Fragments of a chapter on Encryption Schemes

(Extracts from Foundations of Cryptography – in preparation)

(revised, second posted version)

Oded Goldreich

Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, ISRAEL.

June 9, 2001

to Dana

Preface

The current manuscript consists of fragments of a chapter on encryption schemes, which is suppose to be Chapter 5 of the three-volume work *Foundations of Cryptography*. These fragments provide a draft of the first three sections of this chapter, covering the basic setting, definitions and constructions. Also included is a plan of the fourth section (i.e., *beyond eavesdropping security*), and fragments for the Miscellaneous section of this chapter. This manuscript subsumes a previous version posted in Dec. 1999.

The bigger picture. The current manuscript consists of fragments of a chapter on encryption schemes, which is suppose to constitute Chapter 5 of the three-part work *Foundations of Cryptography* (see Figure 0.1). The three parts of this work are *Basic Tools*, *Basic Applications*, and *Beyond the Basics*. The first part (containing Chapters 1–4) has been published by Cambridge University Press (in June 2001). The second part, consists of Chapters 5–7 (regarding Encryptioni Schemes, Signatures Schemes, and General Cryptographic Protocols, respectively). We hope to publish the second part with Cambridge University Press within a few years.

Part 1: Introduction and Basic Tools
- Chapter 1: Introduction
- Chapter 2: Computational Difficulty (One-Way Functions)
- Chapter 3: Pseudorandom Generators
- Chapter 4: Zero-Knowledge Proofs

Part 2: Basic Applications
- Chapter 5: Encryption Schemes
- Chapter 6: Signature Schemes
- Chapter 7: General Cryptographic Protocols

Part 3: Beyond the Basics
- . . .

Figure 0.1: Organization of this work

The partition of the work into three parts is a logical one. Furthermore, it offers the advantage of publishing the first part without waiting for the completion of the other parts. Similarly, we hope to complete the second part within a couple of years, and publish it without waiting for the third part.

Prerequisites. The most relevant background for this text is provided by basic knowledge of algorithms (including randomized ones), computability and elementary probability theory. Background on (computational) number theory, which is required for specific implementations of certain constructs, is not really required here.

Using this text. The text is intended as part of a work that is aimed to serve both as a textbook and a reference text. That is, it is aimed at serving both the beginner and the expert. In order to achieve this aim, the presentation of the basic material is very detailed so to allow a typical CS-undergraduate to follow it. An advanced student (and certainly an expert) will find the pace (in these parts) way too slow. However, an attempt was made to allow the latter reader to easily skip details obvious to him/her. In particular, proofs are typically presented in a modular way. We start with a high-level sketch of the main ideas, and only later pass to the technical details. Passage from high-level descriptions to lower level details is typically marked by phrases such as *details follow*.

> In a few places, we provide straightforward but tedious details in indented paragraphs as this one. In some other (even fewer) places such paragraphs provide technical proofs of claims that are of marginal relevance to the topic of the book.

More advanced material is typically presented at a faster pace and with less details. Thus, we hope that the attempt to satisfy a wide range of readers will not harm any of them.

Teaching. The material presented in the full (three-volume) work is, on one hand, way beyond what one may want to cover in a course, and on the other hand falls very short of what one may want to know about Cryptography in general. To assist these conflicting needs we make a distinction between *basic* and *advanced* material, and provide suggestions for further reading (in the last section of each chapter). In particular, sections, subsections, and subsubsections marked by an asterisk (*) are intended for advanced reading.

Table of Contets

Part II

Basic Applications

Chapter 5

Encryption Schemes

Upto the 1970's, Cryptography was understood as the art of building encryption schemes; that is, the art of constructing schemes allowing secret data exchange over insecure channels. Since the 1970's, other tasks (e.g., signature schemes) have been recognized as falling within the domain of Cryptography (and even as being 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. It turns out that using randomness throughout the encryption process (i.e., not only at the key-generation phase) is essential to security. We present some basic constructions of secure (private-key and public-key) encryption schemes. Finally, we discuss "dynamic" notions of security culminating in robustness against chosen ciphertext attacks.

> Author's Note: *Currently the write-up contains only a rough draft for the first 3 sections of this chapter. Furthermore, this write-up was NOT carefully proofread, and may contain various (hopefully minor) errors.*

Teaching Tip: We assume that the reader is familiar with the material in previous chapters (and specifically with Sections 2.2, 2.4, 2.5, 3.2–3.4, and 3.6). This familiarity is important not only because we use some of the notions and results presented in these sections, but rather because we use similar proof techniques (and do it while assuming that this is *not* the reader's first encounter with these techniques).

5.1 The Basic Setting

Loosely speaking, encryption schemes are supposed to enable private communication between parties that communicate over an insecure channel. Thus, the basic setting consists of a *sender*, a *receiver*, and an *insecure channel* that 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 that 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 retrieve the plaintext from the ciphertext, but the adversary cannot do so. Clearly, something must distinguish the receiver (who is able to retrieve the plaintext from the corresponding ciphertext) from the adversary (who cannot do so). Specifically, the receiver know something that 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 hypothesis that the adversary does not know the keys.[1] 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).

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 description algorithm must employ a *key* that 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 that 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.

5.1.1 Private-Key versus Public-Key Schemes

A fundamental distinction between encryption schemes refers to the relation between the two keys (mentioned above). The simpler (and older) notion assumes that the encryption key equals the decryption key. Such schemes are called

[1] In fact, in many cases, the legitimate interest may be served best by publicizing the scheme itself. In our opinion, this is the best way to obtain an (unbiased) expert evaluation of the security of the scheme.

