption is a way of extending a private channel over time: If the parties can use a private channel today (e.g., they are currently in the same physical location) but not tommorow, then they can use the private channel today to exchange a secret key which they may use tomorrow for secret communication. A simple example of a private-key encryption scheme is the *one-time pad*. The secret key is merely a uniformly chosen sequence of n bits, and an n-bit long ciphertext is produced by XORing the plaintext, bit-by-bit, with the key. The plaintext is recovered from the ciphertext in the same way. Clearly, the one-time pad provides absolute security. However, its usage of the key is inefficient; or, put in other words, it requires keys of length comparable to the total length of data communicated. In the rest of this chapter we will only discuss encryption schemes where n-bit long keys allow to communicated data of length greater than n (but still polynomial in n).

A new type of encryption schemes has emerged in the 1970's. In these schemes, called public-key (or *asymmetric*), the decryption key differs from the encryption key. Furthermore, it is infeasible to find the decryption key, given the encryption key. These schemes enable secure communication without ever using a secure channel. Instead, each party applies the key-generation algorithm to produce a pair of keys. The party, called P, keeps the decryption key, denoted d_P, secret and publishes the encryption key, denoted e_P. Now, any party can send P private messages by encrypting them using the encryption key e_P. Party P can decrypt these messages by using the decryption key d_P, but nobody else can do so.

5.1.2 A Formulation of Encryption Schemes

We start by defining the basic mechanism of encryption schemes. This definition says nothing about the security of the scheme (which is the subject of the next section).

Definition 5.1.1 (encryption scheme): *An* encryption scheme *is a triple, (G, E, D), of probabilistic polynomial-time algorithms satisfying the following two conditions*

1. *On input 1^n, algorithm G (called the* key generator*) outputs a pair of bit strings.*
2. *For every pair (e, d) in the range of $G(1^n)$, and for each $\alpha \in \{0,1\}^*$, algorithms E* (encryption) *and D* (decryption) *satisfy*

$$\Pr(D(d, E(e,\alpha)) = \alpha) = 1$$

where the probability is over the internal coin tosses of algorithms E and D.

The integer n serves as the security parameter *of the scheme. Each (e, d) in the range of $G(1^n)$ consitutes a pair of corresponding* encryption/decryption keys. *The string $E(e,\alpha)$ is the* encryption of the plaintext $\alpha \in \{0,1\}^*$ *using the encryption key e, whereas $D(d,\beta)$ is the* decryption of the ciphertext β *using the decryption key d.*

Observe that Definition 5.1.1 does not distinguish private-key encryption schemes from public-key ones. The difference between the two types is introduced in the security definitions: In a public-key scheme the "breaking algorithm" gets the encryption key (i.e., e) as an additional input (and thus $e \neq d$ follows); while in private-key schemes e is not given to the "breaking algorithm" (and thus one may assume, without loss of generality, that $e = d$).

Notation: In the sequel we write $E_e(\alpha)$ instead of $E(e,\alpha)$ and $D_d(\beta)$ instead of $D(d,\beta)$. Whenever there is little risk of confusion, we drop these subscripts. Also, we let $G_1(1^n)$ (resp., $G_2(1^n)$) denote the first (resp., second) element in the pair $G(1^n)$.

Comments: The above definition may be relaxed in several ways without significantly harming its usefulness. For example, we may relax Condition (2) and allow a negligible decryption error (e.g., $\mathsf{Pr}(D(G_2(1^n), E(G_1(1^n), \alpha)) \neq \alpha) < 2^{-n}$). Alternatively, one may postulate that Condition (2) holds for all but a negligible measure of the key-pairs generated by $G(1^n)$. At least one of these relaxations is essential for all popular suggestions of encryption schemes.

Another relaxation consists of restricting the domain of possible plaintexts (and ciphertexts). For example, one may restrict Condition (2) to α's of length $\ell(n)$, where $\ell : \mathbb{N} \mapsto \mathbb{N}$ is some fixed function. Given a scheme of the latter type (with plaintext length ℓ), we may construct a scheme as in Definition 5.1.1 by breaking pliantexts into blocks of length $\ell(n)$ and applying the restricted scheme separetly to each block. For more details see Section 5.2.4.

5.2 Security of Encryption Schemes

In this section we present two fundamental definitions of security and prove their equivalence. The first definition, called *semantic security*, is the most natural one. Semantic security is a computational complexity analogue of Shannon's definition of perfect privacy. Loosely speaking, an encryption scheme is semantically secure if the encryption of a message does not yield any information on the message to an adversary which is computationally restricted (e.g., to polynomial-time). The second definition has a more technical flavour. It interprets security as the infeasibility of distinguishing between encryptions of a given pair of messages. This definition is useful in demonstrating the security of a proposed encryption scheme, and for arguments concerning properties of cryptographic protocols which utilize an encryption scheme.

We stress that the definitions presented below go way beyond saying that it is infeasible to recover the plaintext from the ciphertext. The latter statement is indeed a minimal requirement from a secure encryption scheme, but we claim that it is way too weak a requirement: An encryption scheme is typically used in applications where obtaining specific partial information on the plaintext endangers the security of the application. When designing an application-independent encryption scheme, we do not know which partial information endangers the application and which does not. Furthermore, even if one wants to design an encryption scheme tailored to one's own specific applications, it is rare (to say the least) that one has a precise characterization of all possible partial information which endanger these applications. Thus, we require that it is infeasible to obtain any information about the plaintext from the ciphertext. Furthermore, in most applications the plaintext may not be uniformly distributed and some a-priori information regarding it is available to the adversary. We require that the secrecy of all partial information is preserved also in such a case. That is, even in presence of a-priori information on the plaintext, it is infeasible to obtain any (new) information about the plaintext from the ciphertext (beyond what is feasible to obtain from the a-priori information on the plaintext). The definition of semantic security postulates all of this.

To simplify the exposition, we adopt a non-uniform formulation. Namely, in the security definitions we expand the domain of efficient adversaries/algorithms to include polynomial-size circuits (rather than only probabilistic polynomial-time machines). Likewise, we make no computation restriction regarding the probability distribution from which messages are taken, nor regarding the a-priori information available on these messages. We note that employing such a non-uniform formulation (rather than a uniform one) may only strengthen the definitions; yet, it does weaken the implications proven between the definitions, since these (simpler) proofs make free usage of non-uniformity.

5.2.1 Semantic Security

Loosely speaking, semantic security means that whatever can be efficiently computed from the ciphertext, can be efficiently computed given only the length of the plaintext. Note that this formulation does not role out the possibility that the length of the plaintext can be inferred from the ciphertext. Indeed, some information about the length of the plaintext must be revealed by the ciphertext (see Exercise 2). We stress that other than information about the length of the plaintext, the ciphertext is required to yield nothing about the plaintext. Thus, an adversary gains nothing by intercepting ciphertexts sent between communicating parties who use a semantically secure encryption scheme.

We augment this formulation by requiring that the above remains valid even in presence of auxiliary partial information about the plaintext. Namely, whatever can be efficiently computed from the ciphertext and additional partial information about the plaintext, can be efficiently computed given only the length of the plaintext and the same partial information.

Secrurity holds only for plaintexts of length polynomial in the security parameter. This is captured below by the restriction $|X_n| = \text{poly}(n)$. Note that we cannot hope to provide computational security for plaintexts of unbounded length in the security parameter (see Exercise 1). Likewise, we restrict the functions f and h to be *polynomially-bounded*; that is, $|f(x)|, |h(x)| = \text{poly}(|x|)$.

The difference between private-key and public-key encryption schemes is manisfested in the definition of security. In the latter the adversary, trying to obtain information on the plaintext, is given the encryption key whereas in the former it is not. Thus, the difference between these schemes amounts to a difference in the adversary model (considered in the definition of security). We start by presenting the definition for private-key encryption schemes.

Definition 5.2.1 (semantic security – private-key): *An encryption scheme, (G, E, D), is* semantically secure (in the private-key model) *if there exists a polynomail-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \mapsto \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n*

$$\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) < \mathsf{Pr}\left(C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}$$

where $C'_n \stackrel{\rm def}{=} T(C_n)$ *is the circuit produced by* T *on input* C_n. (The probability in the above terms is taken over X_n as well as over the internal coin tosses of algorithms G and E.)

The function h provides both algorithms with partial information on the plaintext X_n. In addition both algorithms get the length of X_n. These algorithms then try to guess the value $f(X_n)$; namely, they try to infer information about the plaintext X_n. Loosely speaking, in semantically secure encryption scheme the ciphertext does not help in this inference task. That is, the success probability of any efficient algorithm (i.e., the circuit family $\{C_n\}$) which is given the ciphertext, can be matched, upto a negligible fraction, by the success probability of an efficient algorithm (i.e., the circuit family $\{C'_n\}$) which is not given the ciphertext at all. (See Exercise 8.)

Definition 5.2.1 refers to private-key encryption schemes. To derive a definition of security for public-key encryption schemes, the public-key (i.e., $G_1(1^n)$) should be given to the algorithms as an additional input. That is,

Definition 5.2.2 (semantic security – public-key): *An encryption scheme,* (G,E,D)*, is* semantically secure (in the public-key model) *if there exists a polynomail-time transformation,* T*, so that for every polynomial-size circuit family* $\{C_n\}$*, and for every* $\{X_n\}_{n\in\mathbb{N}}$*,* $f, h,$ $p(\cdot)$ *and* n *as in Definition 5.2.1*

$$\begin{aligned}&\mathsf{Pr}\left(C_n(G_1(1^n), E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right)\\ &\quad < \ \mathsf{Pr}\left(C'_n(G_1(1^n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}\end{aligned}$$

where $C'_n \stackrel{\rm def}{=} T(C_n)$.

For sake of simplicity, we refer to an encryption scheme which is semantically secure in the private-key (resp., public-key) model as to a semantically-secure private-key (resp., public-key) encryption scheme.

5.2.1.1 * Discussion of some definitional choices

We discuss some fine points regarding Definitions 5.2.1 and 5.2.2.

Effecient transformation of adversaries. Our definitions require that adversaries capturing what can be inferred from the ciphertext can be *effectively* transformed into "equivalent" adversaries which operate without being given the ciphertext. This is stronger than only requiring that corresponding "equivalent" adversaries exist. The strenthening seems especially appropriate since we are using a non-uniform model of adversary strategies.

Deterministic versus randomized adversaries. Our definitions refer implicitly to deterministic adversaries (modelled by non-uniform families of circuits which are typically assumed to be deterministic). This is in accordance with the general thesis by which the harm of non-uniform adversaries may be maximized by deterministic ones (i.e., by fixing the "worst" coin-sequence). However, we need to verify that a transformation of adversaries (as discussed above) referring to deterministic adversaries can be extended to randomized ones. This is indeed the case; see Exercise 5. (In fact, the above non-uniform formulation is equivalent to a uniform formulation in which the adversaries are given identical auxiliary input: See Exercise 4.)

Lack of restrictions on the functions f and g. We do not require that these functions are even computable. This seems strange at first glance. However, as we shall see in the sequel (see also Exercise 8), the meaning of semantic security is essentially that the distribution ensembles $(E(X_n), 1^{|X_n|}, h(X_n))$ and $(E(1^{|X_n|}), 1^{|X_n|}, h(X_n))$ are computationally indistinguishable (and so whatever C_n can compute can be computed by C'_n).

5.2.2 Indistinguishability of Encryptions

The following technical interpratation of security states that it is infeasible to distinguish the encryptions of two plaintexts (of the same length). That is, such ciphertexts are computationally indistinguishable as defined in Definition 3.2.2. Again, we start with the private-key variant.

Definition 5.2.3 (indistinguishability of encryptions – private-key): *An encryption scheme, (G, E, D), has* indistinguishable encryptions (in the private-key model) *if for every polynomial-size circuit family $\{C_n\}$, every polynomial p, all sufficiently large n and every $x, y \in \{0,1\}^{\mathrm{poly}(n)}$* (i.e., $|x| = |y|$) *and $z \in \{0,1\}^{\mathrm{poly}(n)}$,*

$$|\mathsf{Pr}\left(C_n(z, E_{G_1(1^n)}(x))=1\right) - \mathsf{Pr}\left(C_n(z, E_{G_1(1^n)}(y))=1\right)| < \frac{1}{p(n)}$$

The probability in the above terms is taken over the internal coin tosses of algorithms G and E.

The string z models additional information, on the potential plaintexts, given to the algorithm which tries to distinguish the encryptions of these messages. In fact, the string z can be incorporated into the circuit C_n (so that the circuit models both the adversary's strategy and its a-priori information): See Exercise 6.

> Author's Note: *Option: define indistinguishable-encryptions without auxiliary inputs, and explain that non-uniformity takes care of it.*

Again, the security definition for public-key encryption schemes can be derived by adding the public-key (i.e., $G_1(1^n)$) as an additional input to the algorithm. That is,

Definition 5.2.4 (indistinguishability of encryptions – public-key): *An encryption scheme, (G, E, D), has* indistinguishable encryptions (in the public-key model) *if for every polynomial-size circuit family $\{C_n\}$, and every $p(\cdot)$, n, x, y and z as in Definition 5.2.3*

$$|\Pr\left(C_n(z, G_1(1^n), E_{G_1(1^n)}(x))=1\right) - \Pr\left(C_n(z, G_1(1^n), E_{G_1(1^n)}(y))=1\right)| < \frac{1}{p(n)}$$

For sake of simplicity, we refer to an encryption scheme which has indistinguishable encryptions in the private-key (resp., public-key) model as to a ciphertext-indistinguishable private-key (resp., public-key) encryption scheme.

5.2.3 Equivalence of the Security Definitions

The following theorem is stated and proven for private-key encryption schemes. Similar results hold for public-key encryption schemes (see Exercise 7).

Theorem 5.2.5 *: A private-key encryption scheme is semantically secure if and only if it has indistinguishable encryptions.*

Let (G, E, D) be an encryption scheme. We formulate a proposition for each of the two directions of the above theorem. Both propositions are in fact stronger than the corresponding direction stated in Theorem 5.2.5. The more useful direction is stated first: It asserts that the technical interpration of security, in terms of ciphertext-indistinguishability, implies the natural notion of sematic security. Thus, the following proposition yields a methodology for designing sematically secure encryption schemes – design and prove your scheme to be ciphertext-indistinguishability, and conclude (by the following) that it is sematically secure.

Proposition 5.2.6 *: Suppose that (G, E, D) is a ciphertext-indistinguishable private-key encryption scheme. Then (G, E, D) is semantically-secure. Furthermore, the conclusion holds even if the definition of indistinguishable encryptions is restricted to the case where no auxiliary information is given* (i.e., $z = \lambda$).

(In view of the comment above – see Exercise 6 – the furthermore clause follows anyhow from the main claim.

Proposition 5.2.7 *: Suppose that (G, E, D) is a semantically secure private-key encryption scheme. Then (G, E, D) has indistinguishable encryptions. Furthermore, the conclusion holds even if the definition of semantic security is restricted to the special case where h is a constant function, X_n is uniformly distributed over a set containing two strings, and the transformation T is not even required to be computable.*

Observe that in the latter special case, it suffices to consider a function f which assigns '1' to one string in the support of X_n and '0' to the other.

Proof of Proposition 5.2.6: Suppose that (G, E, D) has indistinguishable encryptions (in the restricted sense of no auxiliary input; i.e., $z = \lambda$). We show that (G, E, D) is semantically secure by constructing for every polynomial-size circuit family $\{C_n\}$, a polynomial-size circuit family $\{C'_n\}$ so that for every $\{X_n\}_{n\in\mathbb{N}}$, f and h, $\{C'_n\}$ guesses $f(X_n)$ from $(1^{|X_n|}, h(X_n))$ essentially as good as $\{C_n\}$ guesses $f(X_n)$ from $(E(X_n), 1^{|X_n|}, h(X_n))$.

Let C_n be a circuit which tries to infer partial information (i.e., the value $f(X_n)$) from the encryption of the message X_n ($1^{|X_n|}$ and a-priori information $h(X_n)$). Namely, on input $E(\alpha)$ and $(1^{|\alpha|}, h(\alpha))$, the circuit C_n tries to guess $f(\alpha)$. We construct a new circuit, C'_n, which performs as well without getting the input $E(\alpha)$. The new circuit consists of invoking C_n on input $E_{G_1(1^n)}(1^{|\alpha|})$ and $(1^{|\alpha|}, h(\alpha))$. That is, C'_n invokes the key-generator G (on input 1^n), obtains an encryption-key $e = G_1(1^n)$, invokes the encryption algorithm with key e and ("dummy") plaintext $1^{|\alpha|}$, obtaining a ciphertext which it feeds to C_n together with the inputs $(1^{|\alpha|}, h(\alpha))$. Observe that C'_n can be efficiently computed from C_n (i.e., by augmenting it with the unifom circuit for computing algorithms G and E).

Indistinguishability of encryptions will be used to prove that C'_n performs as well as C_n. Note that the construction of C'_n does not depend on the functions h and f or on the distribution of messages to be encrypted.

Claim: Let $\{C'_n\}$ be as above. Then, for any polynomial p, and all sufficiently large n's

$$\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) < \mathsf{Pr}\left(C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}$$

Proof: To simplify the notations, let us incorporate $1^{|\alpha|}$ into $h(\alpha)$. Using the definition of C'_n, we can rewritten the claim as asserting

$$\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n), h(X_n)) = f(X_n)\right) < \mathsf{Pr}\left(C_n(E_{G_1(1^n)}(1^{|X_n|}), h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}$$

Assume, to the contradiction that for some polynomail p and infinitely many n's the above inequality is violated. Then, for each such n, we have $\mathsf{E}(\Delta(X_n)) > 1/p(n)$, where

$$\Delta(x) \stackrel{\text{def}}{=} \left|\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(x), h(x)) = f(x)\right) - \mathsf{Pr}\left(C_n(E_{G_1(1^n)}(1^{|x|}), h(x)) = f(x)\right)\right| \qquad (5.1)$$

Let $x_n \in \{0,1\}^{\text{poly}(n)}$ be a string for which $\Delta(x)$ is maximum, and so $\Delta(x_n) > 1/p(n)$. Using this x_n, we introduce a new circuit D_n, which incorporates $f(x_n)$ and $h(x_n)$, and operates as follows. On input $\beta = E(\alpha)$, the circuit D_n invokes $C_n(\beta, h(x_n))$ and outputs 1 if and only if C_n outputs the value $f(x_n)$. (Otherwise, D_n outputs 0.) Clearly,

$$\mathsf{Pr}\left(D_n(E_{G_1(1^n)}(\alpha)) = 1\right) = \mathsf{Pr}\left(C_n(E_{G_1(1^n)}(\alpha), h(x_n)) = f(x_n)\right)$$

and so

$$\left|\mathsf{Pr}\left(D_n(E_{G_1(1^n)}(x_n)) = 1\right) - \mathsf{Pr}\left(D_n(E_{G_1(1^n)}(1^{|x_n|})) = 1\right)\right| > \frac{1}{p(n)}$$

in contradiction to our hypothesis that E has indistinguisahble encryptions. Thus, the claim follows. □

Proposition 5.2.6 follows. ■

Proof of Proposition 5.2.7: We now show that if (G, E, D) has distinguishable encryptions then it is not semantically secure (not even in the restricted sense mentioned in the furthermore-clause of the proposition). Towards this end, we assume that there exists a polynomial p, a polynomial-size circuit family $\{D_n\}$, such that for infinitely many n's there exists $x_n, y_n \in \{0,1\}^{\text{poly}(n)}$ so that

$$\left|\mathsf{Pr}\left(D_n(E_{G_1(1^n)}(x_n))=1\right) - \mathsf{Pr}\left(D_n(E_{G_1(1^n)}(y_n))=1\right)\right| > \frac{1}{p(n)} \tag{5.2}$$

(Recall that the auxiliary input z in Definition 5.2.3 can be incorporated into the circuit: See Exercise 6.) We define a random variable X_n which is uniformly distributed over $\{x_n, y_n\}$, and $f : \{0,1\}^* \mapsto \{0,1\}$ so that $f(x_n) = 1$ and $f(y_n) = 0$. Note that $f(X_n) = 1$ with probability 1/2 and is 0 otherwise. (We may define $h(\alpha) = 1^{|\alpha|}$ and supply it to C_n defined below, but this has no real effect.) We will show that D_n can be transformed into a polynomial-size circuit C_n which guesses the value of $f(X_n)$, from the encryption of X_n, and does so significantly better that with probability $\frac{1}{2}$. This violates (even the restricted form of) semantic security, since no circuit (regardless of its size) can guess $f(X_n)$ better than with probability 1/2 when only given $1^{|X_n|}$ (since given the constant value $1^{|X_n|}$, the value of $f(X_n)$ is uniformly distributed over $\{0,1\}$).