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 that is assumed to be secure. A crucial point is that the key is generated independently 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 tomorrow, then they can use the private channel today to exchange a secret key that 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 securely 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 The Syntax 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 every $\alpha \in \{0,1\}^*$, algorithms E* (encryption) *and D* (decryption) *satisfy*

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

where the probability is taken 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 $G(1^n)$ constitutes 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.*

We stress that Definition 5.1.1 says nothing about security, and so trivial (insecure) algorithms may satisfy it (e.g., $E(e,\alpha) \stackrel{\rm def}{=} \alpha$ and $D(d,\beta) \stackrel{\rm def}{=} \beta$). Furthermore, 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 $e = d$).

We stress that the above definition requires the scheme to operate for every plaintext, and specifically for plaintext of length exceeding the length of the encryption key. (This rules out the information theoretic secure "one-time pad" scheme mentioned above.)

Notation: In the rest of this text, we write $E_e(\alpha)$ instead of $E(e,\alpha)$ and $D_d(\beta)$ instead of $D(d,\beta)$. Sometimes, when 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 $G(1^n)$. That is, $G(1^n) = (G_1(1^n), G_2(1^n))$. Without loss of generality, we may assume that $|G_1(1^n)|$ and $|G_2(1^n)|$ are polynomially related to n, and that each of these integers can be efficiently computed from the other. (In fact, we may even assume that $|G_1(1^n)| = |G_2(1^n)| = n$; see Exercise 5.)

Comments: Definition 5.1.1 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., $\Pr[D_d(E_e(\alpha)) \neq \alpha] < 2^{-n}$). 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 each of the 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 $\ell : \mathbb{N} \rightarrow \mathbb{N}$ 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 plaintexts into blocks of length $\ell(n)$ and applying the restricted scheme separately to each block. For more details see Sections 5.2.4 and 5.3.2.

5.2 Definitions of Security

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 (which requires that the ciphertext yields no information regarding the plaintext). Loosely speaking, an encryption scheme is semantically secure if it is *infeasible* to learn anything about the plaintext from the ciphertext (i.e., impossibility is replaced by infeasibility). The second definition has a more technical flavor. 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 the analysis of cryptographic protocols that 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 that 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 also without the ciphertext. Thus, an adversary gains nothing by intercepting ciphertexts sent between communicating parties who use a semantically secure encryption scheme, since it could have obtained the same without intercepting these ciphertexts. Indeed, this formulation follows the simulation paradigm: "lack of gain" is captured by asserting that whatever is learned from the ciphertext can be learned within related complexity also without the ciphertext.

5.2.1.1 The actual definitions

To be somewhat more accurate, semantic security means that whatever can be efficiently computed from the ciphertext, can be efficiently computed when *given only the length of the plaintext*. Note that this formulation does not rule 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 3). We stress that other than information about the length of the plaintext, the ciphertext is required to yield nothing about the plaintext.

In the actual definitions, we consider only information regarding the plaintext (rather than anything) which can be obtained from the ciphertext. Furthermore, we restrict our attention to functions applied to the plaintext. We do so because of the intuitive appeal of this special case, and are comfortable doing so because this special case implies the general one (cf. Exercise 11). 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. In the actual definition, the information regarding the plaintext that the adversary tries to obtain is captured by the function f, whereas the a-priori partial information about the plaintext is captured by the function h. The above is required to hold for any distribution of plaintexts, captured by the probability ensemble $\{X_n\}_{n\in\mathbb{N}}$.

Security 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 2). 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 manifested in the definition of security. In the latter, the adversary (which is 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 for every proba-*

bilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' so that for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \mathrm{poly}(n)$, every pair of polynomially-bounded functions $f,h : \{0,1\}^* \rightarrow \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

$$\mathsf{Pr}\left[A(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right] \tag{5.1}$$

$$< \quad \mathsf{Pr}\left[A'(1^{|X_n|}, h(X_n)) = f(X_n)\right] + \frac{1}{p(n)} \tag{5.2}$$

(*The probability in the above terms is taken over X_n as well as over the internal coin tosses of algorithms G, E and either A or A'.*)

The function h provides both algorithms with partial information regarding the plaintext X_n. Furthermore, h also makes the definition implicitly non-uniform; see further discussion below. 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., algorithm A) that is given the ciphertext, can be matched, upto a negligible fraction, by the success probability of an efficient algorithm (i.e., algorithm A') that is not given the ciphertext at all.

Definition 5.2.1 refers to private-key encryption schemes. To derive a definition of security for public-key encryption schemes, the encryption-key (i.e., $G_1(1^n)$) should be given to the adversaries 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 for every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm A' such that for every $\{X_n\}_{n\in\mathbb{N}}$, f,h, $p(\cdot)$ and n as in Definition 5.2.1*

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

We comment that the encryption-key can be omitted from the input to A', since A' may generate it by itself.

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

The reader may note that a semantically-secure *public-key* encryption scheme cannot employ a deterministic encryption algorithm; that is, $E_e(x)$ must be a random variable rather than a fixed string. This is more evident with respect to the equivalent Definition 5.2.4 (below). See further discussion following Definition 5.2.4.

5.2.1.2 Further discussion of some definitional choices

We discuss several secondary issues regarding Definitions 5.2.1 and 5.2.2. The interested reader is also referred to Exercises 14 and 15 that present additional variants of the definition of semantic security.