Let us assume, without loss of generality, that

$$\mathsf{Pr}\left(D_n(E_{G_1(1^n)}(x_n))=1\right) > \mathsf{Pr}\left(D_n(E_{G_1(1^n)}(y_n))=1\right) + \frac{1}{p(n)} \tag{5.3}$$

We modify D_n so that on input $\beta = E(\alpha)$, the new circuit C_n feeds D_n with input β and outputs 1 if D_n outputs 1 (otherwise, C_n outputs 0). It is left to analyze the probability that $C_n(E(X_n))$ equals $f(X_n)$.

$$\begin{aligned}
\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n))=f(X_n)\right) &= \frac{1}{2}\cdot\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n))=f(X_n)|X_n=x_n\right) \\
&\quad +\frac{1}{2}\cdot\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n))=f(X_n)|X_n=y_n\right) \\
&= \frac{1}{2}\cdot\left[\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(x_n))=1\right)+\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(y_n))=0\right)\right] \\
&= \frac{1}{2}\cdot\left[\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(x_n))=1\right)+1-\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(y_n))=1\right)\right] \\
&> \frac{1}{2}+\frac{1}{2p(n)}
\end{aligned}$$

where the inequality is due to Eq. (5.3). In contrast, as observed above, for every circuit C'_n, $\Pr\left(C'_n(1^{|X_n|})=f(X_n)\right) \leq \frac{1}{2}$. This contradicts the hypothesis that the scheme is semantically secure (even in the restricted sense mentioned in the furthermore-clause of the proposition). Thus, the poroposition follows. ■

5.2.4 Multiple Messages

The above definitions only refer to the security of a scheme which is used to encrypt a single plaintext (per key generated). Clearly, in reality, we want to use encryption schemes to encrypt many messages with the same key. We show that in the public-key model, security in the single-message setting (discussed above) implies security in the multiple-message setting (defined below). This is not necessarily true for the private-key model. Let us start by presenting the definitions.

Definition 5.2.8 (semantic security – mulitple messages): *An encryption scheme, (G,E,D), is* **semantically secure for multiple messages** in the private-key model *if there exists a polynomail-time transformation, T, so that for every polynomail $t(\cdot)$ and every polynomial-size circuit family $\{C_n\}$, for every ensemble $\{\overline{X}_n\}_{n\in\mathbb{N}}$, with $|\overline{X}_n| = t(n)\cdot\mathrm{poly}(n)$, every pair of functions $f,h:\{0,1\}^*\mapsto\{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n*

$$\begin{aligned}
&\Pr\left(C_n(\overline{E}_{G_1(1^n)}(\overline{X}_n),1^{|\overline{X}_n|},h(\overline{X}_n))=f(\overline{X}_n)\right)\\
&\quad< \Pr\left(C'_n(1^{|\overline{X}_n|},h(\overline{X}_n))=f(\overline{X}_n)\right)+\frac{1}{p(n)}
\end{aligned}$$

where $C'_n \stackrel{\rm def}{=} T(C_n)$, $\overline{X}_n = (X_n^{(1)},...,X_n^{(t(n))})$ and $\overline{E}_e(\overline{X}_n) \stackrel{\rm def}{=} E_e(X_n^{(1)}),...,E_e(X_n^{(t(n))})$. An encryption scheme, (G,E,D), is **semantically secure for multiple messages** in the public-key model *if for $t(\cdot)$, $\{C_n\}$, $\{C'_n\}$, $\{\overline{X}_n\}_{n\in\mathbb{N}}$, f,h, $p(\cdot)$ and n as above*

$$\begin{aligned}
&\Pr\left(C_n(G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{X}_n),1^{|\overline{X}_n|},h(\overline{X}_n))=f(\overline{X}_n)\right)\\
&\quad< \Pr\left(C'_n(G_1(1^n),1^{|\overline{X}_n|},h(\overline{X}_n))=f(\overline{X}_n)\right)+\frac{1}{p(n)}
\end{aligned}$$

Definition 5.2.9 (indistinguishability of encryptions – mulitple messages): *An encryption scheme, (G,E,D), has* **indistinguishable encryptions for multiple messages** in the private-key model *if for every polynomail $t(\cdot)$, every polynomial-size circuit family $\{C_n\}$, every polynomial p, all sufficiently large n and every $x_1,...,x_{t(n)},y_1,...,y_{t(n)}\in\{0,1\}^{\mathrm{poly}(n)}$ and $z\in\{0,1\}^{\mathrm{poly}(n)}$,*

$$\left|\Pr\left(C_n(z,\overline{E}_{G_1(1^n)}(\bar{x}))=1\right)-\Pr\left(C_n(z,\overline{E}_{G_1(1^n)}(\bar{y}))=1\right)\right|<\frac{1}{p(n)}$$

where $\bar{x}=(x_1,...,x_{t(n)})$, $\bar{y}=(y_1,...,y_{t(n)})$, and $\overline{E}_e$ is as in Definition 5.2.8. An encryption scheme, (G,E,D), has **indistinguishable encryptions for multiple messages** in the public-key model *if for* $t(\cdot)$, $\{C_n\}$, p, n *and* $x_1, ..., x_{t(n)}, y_1, ..., y_{t(n)}, z$ *as above*

$$|\mathsf{Pr}\left(C_n(G_1(1^n), z, \overline{E}_{G_1(1^n)}(\bar{x}))=1\right) - \mathsf{Pr}\left(C_n(G_1(1^n), z, \overline{E}_{G_1(1^n)}(\bar{y}))=1\right)| < \frac{1}{p(n)}$$

The equivalence of Definitions 5.2.8 and 5.2.9 can be established analogously to the proof of Theorem 5.2.5. Thus, proving that single-message security implies multiple-message security for one definition of security yields the same for the other. We may thus concentrate on the ciphertext-indistinguishability definitions. We first consider public-key encryption schemes.

Theorem 5.2.10 *: A* public-key *encryption scheme has indistinguishable encryptions* for multiple messages *(i.e., satisfies Definition 5.2.9 in the public-key model) if and only if it has indistinguishable encryptions* for a single message *(i.e., satisfies Definition 5.2.4).*

Proof: Clearly, multiple-message security implies single-message security as a special case. The other direction follows by adapting the proof of Theorem 3.2.6 to the non-uniform case.

Suppose, towards the contradiction, that there exist a polynomail $t(\cdot)$, a polynomial-size circuit family $\{C_n\}$, and a polynomial p, such that for infinitely many n's, there exists $x_1, ..., x_{t(n)}, y_1, ..., y_{t(n)} \in \{0,1\}^{\mathrm{poly}(n)}$ so that

$$|\mathsf{Pr}\left(C_n(G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{x}))=1\right) - \mathsf{Pr}\left(C_n(G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{y}))=1\right)| > \frac{1}{p(n)}$$

(By the above, we may assume without loss of generality that $z = \lambda$. Alternatively, one may incorporate the z's into the circuit family.) Let us consider such a generic n and the corresponding sequences $x_1, ..., x_{t(n)}, y_1, ..., y_{t(n)}$. We now use a hybrid argument. We start by observing that there exists an $i \in \{0, ..., t(n)-1\}$ so that

$$\left|\mathsf{Pr}\left(C_n(G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{h}^{(i)}))=1\right) - \mathsf{Pr}\left(C_n(G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{h}^{(i+1)}))=1\right)\right| > \frac{1}{t(n)\cdot p(n)}$$

where $\bar{h}^{(j)} \stackrel{\mathrm{def}}{=} x_1 \cdots x_j \cdot y_{j+1} \cdots y_{t(n)}$. We now construct a circuit D_n, which on input e and β operates as follows. (The constructiuon relies on D_n's knowledge of the encryption-key and hence the public-key model is essential for it.) For every $j \leq i$, the circuit D_n generates an encryption of x_j using the encryption key e. Similarly, for every $j > i+1$, the circuit D_n generates an encryption of y_j using the encryption key e. Let us denote the resulting ciphertexts by $\beta_1, ..., \beta_i, \beta_{i+2}, ..., \beta_{t(n)}$. Finally, D_n invokes C_n on input e and $\beta_1, ..., \beta_i, \beta, \beta_{i+2}, ..., \beta_{t(n)}$.

Now, suppose that β is a (random) encryption of x_{i+1} with key e; that is, $\beta = E_e(x_{i+1})$. Then, $D_n(e, \beta) = C_n(e, \bar{h}^{(i+1)})$, where equality means that the two random variables are identically distributed. Similarly, for $\beta = E_e(y_{i+1})$ we have $D_n(e, \beta) = C_n(e, \bar{h}^{(i)})$. Thus, D_n (given the encryption key) distinguishes the encryption of x_{i+1} from the encryption of y_{i+1}, in contradiction to our hypothesis that (G, E, D) is a ciphertext-indistinguishable public-key encryption scheme. The theorem follows. ■

In contrary to Theorem 5.2.10, ciphertext-indistinguishability for a single message does NOT necessarily imply ciphertext-indistinguishability for multiple messages *in the private-key model*. A counterexample to such a claim follows.

Proposition 5.2.11 *Suppose that there exist pseudorandom generators* (robust against polynomial-size circuits). *Then, there exists a private-key encryption scheme which satisfies Definition 5.2.3 but does not satisfy Definition 5.2.9.*

Proof: We start with the construction of the private-key encryption scheme. The encryption/decryption key for security parameter n is a uniformly distributed n-bit long string, denoted s. To encrypt a ciphertext, x, the encryption algorithm uses the key s as a seed for a pseudorandom generator, denoted g, which stretches seeds of length n into sequences of length $|x|$. The ciphertext is obtained by a bit-by-bit exclusive-or of x and $g(s)$. Decryption is done in the obvious manner.

It is easy to see that this encryption scheme satisfies Definition 5.2.3. Specifically, suppose that a circuit C_n can distinguish encryptions of x from encryptions of y (relative to security parameter n), where $|x| = |y| = \text{poly}(n)$. That is,

$$|\mathsf{Pr}(C_n(x \oplus g(U_n))=1) - \mathsf{Pr}(C_n(y \oplus g(U_n))=1)| > \frac{1}{\text{poly}(n)}$$

where U_n is uniformly distributed over $\{0,1\}^n$. Since $\mathsf{Pr}(C_n(x \oplus U_{|x|})=1) = \mathsf{Pr}(C_n(y \oplus U_{|y|})$, we have without loss of generality

$$|\mathsf{Pr}(C_n(x \oplus g(U_n))=1) - \mathsf{Pr}(C_n(x \oplus U_{|x|})=1)| > \frac{1}{2 \cdot \text{poly}(n)}$$

Incorporating x into the circuit C_n we obtain a circuit which distinguishes random sequences from sequences generated by g, in contradiction to our hypothesis.

Finally, we observe that the above encryption scheme is no secure when encrypting two messages. Intuitively, any plaintext-ciphertext pair yields a corresponding prefix of the pseudorandom sequence, and knowledge of this prefix violates the security of additional plaintexts. For concreteness, let us show that given the encryption of the plaintexts 0^n and U_n, we can retreive U_n. On input the ciphertexts β_1, β_2, the adversary, knowing that the first plaintext is 0^n, first retreives the pseudorandom sequence (i.e., it is just β_1) and next retreives the second plaintext (i.e., by computing $\beta_2 \oplus \beta_1$). ■

Comment: Indeed, as we show below, the above construction can be modified to yield a private-key encryption secure for multiple message encryptions. All that is needed is to make sure that the same part of the pseudorandom sequence is never used twice.

5.3 Constructions of Secure Encryption Schemes

In this subsection we present constructions of secure private-key and public-key encryption schemes. Here and throughout this section security means semantic security in the multiple-message setting. Recall that this is equivalent to ciphertext-indistinguishability (in the multiple-message setting). Also recall that for public-key schemes it suffices to prove ciphertext-indistinguishability (in the single-message setting). The main results of this section are

- Using any (non-uniformly robust) pseudorandom function, one can construct secure private-key encryption schemes (in the multiple message setting). Recall, that the former can be constructed using any (non-uniformly strong) one-way function.
- Using any (non-uniform strong) trapdoor one-way permutation, one can construct secure public-key encryption schemes (in the multiple message setting).

In addition, we review some popular suggestions for private and public-key encryption schemes.

Probabilistic Encryption: Before starting, we note that a secure *public-key* encryption scheme must employ a probabilistic (i.e., randomized) encryption algorithm. Otherwise, given the encryption key as (additional) input, it is easy to distinguish the encryption of the all-zero message from the encryption of the all-ones message. The same holds for *private-key* encryption schemes when considering the multi-message setting.[1] For example, using a deterministic (private-key) encryption algorithm allows the adversary to distinguish two encryptions of the same message from the encryptions of a pair of different messages. This explains the linkage between the above robust security definitions and the *randomization paradigm* (discussed below).

5.3.1 Stream-Ciphers

Author's Note: *Define and discuss the notion of a stream-cipher.*

Author's Note: *The definition of a stream-cipher deviates from our formulation of encryption schemes (in having memory – counter).*

It is common practice to use "pseudorandom generators" as a basis for private-key stream ciphers. We stress that this is a very dangerous practice when the "pseudorandom generator" is easy to predict (such as the linear congruential generator or some modifications

[1] We note that the above does not hold with respect to private-key schemes in the single-message setting. (Hint: the private-key can be augmented to include a seed for a pseudorandom generator, the output of which can be used to eliminate randomness from the encryption algorithm. Question: why does the argument fail in the multi-message private-key setting? Same for the public-key setting).

of it which output a constant fraction of the bits of each resulting number). However, this common practice becomes sound provided one uses pseudorandom generators (as defined in Chapter 3).

Author's Note: *Ellaborate.*

5.3.2 Block-Ciphers

Many encryption schemes are more conveniently presented by first presenting a restricted type of encryption scheme which we call a *block-cipher*.[2] In contrast to encryption schemes (as defined in Definition 5.1.1), block-ciphers (defined below) are only required to operate on plaintext of a specific length (which is a function of the security parameter). As we shall see, given a secure block-cipher we can easily construct a (general) secure encryption scheme.

Definition 5.3.1 (block-cipher): *A* block-cipher *is a triple, (G, E, D), of probabilistic polynomial-time algorithms satisfying the following two conditions*

1. *On input 1^n, algorithm G outputs a pair of bit strings.*
2. *There exists a polynomially-bounded function $\ell : \mathbb{N} \mapsto \mathbb{N}$, called the* block length, *so that for every pair (e, d) in the range of $G(1^n)$, and for each $\alpha \in \{0,1\}^{\ell(n)}$, algorithms E and D satisfy*
$$\Pr(D(d, E(e, \alpha)) = \alpha) = 1$$

All conventions are as in Definition 5.1.1.

Typically, either $\ell(n) = \Theta(n)$ or $\ell(n) = 1$. Analogously to Definition 5.1.1, the above definition does not distinguish private-key encryption schemes from public-key ones. The difference between the two types is captured in the security definitions, which remain as they were above with the modification that we only consider plaintexts of length $\ell(n)$. For example, the analogue of Definition 5.2.1 reads

Definition 5.3.2 (semantic security – private-key block-ciphers): *A block-cipher, (G, E, D), with block length ℓ is* semantically secure (in the private-key model) *if there exists a polynomail-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \ell(n)$, and $f, h, p(\cdot)$ and n as in Definition 5.2.1*

$$\Pr\left(C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) < \Pr\left(C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}$$

where $C'_n \stackrel{\text{def}}{=} T(C_n)$ is the circuit produced by T on input C_n. (The probability in the above terms is taken over X_n as well as over the internal coin tosses of algorithms G and E.)

[2] Doing so we abuse standard terminology by which a block-cipher must, in addition to operating on plaintext of specific length, produce ciphertexts equal in length to the length of the corresponding plaintexts.

There are several obvious ways of transforming a block-cipher into a general encryption scheme. The basic idea is to break the plaintexts (for the resulting scheme) into blocks and encode each block separately by using the block-cipher. Thus, the security of the block-cipher (in the multiple-message settings) implies the security of the resulting encryption scheme. The only technicality we need to deal with is how to encrypt plaintexts of length which is not an integer multiple of the block-length (i.e., $\ell(n)$). This is easily resolved by padding (the last block).

Construction 5.3.3 *Let (G, E, D) be a block-cipher with block length function ℓ. We construct an encryption scheme, (G', E', D') as follows. The key-generation algorithm, G', is identical to G. To encrypt a message α* (with encryption key e generated under security parameter n)*, we break it into consequetive blocks of length $\ell(n)$, while possibly augmenting the last block. Let $\alpha_1, ..., \alpha_t$ be the resulting blocks. Then*

$$E'_e(\alpha) \stackrel{\text{def}}{=} (1^{|\alpha|}, E_e(\alpha_1), ..., E_e(\alpha_t))$$

To decrypt the ciphertext $(1^m, \beta_1, ..., \beta_t)$ (with decryption key d)*, we let $\alpha_i = D_d(\beta_i)$ for $i = 1, ..., t$, and let the plaintext be the m-bit long prefix of the concatanated string $\alpha_1 \cdots \alpha_t$.*

The above construction yields ciphertexts which reveal the exact length of the plaintext. Recall that this is not prohibited by the definitions of security. However, we can easily construct encryption schemes which hide some information about the length of the plaintext; see examples in Exercise 9. (Recall that we cannot hope to entirely hide the length.) Also, note that the above construction applies even to the special case where ℓ is identically 1.