Implicit non-uniformity of the definitions. The fact that h is not required to be computable, makes the above definitions non-uniform. This is the case because both algorithms are given $h(X_n)$ as auxiliary input, and this may account for arbitrary (polynomially-bounded) advise. For example, letting $h(x) = a_{|x|}$, means that both algorithms are supplied with (non-uniform) advice (as in one of the possible formulations of non-uniform polynomial-time; see Section 1.3.3). In general, the function h can code both information regarding its input and non-uniform advice depending on its input length (i.e., $h(x) = (h'(x), a_{|x|})$). Thus, the above definitions are equivalent to allowing A and A' be *related* families of non-uniform circuits, where by 'related' we mean that the circuits in the family $A' = \{A'_n\}_{n\in\mathbb{N}}$ can be efficiently computed from the corresponding circuits in the family $A = \{A_n\}_{n\in\mathbb{N}}$. For further discussion, see Exercise 7.

Lack of computational restrictions regarding the function f. We do not require that the function f is even computable. This seems strange at first glance, because (unlike the situation w.r.t h which codes a-priori information given to the algorithms) the algorithms are asked to guess the value of f (on a plaintext implicit in the ciphertext given only to A). However, as we shall see in the sequel (see also Exercise 11), 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 A can compute can be computed by A').

Other modifications of no impact. Actually, inclusion of a-priori information regarding the plaintext (captured by the function h) does not affect the definition of semantic security: Definition 5.2.1 remains intact if we restrict h to only depend on the length of the plaintext (and so only provide non-uniform advice). (This can be shown in various ways; e.g., see Exercise 12.) Also, the function f can be restricted to be a Boolean function having polynomial-size circuits, and the random variable X_n may be restricted to be very "dull" (e.g., have only two strings in its support): See proof of Theorem 5.2.5. On the other hand, Definition 5.2.1 implies stronger forms discussed in Exercises 11, 15 and 16.

5.2.2 Indistinguishability of Encryptions

The following technical interpretation 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 3.2.7. 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., $|x| = |y|$),

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

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

Note that the potential plaintexts to be distinguished can be incorporated into the circuit C_n. Thus, the circuit models both the adversary's strategy and its a-priori information: See Exercise 9.

Again, the security definition for public-key encryption schemes can be derived by adding the encryption-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 and y as in Definition 5.2.3*

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

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

Failure of deterministic encryption algorithms: A ciphertext-indistinguishable *public-key* encryption scheme cannot employ a deterministic encryption algorithm (i.e., $E_e(x)$ cannot be a fixed string). For a public-key encryption scheme with a *deterministic* encryption algorithm E, given an encryption-key e and a pair of candidate plaintexts (x, y), one can easily distinguish $E_e(x)$ from $E_e(y)$ (by merely applying E_e to x and comparing the result to the given ciphertext). In contrast, in case the encryption algorithm itself is randomized, the same plaintext can be encrypted in exponentially many different ways, under the same encryption key. Furthermore, the probability that applying E_e twice to the same message (while using independent randomization in E_e) results in the same ciphertext may be exponentially vanishing. (Indeed, as shown below, public-key encryption scheme having indistinguishable encryptions can be constructed based on any trapdoor permutations, and these schemes employ randomized encryption algorithms.)

5.2.3 Equivalence of the Security Definitions

The following theorem is stated and proven for private-key encryption schemes. A similar result holds for public-key encryption schemes (see Exercise 10).

Theorem 5.2.5 (equivalence of definitions – private-key): *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. Each proposition is 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 interpretation of security, in terms of ciphertext-indistinguishability, implies the natural notion of semantic security. Thus, the following proposition yields a methodology for designing semantically secure encryption schemes: design and prove your scheme to be ciphertext-indistinguishable, and conclude (by applying the proposition) that it is semantically secure. The opposite direction (of Theorem 5.2.5) establish the "completeness" of the latter methodology, and more generally assert that requiring an encryption scheme to be ciphertext-indistinguishable does not rule out schemes that are semantically secure.

Proposition 5.2.6 (useful direction – "indistinguishability" implies "security"): *Suppose that (G, E, D) is a ciphertext-indistinguishable private-key encryption scheme. Then (G, E, D) is semantically-secure. Furthermore, the simulating algorithm A' (which is used to establish semantic-security) captures the computation of a probabilistic polynomial-time oracle machine that is given oracle access to original adversary algorithm A.*

Proposition 5.2.7 (opposite direction – "security" implies "indistinguishability"): *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 satisfying the following four conditions:*

1. *the random variable X_n is uniformly distributed over a set containing two strings;*
2. *the value of h depends only on the length of its input (i.e., $h(x) = h'(|x|)$);*
3. *the function f is Boolean and is computable by a polynomial-size circuit;*
4. *the algorithm A is deterministic.*

In addition, no computational restrictions are placed on algorithm A' and it can be replaced by any function, which may depend on $\{X_n\}_{n\in\mathbb{N}}$, h, f and A.

Observe that the above four itemized conditions limit the scope of the four universal quantifiers in Definition 5.2.1, whereas the last sentence removes a restriction on the existential quantifier (i.e., removes the complexity bound on A') and allows the latter to depend on all universal quantifiers. Each of these modifications makes the resulting definition potentially weaker. Still, combining Propositions 5.2.7 and 5.2.6 it follows that a weak version of Definition 5.2.1 implies (an even stronger version than) the one stated in Definition 5.2.1.

5.2.3.1 Proof of Proposition 5.2.6.

Suppose that (G,E,D) has indistinguishable encryptions. We show that (G,E,D) is semantically secure by *constructing*, for every probabilistic polynomial-time algorithm A, a probabilistic polynomial-time algorithm A' such that the following holds: *for every $\{X_n\}_{n\in\mathbb{N}}$, f and h, algorithm A' guesses $f(X_n)$ from $(1^{|X_n|},h(X_n))$ essentially as good as A guesses $f(X_n)$ from $(E(X_n),1^{|X_n|},h(X_n))$.* Algorithm A' merely invokes A on input $(E(1^{|X_n|}),1^{|X_n|},h(X_n))$, and returns whatever A does. Intuitively, the indistinguishability of encryptions implies that A behaves as well when invoked by A' (and given a dummy encryption) as when given the encryption of X_n. Details follow.

Let A be an algorithm that tries to infer partial information (i.e., the value $f(X_n)$) from the encryption of the message X_n (when also given $1^{|X_n|}$ and a-priori information $h(X_n)$). Namely, on input $E(\alpha)$ and $(1^{|\alpha|},h(\alpha))$, algorithm A tries to guess $f(\alpha)$. We construct a new algorithm, A', that performs as well without getting the input $E(\alpha)$. The new algorithm consists of invoking A on input $E_{G_1(1^n)}(1^{|\alpha|})$ and $(1^{|\alpha|},h(\alpha))$, and outputting whatever A does. That is, on input $(1^{|\alpha|},h(\alpha))$, algorithm A' proceeds as follows:

1. A' invokes the key-generator G (on input 1^n), and obtains an encryption-key $e\leftarrow G_1(1^n)$.
2. A' invokes the encryption algorithm with key e and ("dummy") plaintext $1^{|\alpha|}$, obtaining a ciphertext $\beta\leftarrow E_e(1^{|\alpha|})$.
3. A' invokes A on input $(\beta,1^{|\alpha|},h(\alpha))$, and outputs whatever A does.

Observe that A' is described in terms of an oracle machine that makes a single oracle call to (any given) A, in addition to invoking the fixed algorithms G and E. Furthermore, the construction of A' does not depend on the functions h and f or on the distribution of messages to be encrypted (represented by the probability ensembles $\{X_n\}_{n\in\mathbb{N}}$). Thus, A' is probabilistic polynomial-time whenever A is probabilistic polynomial-time (and regardless of the complexity of h, f and $\{X_n\}_{n\in\mathbb{N}}$)

Indistinguishability of encryptions will be used to prove that A' performs essentially as well as A. Specifically, the proof will use a reducibility argument.

Claim 5.2.6.1: Let A' be as above. Then, for every $\{X_n\}_{n\in\mathbb{N}}$, f, h and p as in Definition 5.2.1, and all sufficiently large n's

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

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

$$\begin{aligned}&\Pr\left[A(E_{G_1(1^n)}(X_n),h(X_n))=f(X_n)\right]\\&\quad<\ \Pr\left[A(E_{G_1(1^n)}(1^{|X_n|}),h(X_n))=f(X_n)\right]+\frac{1}{p(n)}\end{aligned}$$

Intuitively, this follows by the indistinguishability of encryptions, by fixing a violating value of X_n and incorporating the corresponding values of $h(X_n)$ and $f(X_n)$ in a description of a circuit (which will distinguish an encryption of this value of X_n from an encryption of $1^{|X_n|}$). Details follow.

Assume, towards the contradiction that for some polynomial 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

$$\Delta(x) \stackrel{\rm def}{=} \left|\mathsf{Pr}\left[A(E_{G_1(1^n)}(x),h(x))=f(x)\right] - \mathsf{Pr}\left[A(E_{G_1(1^n)}(1^{|x|}),h(x))=f(x)\right]\right|$$

We use an averaging argument to single out a string x_n in the support of X_n such that $\Delta(x_n) \geq \Delta(X_n)$: That is, let $x_n \in \{0,1\}^{\mathrm{poly}(n)}$ be a string for which the value of $\Delta(\cdot)$ is maximum, and so $\Delta(x_n) > 1/p(n)$. Using this x_n, we introduce a circuit C_n, which incorporates the fixed values $f(x_n)$ and $h(x_n)$, and distinguishes the encryption of x_n from the encryption of $1^{|x_n|}$. The circuit C_n operates as follows. On input $\beta = E(\alpha)$, the circuit C_n invokes $A(\beta, h(x_n))$ and outputs 1 if and only if A outputs the value $f(x_n)$. Otherwise, C_n outputs 0.

The above circuit is indeed of polynomial-size because it merely incorporates strings of polynomial length (i.e., $f(x_n)$ and $h(x_n)$) and emulates a polynomial-time computation (i.e., of A). (Note that the circuit family $\{C_n\}$ is indeed non-uniform since its definition is based on a non-uniform selection of x_n's as well as on a hard-wiring of (possibly uncomputable) corresponding strings $h(x_n)$ and $f(x_n)$.) Clearly,

$$\mathsf{Pr}\left[C_n(E_{G_1(1^n)}(\alpha))=1\right] = \mathsf{Pr}\left[A(E_{G_1(1^n)}(\alpha),h(x_n))=f(x_n)\right] \qquad (5.3)$$

Combining Eq. (5.3) with the definition of $\Delta(x_n)$, we get

$$\begin{aligned} \left|\mathsf{Pr}\left[C_n(E_{G_1(1^n)}(x_n))=1\right] - \mathsf{Pr}\left[C_n(E_{G_1(1^n)}(1^{|x_n|}))=1\right]\right| &= \Delta(x_n) \\ &> \frac{1}{p(n)} \end{aligned}$$

This contradicts our hypothesis that E has indistinguishable encryptions, and the claim follows. □

We have just shown that A' performs essentially as well as A, and so Proposition 5.2.6 follows. ■

Comments: The fact that we deal with a non-uniform model of computation allows the above proof to proceed regardless of the complexity of f and h. All that our definition of C_n requires is the hardwiring of the values of f and h on a single string, and this can be done regardless of the complexity of f and h (provided that they are both polynomially-bounded).