Theorem 5.3.4 *Let (G, E, D) and (G', E', D') be as in Contruction 5.3.3. Suppose that the former a secure private-key* (resp., public-key) *block-cipher. Then the latter is a secure private-key* (resp., public-key) *encryption scheme.*

Proof Sketch: We use the definition of ciphertext-indistinguishability. That is, assuming towards the contradiction that one can distinguish ciphertexts (or multiple ciphertexts) of (G', E', D'), one obtains a distinguisher for multiple-ciphertexts of (G, E, D). $\Box$

5.3.3 Private-key encryption schemes

Secure private-key encryption schemes can be easily constructed using any efficiently computable pseudorandom function ensemble (see Section 3.6). We first present a block cipher with block length $\ell(n) = n$. The key generation algorithm consists of selecting a seed, denoted s, for such a function, denoted f_s. To encrypt a message $x \in \{0,1\}^n$ (using key s), the encryption algorithm uniformly selects a string $r \in \{0,1\}^n$ and produces the ciphertext $(r, x \oplus f_s(r))$. To decrypt the ciphertext (r, y) (using key s), the decryption algorithm just computes $y \oplus f_s(r)$. Formally,

Construction 5.3.5 *Let $F = \{F_n\}$ be an efficiently computable function ensemble and let I and V be the algorithms associated with it. That is, $I(1^n)$ selects a function with distribution F_n and $V(i,x)$ returns $f_i(x)$, where f_i is the function associated with the string i. We define a private-key block cipher, (G,E,D),* (with block length $\ell(n) = n$) *by letting $G(1^n) = I(1^n)$, letting $E_i(x) = (r, V(i,r) \oplus x)$ where $x \in \{0,1\}^n$ and r is uniformly chosen in $\{0,1\}^n$. Finally, we let $D_i(r,y) = V(i,r) \oplus y$*

> Author's Note: *Define and discuss pseudoradomness wrt circuits in the PRG chapter.*

Theorem 5.3.6 *Let F and (G,E,D) be as in Contruction 5.3.5, and suppose that F is pseudorandom with respect to polynomail-size circuits. Then (G,E,D) is secure.*

Proof Sketch: The proof consists of two steps (suggested as a general methodology in Section 3.6):

1. Prove that an idealized version of the scheme, in which one uses a uniformly selected function $f:\{0,1\}^n \mapsto \{0,1\}^n$, rather than the pseudorandom function f_s, is secure (in the sense of ciphertext-indistinguishability).
2. Conclude that the real scheme (as presented above) is secure (since otherwise one could distinguish a pseudorandom function from a truly random one).

□

Comments. Note that we could have gotten rid of the randomization if we had allowed the encryption algorithm to be history dependent (e.g., use a counter in the role of r). Furthermore, if the encryption scheme is used for FIFO communication between the parties and both can maintain the counter value then there is no need for the sender to send the counter value. On the other hand, the common practice of using pseudorandom permutations as block-ciphers[3] is NOT semantically secure (as one can distinguish two encryptions of the same message from encryptions of two different messages).

5.3.4 Public-key encryption schemes

> Author's Note: *Should one present the ineffecient GM82-scheme first?*

> Author's Note: *The rest of this section is taken from my survey, and needs to be greatly ellaborated.*

[3] That is, letting $E_i(x) = p_i(x)$, where p_i is the perumtation associated with the string i.

The randomization paradigm [GM]: To demonstrate this paradigm suppose we have a trapdoor one-way permutation, $\{p_\alpha\}_\alpha$, and a hard-core predicate, b, for it. The key generation algorithm consists of selecting at random a permutation p_α together with a trapdoor for it: The permutation (or rather its description) serves as the public-key, whereas the trapdoor serves as the private-key. To encrypt a single bit σ (using public key p_α), the encryption algorithm uniformly selects an element, r, in the domain of p_α and produces the ciphertext $(p_\alpha(r), \sigma \oplus b(r))$. To decrypt the ciphertext (y, τ) (using the private key), the decryption algorithm just computes $\tau \oplus b(p_\alpha^{-1}(y))$ (where the inverse is computed using the trapdoor (i.e., private-key)). The above scheme is quite wasteful in bandwidth; however, the paradigm underlying its construction is valuable in practice. For example, it is certainly better to randomly pad messages (say using padding equal in length to the message) before encrypting them using RSA than to employ RSA on the plain message. Such a heuristic could be placed on firm grounds if the following conjecture is supported. That is, assume that the first $n/2$ least significant bits of the argument constitute a hard-core function of RSA with n-bit long moduli. Then, encrypting $n/2$-bit messages by padding the message with $n/2$ random bits and applying RSA (with an n-bit moduli) on the result constitutes a secure public-key encryption system, hereafter referred to as Randomized RSA.

An alternative public-key encryption scheme is presented in [BlGw]. The encryption scheme augments the Construction 3.4.2 (of a pseudorandom generator based on one-way permutations) as follows. The key-generation algorithm consists of selecting at random a permutation p_α together with a trapdoor. To encrypt the n-bit string x (using public key p_α), the encryption algorithm uniformly selects an element, s, in the domain of p_α and produces the ciphertext $(p_\alpha^n(s), x \oplus G_\alpha(s))$, where $G_\alpha(s) = b(s) \cdot b(p_\alpha(s)) \cdots b(p_\alpha^{n-1}(s))$. (We use the notation $p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x))$ and $p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x))$.) To decrypt the ciphertext (y, z) (using the private key), the decryption algorithm first recovers $s = p_\alpha^{-n}(y)$ and then outputs $z \oplus G_\alpha(s)$.

Assuming that factoring Blum Integers (i.e., products of two primes each congruent to $3 \pmod 4$) is hard, one may use the modular squaring function in role of the trapdoor permutation above (see [BlGw,ACGS,VV,FnSn]). This yields a secure public-key encryption scheme (depicted in Figure 5.1) with efficiency comparable to that of RSA. Recall that RSA itself is not secure (as it employs a deterministic encryption algorithm), whereas Randomized RSA (defined above) is not known to be secure under standard assumption such as intractability of factoring (or of inverting the RSA function).[4]

5.4 Stronger notions of security

The security definitions presented above are "static" in the sense that they persive the adversary as a passive line-tapper who intercepts ciphertexts and tries to figure out in-

[4] Recall that Randomized RSA is secure assuming that the $n/2$ least significant bits constitute a hard-core function for n-bit RSA moduli. We only know that the $O(\log n)$ least significant bits constitute a hard-core function for n-bit moduli [ACGS].

private-key: Two $n/2$-bit primes p, q, each congruent to $3 \pmod 4$.

public-key: Their product $N \stackrel{\text{def}}{=} pq$.

encryption of message $x \in \{0,1\}^n$:

1. Uniformly select $s_0 \in \{1, ..., N\}$.
2. For $i = 1, .., n+1$, compute $s_i \leftarrow s_{i-1}^2 \bmod N$ and $\sigma_i = \text{lsb}(s_i)$.

The ciphertext is (s_{n+1}, y), where $y = x \oplus \sigma_1\sigma_2 \cdots \sigma_n$.

decryption of the ciphertext (r, y). Let $d = 2^{-n} \bmod \phi(N)$ [precomputed].

1. Let $s_1 \leftarrow r^d \bmod N$.
2. For $i = 1, .., n$, compute $\sigma_i = \text{lsb}(s_i)$ and $s_{i+1} \leftarrow s_i^2 \bmod N$.

The plaintext is $y \oplus \sigma_1\sigma_2 \cdots \sigma_n$.

Figure 5.1: The Blum–Goldwasser Public-Key Encryption Scheme [BlGw].

formation regarding the corresponding plaintexts. Furthermore, the encrypted messages are selected obliviously from public information available in the public-key case. In many setting this basic level of security suffices, but in other settings one may need to consider more "active" adversaries. Before considering these stronger notions of security, we note that the above basic definitions also cover the case where the adversary obatins some of the plaintexts themselves. In this case it is still infeasible for him/her to obtain infromation about the missing plaintexts (see Exercise 10).

In some settings it is feasible for the adversary to make the sender encrypt a message of the adversary's choice, and in some settings the adversary may even make the receiver decrypt a ciphertext of the adversary's choice. This gives rise to *chosen message attacks* (resp., *chosen ciphertext attacks*) which are not covered by the above security definitions. The first two subsections are devoted to these two types of attacks.

We conclude with a discussion of non-mallaebale encryption schemes. Loosely speaking, these not only disallow the adversary to learn anything about the plaintexts, but also disallow it to produce ciphertexts of related messages (i.e., given $E_e(x)$ it should be infeasible to generate an encrypytion of $x \oplus 1^{|x|}$).

Author's Note: *This section is yet to be written. The current material is merely a collection of extracts from my survey.*

5.4.1 Chosen plaintext attack

Clearly, a chosen message attack is of no help to an adversary which attacks a public-key (as it may encrypt messages by itself). But still may select message space based on public-key??

???

5.4.2 Chosen ciphertext attack

> Author's Note: *HERE we refer to a version where the ciphertext to be cracked is supplied after the attack... The case where the attack takes place after the ciphertext is known is dealt in the next subsection.*

Clearly, the private-key encryption scheme based on pseudorandom functions (described above) is secure also against such attacks. Public-key encryption schemes secure against Chosen Ciphertext Attacks can be constructed, assuming the existence of trapdoor permutations and utilizing non-interactive zero-knowledge proofs (which can be constructed under this assumption).

5.4.3 Non-malleable encryption schemes

DEFINITION...

It is easy to turn a private-key encryption scheme into a non-malleable one, by using a message authentication scheme on top. Non-malleable public-key encryption schemes are known to exist assuming the existence of trapdoor permutation.

5.5 Miscellaneous

> Author's Note: *The entire material below is fragmented and tentative.*

5.5.1 Historical Notes

The notion of private-key encryption scheme seems almost as ancient as the alphabet itself. Furthermore, the development of encryption methods went along with the development of communication media. As the amounts of communication grow, more efficient and sophisticated encryption methods were required. Computational complexity considerations were explicitly introduced into the arena by Shannon. In his work [S49], Shannon considered the classical setting where no computational considerations are present. He showed that in this information theoretic setting, secure communication of information was possible only as long as its entropy is lower than the entropy of the key. Thus, if one wishes to have an encryption scheme which is capable of handling messages with total entropy exceeding the length of the key then one must settle for a computational relaxion of the secrecy condition. That is, rather than requiring that the ciphertext yields no information on the plaintext, one has to require that such information cannot be efficiently computed from the ciphertext. The latter requirement indeed coincides with the above definition of semantic security.