When proving the public-key analogue of Proposition 5.2.6, algorithm A' is defined so that it generates an encryption of $1^{|\alpha|}$ relative to the encryption-key given to it as input (rather than relative to an encryption-key that it generates by itself, as done above). In addition, the distinguishing circuit considered in the analysis of the performance of A', obtains the encryption-key as part of its input and passes it to algorithm A (upon invoking it).

5.2.3.2 Proof of Proposition 5.2.7

Here the entire proof is by a reducibility argument. We 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 $\{C_n\}$, such that for infinitely many n's there exists $x_n, y_n \in \{0,1\}^{\mathrm{poly}(n)}$ so that

$$\left|\Pr\left[C_n(E_{G_1(1^n)}(x_n))=1\right] - \Pr\left[C_n(E_{G_1(1^n)}(y_n))=1\right]\right| > \frac{1}{p(n)} \tag{5.4}$$

Using this sequence of C_n's, x_n's and y_n's, we define $\{X_n\}_{n\in\mathbb{N}}$, f and h (referred to in Definition 5.2.1) as follows:

- The probability ensembles $\{X_n\}_{n\in\mathbb{N}}$ is defined such that X_n is uniformly distributed over $\{x_n, y_n\}$.
- The function $f:\{0,1\}^* \to \{0,1\}$ is defined such that $f(x_n) = 1$ and $f(y_n) = 0$, for every n. Note that $f(X_n) = 1$ with probability 1/2 and is 0 otherwise.
- The function h is defined such that $h(X_n)$ equals the description of the circuit C_n. Note that $h(X_n) = C_n$ with probability 1, and thus reveals no information on the value of X_n. (In the sequel, we write $h(X_n) = h'(n) = C_n$.)

(Note that X_n, f and h satisfy the restrictions stated in the furthermore-clause of the proposition.)

We will present a (deterministic) polynomial-time algorithm A that, given $C_n = h(X_n)$, 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 algorithm (regardless of its complexity) can guess $f(X_n)$ better than with probability 1/2 when only given $1^{|X_n|}$ (because given the constant values $1^{|X_n|}$ and $h(X_n)$, the value of $f(X_n)$ is uniformly distributed over $\{0,1\}$). Details follow.

Let us assume, without loss of generality, that for infinitely many n's

$$\Pr\left[C_n(E_{G_1(1^n)}(x_n))=1\right] > \Pr\left[C_n(E_{G_1(1^n)}(y_n))=1\right] + \frac{1}{p(n)} \tag{5.5}$$

Claim 5.2.7.1: There exists a (deterministic) polynomial-time algorithm A such that for infinitely many n's

$$\Pr\left[A(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n))=f(X_n)\right] > \frac{1}{2} + \frac{1}{2p(n)}$$

Proof: Algorithm A uses $C_n = h(X_n)$ in a straightforward manner: On input $\beta = E(\alpha)$ (where α is in the support of X_n) and $(1^{|\alpha|}, h(\alpha))$, algorithm A recovers $C_n = h'(n) = h(\alpha)$, invokes C_n on input β, and outputs 1 if C_n outputs 1 (otherwise, C_n outputs 0).[2]

It is left to analyze the success probability of A. Letting $m = |x_n| = |y_n|$, we have

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

where the inequality is due to Eq. (5.5). $\square$

In contrast, as observed above, no algorithm (regardless of its complexity) can guess $f(X_n)$ with success probability above $1/2$, when given only $1^{|X_n|}$ and $h(X_n)$. That is, we have

Fact 5.2.7.2: For every n and every algorithm A'

$$\Pr\left[A'(1^{|X_n|}, h(X_n)) = f(X_n)\right] \le \frac{1}{2} \tag{5.6}$$

Proof: Just observe that the output of A', on its constant input values $1^{|X_n|}$ and $h(X_n)$, is stochastically independent of the random variable $f(X_n)$, which in turn is uniformly distributed in $\{0,1\}$. Eq. (5.6) follows (and equality holds in case A' always outputs a value in $\{0,1\}$). $\square$

Combining Claim 5.2.7.1 and Fact 5.2.7.2, we reach a contradiction to the hypothesis that the scheme is semantically secure (even in the restricted sense mentioned in the furthermore-clause of the proposition). Thus, the proposition follows. ■

Comment: When proving the public-key analogue of Proposition 5.2.7, algorithm A is defined as above except that it passes the encryption-key, given to it as part of its input, to the circuit C_n. The rest of the proof remains intact.

[2] We comment that the '1' output by C_n is an indication that α is more likely to be x_n, whereas the output of A is a guess of $f(\alpha)$. This point may be better stressed by redefining f so that $f(x_n) \stackrel{\text{def}}{=} x_n$ and $f(x) = y_n$ if $x \neq x_n$, and having A output x_n if C_n outputs 1 and output y_n otherwise.

5.2.4 Multiple Messages

The above definitions only refer to the security of an encryption scheme that is used to encrypt a single plaintext (per a generated key). Since the plaintext may be longer than the key, these definitions are already non-trivial, and an encryption scheme satisfying them (even in the private-key model) implies the existence of one-way functions (see Exercise 1). Still, in many cases, it is desirable to encrypt many plaintexts using the same encryption key. Loosely speaking, an encryption scheme is secure in the multiple-message setting if analogous definitions (to the above) hold also when polynomially-many plaintexts are encrypted using the same encryption 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). We stress that this is not necessarily true *for the private-key model.*

5.2.4.1 Definitions

For a sequence of strings $\overline{x} = (x^{(1)}, ..., x^{(t)})$, we let $\overline{E}_e(\overline{x})$ denote the sequence of the t results that are obtained by applying the randomized process E_e to the t strings $x^{(1)}, ..., x^{(t)}$, respectively. That is, $\overline{E}_e(\overline{x}) = E_e(x^{(1)}), ..., E_e(x^{(t)})$. We stress that in each of these t invocations, the randomized process E_e utilizes independently chosen random coins. For sake of simplicity, we consider the encryption of (polynomially) many strings of the same (polynomial) length (rather than the encryption of strings of various lengths as discussed in Exercise 17).

Definition 5.2.8 (semantic security – multiple messages):

For private-key: *An encryption scheme, (G, E, D), is* semantically secure for multiple messages in the private-key model *if for every polynomial $t(\cdot)$ and every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm A' such that for every ensemble $\{\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})\}_{n\in\mathbb{N}}$, with $|X_n^{(i)}| = \text{poly}(n)$, every pair of functions $f, h : \{0,1\}^* \to \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n*

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

For public-key: *An encryption scheme, (G, E, D), is* semantically secure for multiple messages in the public-key model *if for $t(\cdot)$, A, A', $\{\overline{X}_n\}_{n\in\mathbb{N}}$, f, h, $p(\cdot)$ and n as above*

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

We stress that the elements of $\overline{X}_n$ are not necessarily independent; they may depend on one another. Note that the above definition also cover the case where the adversary obtains some of the plaintexts themselves. In this case it is still infeasible for him/her to obtain information about the missing plaintexts (see Exercise 18).

Definition 5.2.9 (indistinguishability of encryptions – multiple messages):

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

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

where $\bar{x} = (x_1,...,x_{t(n)})$ and $\bar{y} = (y_1,...,y_{t(n)})$.

For public-key: *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 $x_1,...,x_{t(n)},y_1,...,y_{t(n)}$ as above*

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

The equivalence of Definitions 5.2.8 and 5.2.9 can be established analogously to the proof of Theorem 5.2.5.

Theorem 5.2.10 (equivalence of definitions – multiple messages): *A private-key (resp., public-key) encryption scheme is semantically secure for multiple messages if and only if it has indistinguishable encryptions for multiple messages.*

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.

5.2.4.2 The effect on the public-key model

We first consider public-key encryption schemes.

Theorem 5.2.11 (single-message security implies multiple-message security): *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 current setting.

Suppose, towards the contradiction, that there exist a polynomial $t(\cdot)$, 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

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

where $\bar{x} = (x_1, ..., x_{t(n)})$ and $\bar{y} = (y_1, ..., y_{t(n)})$. Let us consider such a generic n and the corresponding sequences $x_1, ..., x_{t(n)}$ and $y_1, ..., y_{t(n)}$. We use a hybrid argument: define

$$\begin{aligned} \bar{h}^{(i)} &\stackrel{\text{def}}{=} (x_1, ..., x_i, y_{i+1}, ..., y_{t(n)}) \\ \text{and} \quad H_n^{(i)} &\stackrel{\text{def}}{=} (G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{h}^{(i)})) \end{aligned}$$

Since $H_n^{(0)} = (G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{y}))$ and $H_n^{(t(n))} = (G_1(1^n), \overline{E}_{G_1(1^n)}(\bar{x}))$, it follows that there exists an $i \in \{0, ..., t(n)-1\}$ so that

$$\left|\Pr\left[C_n(H_n^{(i)})=1\right] - \Pr\left[C_n(H_n^{(i+1)})=1\right]\right| > \frac{1}{t(n) \cdot p(n)} \tag{5.7}$$

We show that Eq. (5.7) yields a polynomial-size circuit that distinguishes the encryption of x_{i+1} from the encryption of y_{i+1}, and thus derive a contradiction to security in the single-message setting. Specifically, we construct a circuit D_n that incorporates the circuit C_n as well as the index i and the strings $x_1, ..., x_{i+1}, y_{i+1}, ..., y_{t(n)}$. On input an encryption-key e and (corresponding) ciphertext β, the circuit D_n operates as follows:

- For every $j \leq i$, the circuit D_n generates an encryption of x_j using the encryption key e. Similarly, for every $j \geq i+2$, the circuit D_n generates an encryption of y_j using the encryption key e.

  Let us denote the resulting ciphertexts by $\beta_1, ..., \beta_i, \beta_{i+2}, ..., \beta_{t(n)}$. That is, $\beta_j \leftarrow E_e(x_j)$ for $j \leq i$ and $\beta_j \leftarrow E_e(y_j)$ for $j \geq i+2$.

- Finally, D_n invokes C_n on input the encryption key e and the sequence of ciphertexts $\beta_1, ..., \beta_i, \beta, \beta_{i+2}, ..., \beta_{t(n)}$, and outputs whatever C_n does.

We stress that the construction of D_n relies in an essential way on the fact that the encryption-key is given to it as input.