The notion of public-key encryption scheme was introduced by Diffie and Hellman [DH76]. First concrete candidates were suggested by Rivest, Shamir and Adleman [RSA78] and by Merkle and Hellman [MH78]. However, satisfactory definitions of security were presented only a few years afterwards, by Goldwasser and Micali [GM82]. The two definitions presented in Section 5.2 originate in [GM82], where it was shown that ciphertext-indistinguishability implies semantic security. The converse direction is due to [MRS].

Regarding the seminal paper of Goldwasser and Micali [GM82], a few additional comments are due. Arguably, this paper is the basis of the entire rigorous approach to cryptography (presented in the current book). The paper's title ("Probabilistic Encryption") is due to the author's realization that public-key encryption schemes in which the encryption algorithm is deterministic cannot be secure in the sense defined in their paper. Indeed, this led the authors to (explicitly) introduce and justify the paradigm of "randomizing the plaintext" as part of the encryption process. Technically speaking, the paper only presents security definitions for public-key encryption schemes, and furthermore some of these definitions are syntactically different from the ones we have presented above (yet, all these definitions are equivalent). Finaly, the term "ciphertext-indistinguishability" used here replaces the (generic) term "polynomial-security" used in [GM82]. Some of our modifications have already appeared in [G89a].

The first construction of a secure encryption scheme based on a simple complexity assumption was given by Goldwasser and Micali [GM82]. Specifically, they constructed a public-key encryption scheme assuming that deciding Quadratic Residiousity modulo composite numbers is intractable. The condition was weaken by Yao [Y82] who prove that any trapdoor permutation will do. The efficient scheme presented in Figure XXX is due to Blum and Goldwasser [BlGw]. The security is based on the fact that the least significant bit of the modular squaring function is a hard-core predicate, provided that factoring is intractable, a result mostly due to [ACGS].

For decades, it has been common practice to use "pseudorandom generators" in the design of stream ciphers. As pointed out by Blum and Micali [BM84], this practice is sound *provided* that one uses pseudorandom generators (as defined in Chapter [pseudo.chap]). The construction of private-key encryption schemes based on pseudorandom functions is due to [GGM84b].

Author's Note: *From this point on – a mess...*

CREDITS for CONS [GM82] and [BlGw,ACGS],

CREDIT FOR — Public-key encryption schemes secure against Chosen Ciphertext Attacks can be constructed, assuming the existence of trapdoor permutations and utilizing non-interactive zero-knowledge proofs [NY90] (which can be constructed under this assumption [FLS]).

The study of *non-malleability* of the encryption schemes, was initiated in [DDN]. Non-malleable public-key encryption schemes are known to exist assuming the existence of trapdoor permutation [DDN].

5.5.2 Suggestion for Further Reading

For discussion of Non-Malleable Cryptography, which actually transcends the domain of encryption, see [DDN].

5.5.3 Open Problems

Author's Note: *upgrade CMA-encryption.*

5.5.4 Exercises

Exercise 1: *Encryption schemes with unbounded-length plaintext*: Suppose that the definition of semantic security is modified so that no bound is placed on the length of plaintexts. Prove that in such a case there exists no sematically secure public-key encryption scheme. (Hint: A plaintext of length exponential in the security parameter allows the adversary to find the decryption key by exhuastive search.)

Exercise 2: *Encryption scheme must leak information about the length of the plaintext*: Suppose that the definition of semantic security is modified so that the algorithms are not given the length of the plaintext. Prove that in such a case there exists no sematically secure encryption scheme.

> **Guideline:** First show that for some polynomial p, $|E(1^n)| < p(n)$, whereas for some $x \in \{0,1\}^{p(n)}$ we have $\mathsf{Pr}(|E(x)| < p(n)) < 1/2$.

Exercise 3: *Deterministic encryption schemes*: Prove that in order to be sematically secure a public-key encryption scheme must have a probabilistic encryption algorithm. (Hint: Otherwise, one can distinguish the encryptions of two candidate plaintexts by computing the unique ciphertext for each of them.)

Exercise 4: Prove that the following definition, in which we use probabilistic polynomial-time algorithms with auxiliary inputs, is equivalent to Definition 5.2.1.

> For every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm B, so that for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \mapsto \{0,1\}^*$, every polynomial $p(\cdot)$, all sufficiently large n and every $z \in \{0,1\}^{p(n)}$,
>
> $$\begin{aligned} &\mathsf{Pr}\left(A(z, E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) \\ &\quad < \mathsf{Pr}\left(B(z, 1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)} \end{aligned}$$

Same for public-key encryption.

> **Guideline:** The alternative view of non-uniformity, discussed in Section 1.3, is useful here. That is, we can view a circuit family as a sequence of advices given to a universal machine. Thus, the original definition states that advices for a machine which gets the ciphertext can be efficiently transformed into advices for a machine which does not get the ciphertext. However, we can incorporate the transformation program into the second

universal algorithm, and so the advices are identical for both machines (and can be viewed as the auxiliary string z in the new formulation). Thus, the original definition is implied by the new definition. To close the gap between the two definitions, one only needs to observe that it suffices to consider one fixed universal machine, A, in the new definition (as any adversarial strategy can be coded in the auxiliary input to this universal machine).

Exercise 5: Prove that a sematically-secure (private-key) encryption scheme satisfies the same requirements with respect to randomized circuits. That is, there exists a polynomail-time transformation, T, so that for every polynomial-size randomized circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \mapsto \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

$$\mathsf{Pr}\left(C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right) < \mathsf{Pr}\left(C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right) + \frac{1}{p(n)}$$

where $C'_n \stackrel{\text{def}}{=} T(C_n)$ is the circuit produced by T on input C_n. Same for public-key encryption.

Guideline: Given a randomized family $\{C_n\}$ as above, consider all possible families of deterministic circuits derived by fixing a sequence of coins for each C_n. Note that you should provide one family of randomized circuits, $\{C'_n\}$, to match the randomized family $\{C_n\}$. The alternative formulation of Exercise 4 is useful here (as one may incorporate and extract the coin-sequence in the auxiliary input).

Exercise 6: Prove that Definition 5.2.3 remains unchanged when restricting the string z to be empty. (Same for Definition 5.2.4.) (Hint: incorporate z in the circuit C_n.)

Exercise 7: *Equivalence of the security definitions in the public-key model*: Prove that a public-key encryption scheme is semantically secure if and only if it has indistinguishable encryptions.

Exercise 8: *Another equivalennt definition of security*: Prove that an encryption scheme, (G, E, D), is (semantically) secure (in the private-key model) if and only if the following holds.

There exists a polynomail-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, the following two ensembles are computationally indistribguishable.

1. $\{C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|})\}_{n\in\mathbb{N}}$.
2. $\{C'_n(1^{|X_n|})\}_{n\in\mathbb{N}}$.

Formulate and prove an analogous claim for the public-key model.

Exercise 9: *Hiding partial information about the length of the plaintext*: Using an arbitrary block cipher, construct an encryption scheme which

1. Hides the length of the plaintext upto a factor of 2.
2. Hides the length of the plaintext upto an additive term of n.

Prove that the resulting encryption scheme inherents the security of the original block-cipher.

(Hint: Just use an adequate padding convention, making sure that it always yields correct decoding.)

Exercise 10: *Known plaintext attacks*: Loosely speaking, in a known palintext attack on a private-key (resp., public-key) encryption scheme the adversary is given some plaintext/ciphertext pairs in addition to some extra ciphertexts (without corresponding plaintexts). Semantic security in this setting means that whatever can be efficiently computed about the missing plaintexts, can be also efficiently computed given only the length of these plaintexts.

1. Provide formal definitions of security for private-key/public-key in both the single-message and multiple-message settings.
2. Prove that any secure public-key encryption scheme is also secure in the presence of known plaintext attack.
3. Prove that any private-key encryption scheme which is secure in the multiple-message setting is also secure in the presence of known plaintext attack.

Exercise 11: *Length parameters*: Assuming the existence of a secure public-key (resp., private-key) encryption scheme, prove the existence of such scheme in which the length of keys equal the security parameter. Show that the length of ciphertexts may be a fixed polynomial in the length of the plaintext.

Author's Note: *First draft written mainly in 1997.*

Chapter 6

Digital Signatures and Message Authentication

The difference between message authentication and digital signatures is analogous to the difference between private-key and public-key encryption schemes. In this chapter we define both type of schemes and the security problem associated to them. We then present several constructions. We show how to construct message authentication schemes using pseudorandom functions, and how to construct signature schemes using one-way permutations (which do not necessarily have a trapdoor).

```
%Plan
\input{sg-def}%%% Definitions of Unforgable Signatures
%................ and Message Authentication
\input{sg-aut}%%% Construction of Message Authentication
\input{sg-con1}%% Construction of Signatures by [NY]
%................ tools: one-time signature, aut-trees, one-way hashing
\input{sg-hash}%% * Collision-free hashing:
%................ def, construct by clawfree, applications (sign., etc.)
\input{sg-con2}%% * Alternative Construction of Signatures [EGM]
\input{sg-misc}%% As usual: History, Reading, Open, Exercises
```

Author's Note: *Temporary material from survey*

6.1 Signatures – Brief Summary from my Essay

Again, there are private-key and public-key versions both consisting of three efficient algorithms: *key generation*, *signing* and *verification*. (Private-key signature schemes are commonly referred to as *message authentication schemes* or *codes* (MAC).) The difference between the two types is again reflected in the definition of security. This difference yields different functionality (even more than in the case of encryption): Public-key signature schemes (hereafter referred to as signature schemes) may be used to produce signatures which are *universally verifiable* (given access to the public-key of the signer). Private-key signature schemes (hereafter referred to as message authentication schemes) are only used to authenticate messages sent among a small set of *mutually trusting* parties (since ability to verify signatures is linked to the ability to produce them). Put in other words, message authentication schemes are used to authenticate information sent between (typically two) parties, and the purpose is to convince *the receiver* that the information was indeed sent by the legitimate sender. In particular, message authentication schemes cannot convince *a third party* that the sender has indeed sent the information (rather than the receiver having generated it by itself). In contrast, public-key signatures can be used to convince third parties: A signature to a document is typically sent to a second party so that in the future this party may (by merely presenting the signed document) convince third parties that the document was indeed generated/sent/approved by the signer.