We now turn to the analysis of the circuit D_n. Suppose that β is a (random) encryption of x_{i+1} with key e; that is, $\beta = E_e(x_{i+1})$. Then, $D_n(e, \beta) \equiv C_n(e, E_e(\bar{h}^{(i+1)})) = C_n(H_n^{(i+1)})$, where $X \equiv Y$ means that the random variables X and Y are identically distributed. Similarly, for $\beta = E_e(y_{i+1})$, we have $D_n(e, \beta) \equiv C_n(e, E_e(\bar{h}^{(i)})) = C_n(H_n^{(i)})$. Thus, by Eq. (5.7), we have

$$\begin{aligned} &\left|\Pr\left[D_n(G_1(1^n), E_{G_1(1^n)}(y_{i+1})=1\right]\right. \\ &\qquad \left. -\Pr\left[D_n(G_1(1^n), E_{G_1(1^n)}(x_{i+1})=1\right]\right| > \frac{1}{t(n) \cdot p(n)} \end{aligned}$$

in contradiction to our hypothesis that (G, E, D) is a ciphertext-indistinguishable public-key encryption scheme (in the single message sense). The theorem follows. ■

Discussion: The fact that we are in the public-key model is essential to the above proof. It allows the circuit D_n to form encryptions relative to the same encryption-key used in the ciphertext given to it. In fact, as stated above (and proven next), the analogous result does not hold in the private-key model.

5.2.4.3 The effect on the private-key model

In contrary to Theorem 5.2.11, *in the private-key model*, ciphertext-indistinguishability for a single message does NOT necessarily imply ciphertext-indistinguishability for multiple messages.

Proposition 5.2.12 *Suppose that there exist pseudorandom generators* (robust against polynomial-size circuits). *Then, there exists a private-key encryption scheme that 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, that 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 an analogous manner.

We first show that this encryption scheme satisfies Definition 5.2.3. Intuitively, this follow from the hypothesis that g is a pseudorandom generator and the fact that $x \oplus U_{|x|}$ is uniformly distributed over $\{0,1\}^{|x|}$. Specifically, suppose towards the contradiction that for some polynomial-size circuit family $\{C_n\}$, a polynomial p, and infinitely many n's

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

where U_n is uniformly distributed over $\{0,1\}^n$ and $|x| = |y| = m = \text{poly}(n)$. On the other hand,

$$\mathsf{Pr}[C_n(x \oplus U_m) = 1] = \mathsf{Pr}[C_n(y \oplus U_m) = 1]$$

Thus, without loss of generality

$$|\mathsf{Pr}[C_n(x \oplus g(U_n)) = 1] - \mathsf{Pr}[C_n(x \oplus U_m) = 1]| > \frac{1}{2 \cdot p(n)}$$

Incorporating x into the circuit C_n we obtain a circuit that distinguishes U_m from $g(U_n)$, in contradiction to our hypothesis (regarding the pseudorandomness of g).

Next, we observe that the above encryption scheme does not satisfy Definition 5.2.9. Specifically, given the ciphertexts of two plaintexts, one may easily retrieve the exclusive-or of the corresponding plaintexts. That is,

$$E_s(x_1) \oplus E_s(x_2) \;=\; (x_1 \oplus g(s)) \oplus (x_2 \oplus g(s)) \;=\; x_1 \oplus x_2$$

This clearly violates Definition 5.2.8 (e.g., consider $f(x_1, x_2) = x_1 \oplus x_2$) as well as Definition 5.2.9 (e.g., consider any $\bar{x} = (x_1, x_2)$ and $\bar{y} = (y_1, y_2)$ such that $x_1 \oplus x_2 \neq y_1 \oplus y_2$). Viewed in a different way, note that any plaintext-ciphertext pair yields a corresponding prefix of the pseudorandom sequence, and knowledge of this prefix violates the security of additional plaintexts. That is, given the encryption of a known plaintext x_1 along with the encryption of an unknown plaintext x_2, we can retrieve x_2. On input the ciphertexts β_1, β_2, knowing that the first plaintext is x_1, first retrieves the pseudorandom sequence (i.e., it is just $r \stackrel{\rm def}{=} \beta_1 \oplus x_1$), and next retrieves the second plaintext (i.e., by computing $\beta_2 \oplus r$). ■

Discussion: The single-message security of the above scheme was proven by considering an ideal version of the scheme in which the pseudorandom sequence is replaced by a truly random sequence. The latter scheme is secure in an information theoretic sense, and the security of the actual scheme followed by the indistinguishability of the two sequences. As we show in Section 5.3.1 (below), the above construction can be modified to yield a private-key "stream-cipher" that is secure for multiple message encryptions. All that is needed is to make sure that (as opposed to the construction above) the same portion of the pseudorandom sequence is never used twice.

5.2.5 * A uniform-complexity treatment

As stated at the beginning of this section, the non-uniform formulation was adopted here for sake of simplicity. In this subsection we sketch a uniform-complexity definitional treatment of security. We stress that by uniform or non-uniform complexity treatment of cryptographic primitives we merely refer to the modeling of the adversary. The honest (legitimate) parties are always modeled by uniform complexity classes (most commonly probabilistic polynomial-time).

The notion of *efficiently constructible ensembles*, defined in Section 3.2.3, is central to the uniform-complexity treatment. Recall that an ensemble, $X = \{X_n\}_{n\in\mathbb{N}}$, is said to be polynomial-time constructible if there exists a probabilistic polynomial time algorithm S so that for every n, the random variables $S(1^n)$ and X_n are identically distributed.

5.2.5.1 The definitions

We present only the definitions of security for multiple messages; the single-message variant can be easily obtained by setting the polynomial t (below) to be identically 1. Likewise, we present the public-key version, and the private-key analogous can be obtained by omitting $G_1(1^n)$ from the inputs to the various algorithms.