6.1.1 Definitions

We consider very powerful attacks on the signature scheme as well as a very liberal notion of breaking it. Specifically, the attacker is allowed to obtain signatures to any message of its choice. One may argue that in many applications such a general attack is not possible (as messages to be signed must have a specific format). Yet, our view is that it is impossible to define a general (i.e., application-independent) notion of admissible messages, and thus a general/robust definition of an attack seems to have to be formulated as suggested here. (Note that at worst, our approach is overly cautious.) Likewise, the adversary is said to be successful if it can produce a valid signature to ANY message for which it has not asked for a signature during its attack. Again, this defines the ability to form signatures to possibly "nonsensical" messages as a breaking of the scheme. Yet, again, we see no way to have a general (i.e., application-independent) notion of "meaningful" messages (so that only forging signatures to them will be consider a breaking of the scheme).

Definition 6.1.1 (unforgeable signatures [GMRi]):

- *A* chosen message attack *is a process which on input a verification-key can obtain signatures* (relative to the corresponding signing-key) *to messages of its choice.*
- *Such an attack is said to* succeeds (in existential forgery) *if it outputs a valid signature to a message for which it has* NOT *requested a signature during the attack.*
- *A signature scheme is* secure (or unforgeable) *if every* feasible *chosen message attack succeeds with at most negligible probability.*

We stress that *plain* RSA (alike plain versions of Rabin's scheme [R79] and DSS [DSS]) is not secure under the above definition. However, it may be secure if the message is "randomized" before RSA (or the other schemes) is applied (cf., [BRsign]). Thus, the randomization paradigm (see Section 5.3) seems pivotal here too.

6.1.2 Constructions

Message authentication schemes can be constructed using pseudorandom functions (see [GGM2] or the better constructions in [BKR,BGR,BCK]). However, as noted in [BCK2], an *extensive* usage of pseudorandom functions seem an overkill for achieving message authentication, and more efficient schemes may be obtained based on other cryptographic primitives. We mention two approaches:

1. *Fingerprinting* the message using a scheme which is *secure against forgery provided that the adversary does not have access to the scheme's outcome* (e.g., using Universal Hashing [CW]), and "*hiding*" the result using a *non-malleable* scheme (e.g., a private-key encryption or a pseudorandom function). (Non-malleability is not required in certain cases; see [WC].)

2. *Hashing* the message *using a collision-free scheme* (cf., [D87,D89]), and *authenticating* the result using a MAC which operates on (short) fixed-length strings [BCK2].

Three central paradigms in the construction of *signature schemes* are the "refreshing" of the "effective" signing-key, the usage of an "authentication tree" and the "hashing paradigm".

The refreshing paradigm [GMRi]: To demonstrate this paradigm, suppose we have a signature scheme which is robust against a "random message attack" (i.e., an attack in which the adversary only obtains signatures to randomly chosen messages). Further suppose that we have a *one-time* signature scheme (i.e., a signature scheme which is secure against an attack in which the adversary obtains a signature to a single message of its choice). Then, we can obtain a secure signature scheme as follows: When a new message is to be signed, we generate a new random signing-key for the one-time signature scheme, use it to sign the message, and sign the corresponding (one-time) verification-key using the fixed signing-key of the main signature scheme[1] (which is robust against a "random message attack") [EGM]. We note that one-time signature schemes (as utilized here) are easy to construct (see, for example [M87]).

The tree paradigm [M80,GMRi]: To demonstrate this paradigm, we show how to construct a general signature scheme using only a one-time signature scheme (alas one where an $2n$-bit string can be signed w.r.t an n-bit long verification-key). The idea is to use the initial singing-key (i.e., the one corresponding to the public verification-key) in order to sign/authenticate two new/random verification keys. The corresponding signing keys are

[1]Alternatively, one may generate the one-time key-pair and the signature to its verification-key ahead of time, leading to an "off-line/on-line" signature scheme [EGM].

used to sign/authenticate four new/random verification keys (two per a signing key), and so on. Stopping after d such steps, this process forms a binary tree with 2^d leaves where each leaf corresponds to an instance of the one-time signature scheme. The signing-keys at the leaves can be used to sign the actual messages, and the corresponding verification-keys may be authenticated using the path from the root. Pseudorandom functions may be used to eliminate the need to store the values of intermediate vertices used in previous signatures [G86]. Employing this paradigm and assuming that the RSA function is infeasible to invert, one obtains a secure signature scheme [GMRi,G86] in which the i^{th} message can be signed/verified in time $2\log_2 i$ slower than plain RSA. Using a tree of large fan-in and assuming that RSA is infeasible to invert, one may obtain a secure signature scheme [DN] which for reasonable parameters is only 5 times slower than plain RSA (alas uses a much bigger key).[2] We stress that plain RSA is not a secure signature scheme, whereas the security of its randomized version (mentioned above) is not known to be reducible to the assumption that RSA is hard to invert.

The hashing paradigm: A common practice is to sign real documents via a two stage process: First the document is hashed into a (relatively) short bit string, and next the basic signature scheme is applied to the resulting string. We note that this heuristic becomes sound provided the hashing function is *collision-free* (as defined in [D87]). Collision-free functions can be constructed assuming the intractability of factoring [D87]. One may indeed postulate that certain off-the-shelf products (as MD5 or SHA) are collision-free, but such assumptions need to be tested (and indeed may turn out false). We stress that using a hashing scheme in the above two-stage process without evaluating whether it is collision-free is a very dangerous practice.

A useful variant on the above paradigm is the use of *Universal One-Way Hash Functions* (as defined in [NY89]), rather than the collision-free hashing used above. In such a case a new hash function is selected per each application of the scheme, and the basic signature scheme is applied to both the (succinct) description of the hash function and to the resulting (hashed) string. (In contrast, when using a collision-free hashing function, the same function – the description of which is part of the signer's public-key – is used in all applications.) The advantage of using Universal One-Way Hash Functions is that their security requirement seems weaker than the collision-free condition (e.g., the former may be constructed using any one-way function [R90], whereas this is not known for the latter).

A plausibility result: By [NY89,R90] signature schemes exist if and only if one-way functions exist. Unlike the constructions of signature schemes described above, the known construction of signature schemes from *arbitrary* one-way functions has no practical significance [R90]. It is indeed an important open problem to provide an alternative construction

[2] This figure refers to signing up-to 1,000,000,000 messages. The scheme requires a universal set of system parameters consisting of 1000–2000 integers of the size of the moduli. We believe that in *some* applications the storage/time trade-off provided by [DN] may be preferred over [GMRi,G86].

which may be practical and still utilize an *arbitrary* one-way function.

6.1.3 Some Suggestions for Further Reading

For a definitional treatment of *signature schemes* the reader is referred to [GMRi] and [P]. Easy to understand constructions appear in [BeM,EGM,DN]. Variants on the basic model are discussed in [P] and in [C82,JLO]. For discussion of *message authentication schemes* (MACs) the reader in referred to [BCK2].

[BCK] M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions Revisited: The Cascade Construction and its Concrete Security. In *37th IEEE Symposium on Foundations of Computer Science*, pages 514–523, 1996.

[BCK2] M. Bellare, R. Canetti and H. Krawczyk. Keying Hash Functions for Message Authentication. In *Crypto96*, Springer Lecture Notes in Computer Science (Vol. 1109), pages 1–15.

[BGR] M. Bellare, R. Guerin and P. Rogaway. XOR MACs: New Methods for Message Authentication using Finite Pseudorandom Functions. In *Crypto95*, Springer-Verlag Lecture Notes in Computer Science (Vol. 963), pages 15–28.

[BKR] M. Bellare, J. Kilian and P. Rogaway. The Security of Cipher Block Chaining. In *Crypto94*, Springer-Verlag Lecture Notes in Computer Science (Vol. 839), pages 341–358.

[BRsign] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *EuroCrypt96*, Springer Lecture Notes in Computer Science (Vol. 1070).

[CW] L. Carter and M. Wegman. Universal Hash Functions. *Journal of Computer and System Science*, Vol. 18, 1979, pages 143–154.

[C82] D. Chaum. Blind Signatures for Untraceable Payments. In *Crypto82*, Plenum Press, pages 199–203, 1983.

[D87] I. Damgård. Collision Free Hash Functions and Public Key Signature Schemes. In *EuroCrypt87*, Springer-Verlag, Lecture Notes in Computer Science (Vol. 304), pages 203–216.

[D89] I. Damgård. A Design Principle for Hash Functions. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 416–427.

[DN] C. Dwork, and M. Naor. An Efficient Existentially Unforgeable Signature Scheme and its Application. To appear in *Journal of Cryptology*. Preliminary version in *Crypto94*.

[EGM] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital signatures. *Journal of Cryptology*, Vol. 9, 1996, pages 35–67.

[G86] O. Goldreich. Two Remarks Concerning the GMR Signature Scheme. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 104–110, 1987.

[GGM2] O. Goldreich, S. Goldwasser, and S. Micali. On the Cryptographic Applications of Random Functions. In *Crypto84*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 276–288, 1985.

[GMRi] S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, April 1988, pages 281–308.

[JLO] A. Juels, M. Luby and R. Ostrovsky. Security of Blind Digital Signatures. In *Crypto97*, Springer Lecture Notes in Computer Science (Vol. 1???).

[M80] R.C. Merkle. Protocols for public key cryptosystems. In *Proc. of the 1980 Symposium on Security and Privacy.*

[M87] R.C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), 1987, pages 369-378.

[M89] R.C. Merkle. A Certified Digital Signature Scheme. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 218–238.

[DSS] National Institute for Standards and Technology. Digital Signature Standard (DSS), *Federal Register*, Vol. 56, No. 169, August 1991.

[NY89] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Application. *21st ACM Symposium on the Theory of Computing*, 1989, pages 33–43.

[P] B. Pfitzmann. *Digital Signature Schemes* (*General Framework and Fail-Stop Signatures*). Springer Lecture Notes in Computer Science (Vol. 1100), 1996.

[R77] M.O. Rabin. Digitalized Signatures. In *Foundations of Secure Computation* (R.A. DeMillo et. al. eds.), Academic Press, 1977.

[R79] M.O. Rabin. Digitalized Signatures and Public Key Functions as Intractable as Factoring. MIT/LCS/TR-212, 1979.

[R90] J. Rompel. One-way Functions are Necessary and Sufficient for Secure Signatures. In *22nd ACM Symposium on the Theory of Computing*, 1990, pages 387–394.

[WC] M. Wegman and L. Carter. New Hash Functions and their Use in Authentication and Set Equality. *Journal of Computer and System Science*, Vol. 22, 1981, pages 265–279.

Chapter 7

Cryptographic Protocols

> Author's Note: *This chapter is a serious obstacle to any future attempt of completing this book.*

```
%Plan
\input{pt-motiv}% Motivation (Examples: voting, OT)
\input{pt-def}%%% Definition (of a protocol problem)
\input{pt-two}%%% Construction of two-party protocols
\input{pt-many}%% Construction of multi-party protocols
\input{pt-misc}%% As usual: History, Reading, Open, Exercises
```

> Author's Note: *Temporary material from survey*

7.1 Cryptographic Protocols – Brief Summary from my Essay

A general framework for casting cryptographic (protocol) problems consists of specifying a random process which maps n inputs to n outputs. The inputs to the process are to be thought of as local inputs of n parties, and the n outputs are their corresponding local outputs. The random process describes the desired functionality. That is, if the n parties were to trust each other (or trust some outside party), then they could each send their local input to the trusted party, who would compute the outcome of the process and send each party the corresponding output. The question addressed in this section is to what extent can this trusted party be "simulated" by the mutually distrustful parties themselves.

7.1.1 Definitions

For simplicity we consider the special case where the specified process is deterministic and the n outputs are identical. That is, we consider an arbitrary n-ary function and n parties which wish to obtain the value of the function on their n corresponding inputs. Each party wishes to obtain the correct value of the function and prevent any other party from gaining anything else (i.e., anything beyond the value of the function and what is implied by it).

We first observe that (one thing which is unavoidable is that) each party may change its local input before entering the protocol. However, this is unavoidable also when the parties utilize a trusted party. In general, the basic paradigm underlying the definitions of *secure multi-party computations* amounts to saying that situations which may occur in the real protocol, can be simulated in the ideal model (where the parties may employ a trusted party). Thus, the "effective malfunctioning" of parties in secure protocols is restricted to what is postulated in the corresponding ideal model. The specific definitions differ in the specific restrictions and/or requirements placed on the parties in the real computation. This is typically reflected in the definition of the corresponding ideal model – see examples below.

An example – computations with honest majority: Here we consider an ideal model in which any minority group (of the parties) may collude as follows. Firstly this minority shares its original inputs and decided together on replaced inputs[1] to be sent to the trusted party. (The other parties send their respective original inputs to the trusted party.) When the trusted party returns the output, each majority player outputs it locally, whereas the colluding minority may compute outputs based on all they know (i.e., the output and all the local inputs of these parties). A *secure multi-party computation with honest majority* is required to simulate this ideal model. That is, the effect of any feasible adversary which controls a minority of the players in the actual protocol, can be essentially simulated by a (different) feasible adversary which controls the corresponding players in the ideal model. This means that in a secure protocol the effect of each minority group is "essentially restricted" to replacing its own local inputs (independently of the local inputs of the majority players) before the protocol starts, and replacing its own local outputs (depending only on its local inputs and outputs) after the protocol terminates. (We stress that in the real execution the minority players do obtain additional pieces of information; yet in a secure protocol they gain nothing from these additional pieces of information.)

Secure protocols according to the above definition may even tolerate a situation where a minority of the parties aborts the execution. An aborted party (in the real protocol) is simulated by a party (in the ideal model) which aborts the execution either before supplying its input to the trusted party (in which case a default input is used) or after supplying its input. In either case, the majority players (in the real protocol) are able to compute the output although a minority aborted the execution. This cannot be expected to happen when there is no honest majority (e.g., in a two-party computation) [C86].

[1] Such replacement may be avoided if the local inputs of parties are verifiable by the other parties. In such a case, a party (in the ideal model) has the choice of either joining the execution of the protocol with its correct local input or not join the execution at all (but it cannot join with a replaced local input). Secure protocols simulating this ideal model can be constructed as well.

Another example – two-party computations: In light of the above, we consider an ideal model where each of the two parties may "shut-down" the trusted (third) party at any point in time. In particular, this may happen after the trusted party has supplied the outcome of the computation to one party but before it has supplied it to the second. A *secure multi-party computation allowing abort* is required to simulate this ideal model. That is, each party's "effective malfunctioning" in a secure protocol is restricted to supplying an initial input of its choice and aborting the computation at any point in time. We stress that, as above, the choice of the initial input of each party may NOT depend on the input of the other party.

7.1.2 Constructions

General plausibility results: Assuming the existence of trapdoor permutations, one may provide secure protocols for any two-party computation (allowing abort) [Y86] as well as for any multi-party computations with honest majority [GMW87]. Thus, a host of cryptographic problems are solvable assuming the existence of trapdoor permutations. As stressed in the case of zero-knowledge proofs, we view these results as asserting that very wide classes of problems are solvable in principle. However, we do not recommend using the solutions derived by these general results in practice. For example, although Threshold Cryptography (cf., [DF89,Ge97]) is merely a special case of multi-party computation, it is indeed beneficial to focus on its specifics.

Analogous plausibility results were obtained in a variety of models. In particular, we mention secure computations in the private channels model [BGW,CCD] and in the presence of mobile adversaries [OY].

7.1.3 Some Suggestions for Further Reading

This area is both most complex and most lacking good expositions. Our own preference is to refer to [C95] for the definitions and to [G89] for the constructions.

[BGW] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[C95] R. Canetti. *Studies in Secure Multi-Party Computation and Applications*. Ph.D. Thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, June 1995.
Available from from http://theory.lcs.mit.edu/˜tcryptol/BOOKS/ran-phd.html.

[CCD] D. Chaum, C. Crépeau and I. Damgård. Multi-party unconditionally Secure Protocols. In *20th ACM Symposium on the Theory of Computing*, pages 11–19, 1988.

[C86] R. Cleve. Limits on the Security of Coin Flips when Half the Processors are Faulty. In *18th ACM Symposium on the Theory of Computing*, pages 364–369, 1986.

[DF89] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In *Crypto89*, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pages 307–315.

[Ge97] P.S. Gemmell. An Introduction to Threshold Cryptography. In *CryptoBytes*, RSA Lab., Vol. 2, No. 3, 1997.

[G89] O. Goldreich. *Lecture Notes on Encryption, Signatures and Cryptographic Protocol.* Spring 1989. Available from `http://theory.lcs.mit.edu/~oded/ln89.html`

[GMW87] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing,* pages 218–229, 1987.

[OY] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In *10th ACM Symposium on Principles of Distributed Computing*, pages 51–59, 1991.

[Y86] A.C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

Part III

Beyond the Basics

Chapter 8

* New Frontiers

Where is the area going?

That's always hard to predict,
but following are some recent and not so recent developments.

```
%Plan
\input{fr-sys}%%% Cryptographic Infrastructure Problems (key-mgmt, replay, etc.)
\input{fr-eff}%%% more stress on efficiency (from a theory perspective!)
\input{fr-rom}%%% the Random Oracle Model (e.g., [BR]).
\input{fr-dyn}%%% Migrating adversaries (in multi-party protocls)
\input{fr-incr}%% Incremental Cryptography [BGG]
\input{fr-traf}%% Trafic Analysis [RS]
\input{fr-soft}%% Software Protection [G,O] (that's not really new...)
```

Chapter 9

* The Effect of Cryptography on Complexity Theory

Cryptography had a fundamental effect on the development of complexity theory. Notions such as computational indistinguishability, pseudorandomness and interactive proofs were first introduced and developed with a cryptographic motivation. However, these notions turned out to influence the development of complexity theory as well, and were further developed within this broader theory. In this chapter we survey some of these developments which have their roots in cryptography and yet provide results which are no longer (directly) relevant to cryptography.

> Author's Note: *Until the time this chapter is written, the reader is referred to my homepage for surveys of Probabilistic Proof Systems.*

```
%Plan
\input{eff-rand}% Deterministic Simulation of Randomized Complexity Classes
%................ (simulations of random-AC0, BPP and RL)
```

9.1 The power of Interactive Proofs

9.2 Probabilistically Checkable Proofs

Theorem 9.2.1 *the pcp characterization of NP*

```
\input{eff-rsr}%% Random Self-Reducibility (DLP/QR, Permanent)
\input{eff-learn}% Learning
\input{eff-misc}% (as usual)
```

Chapter 10

* Related Topics

In this chapter we survey several unrelated topics which are related to cryptography in some way. For example, a natural problem which arises in light of the excessive use of randomness is how to extract almost perfect randomness from sources of weak randomness.

```
%Plan
\input{tp-sour}%% Weak sources of randomness
\input{tp-byz}%%% Byzantine Agreement
\input{tp-check}% Program Checking and Statistical Tests
\input{tp-misc}%% As usual: History, Reading, Open, Exercises
```