The uniformity of the following definitions is reflected in the complexity of the inputs given to the algorithms. Specifically, the plaintexts are taken from polynomial-time constructible ensembles and so are the auxiliary inputs given to the algorithms. For example, in the following definition we require the ensemble $\{\overline{X}_n\}$ to be polynomial-time constructible and the function h to be polynomial-time computable.

Definition 5.2.13 (semantic security – uniform-complexity version): *An encryption scheme, (G, E, D), is* uniformly semantically secure in the public-key model *if for every polynomial t, and every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that for every polynomial-time constructible ensemble $\{\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})\}_{n\in\mathbb{N}}$, with $|X_n^{(i)}| = \mathrm{poly}(n)$, every polynomial-time computable $h : \{0,1\}^* \to \{0,1\}^*$, every $f : \{0,1\}^* \to \{0,1\}^*$, every positive polynomial p and all sufficiently large n's*

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

where $\overline{E}_e(\overline{x}_n) \stackrel{\mathrm{def}}{=} E_e(x_n^{(1)}), ..., E_e(x_n^{(t(n))})$ (for $\overline{x} = (x_n^{(1)}, ..., x_n^{(t(n))})$) is as in Definition 5.2.8.

Again, we stress that $\overline{X}_n$ is a sequence of random variables, which may depend on one another. Also, the encryption-key $G_1(1^n)$ was omitted from the input of A' (since the latter may generate it by itself). We stress that even here (i.e., in the uniform complexity setting) no computational limitation are placed on the function f.

Definition 5.2.14 (indistinguishability of encryptions – uniform-complexity version): *An encryption scheme, (G, E, D), has* uniformly indistinguishable encryptions in the public-key model *if for every polynomial t, every probabilistic polynomial-time algorithm D', every polynomial-time constructible ensemble $\overline{T} \stackrel{\mathrm{def}}{=} \{\overline{T}_n = \overline{X}_n\overline{Y}_nZ_n\}_{n\in\mathbb{N}}$, with $\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})$, $\overline{Y}_n = (Y_n^{(1)}, ..., Y_n^{(t(n))})$, and $|X_n^{(i)}| = |Y_n^{(i)}| = \mathrm{poly}(n)$,*

$$\begin{aligned} &|\mathsf{Pr}\left[D'(Z_n, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{X}_n)) = 1\right] \\ &\quad - \mathsf{Pr}\left[D'(Z_n, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{Y}_n)) = 1\right] | < \frac{1}{p(n)} \end{aligned}$$

for every positive polynomial p and all sufficiently large n's.

The random variable Z_n captures a-priori information about the plaintexts for which encryptions should be distinguished. A special case of interest is when

$Z_n = \overline{X}_n \overline{Y}_n$. Uniformity is captured in the requirement that D' is a probabilistic polynomial-time algorithm (rather than a family of polynomial-size circuits) and that the ensemble $\{\overline{T}_n = \overline{X}_n \overline{Y}_n Z_n\}_{n\in\mathbb{N}}$ be polynomial-time constructible.

5.2.5.2 Equivalence of the multiple-message definitions

We prove the equivalence of the uniform-complexity definitions (presented above) for (multiple-message) security.

Theorem 5.2.15 (equivalence of definitions – uniform treatment): *A public-key encryption scheme satisfies Definition 5.2.13 if and only if it satisfies Definition 5.2.14. Furthermore, this holds even if Definition 5.2.14 is restricted to the special case where $Z_n = \overline{X}_n \overline{Y}_n$, and even if Definition 5.2.13 is restricted to the special case where f is polynomial-time computable.*

An analogous result holds for the private-key model. The important direction of the theorem holds also for the single-message version (this is quite obvious from the proof below). In the other direction, we seem to use the multiple-message version (of semantic security) in an essential way.

Proof Sketch: Again, we start with the more important direction; that is, assuming that (G, E, D) has (uniformly) indistinguishable encryptions in the special case where $Z_n = \overline{X}_n \overline{Y}_n$, we show that it is (uniformly) semantically secure. Our construction of algorithm A' is analogous to the construction used in the non-uniform treatment. Specifically, on input $(1^{|\overline{\alpha}_n|}, h(\overline{\alpha}_n))$, algorithm A' generates a random encryption of a dummy sequence of message (i.e., $1^{|\overline{\alpha}_n|}$), feeds it to A, and outputs whatever A does. That is,

$$A'(1^{|\overline{\alpha}_n|}, h(\overline{\alpha}_n)) = A(G(1^n), \overline{E}_{G(1^n)}(1^{|\overline{\alpha}_n|}), 1^{|\overline{\alpha}_n|}, h(\overline{\alpha}_n)) \tag{5.8}$$

As in the non-uniform case, the analysis of algorithm A' reduces to the following claim.

Claim 5.2.15.1: For every polynomial-time constructible ensemble $\{\overline{X}_n\}_{n\in\mathbb{N}}$, with $\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})$ and $|X_n^{(i)}| = \text{poly}(n)$, every polynomial-time computable h, every positive polynomial p and all sufficiently large n's

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

Proof sketch: Analogously to the non-uniform case, assuming towards the contradiction that the claim does not hold, yields an algorithm that distinguishes encryptions of $\overline{X}_n$ from encryptions of $\overline{Y}_n = 1^{|\overline{X}_n|}$, when getting auxiliary information $Z_n = \overline{X}_n \overline{Y}_n = \overline{X}_n 1^{|\overline{X}_n|}$. Thus, we derive contradiction to Definition 5.2.14 (even under the special case postulated in the theorem).

We note that the auxiliary information that is given to the distinguishing algorithm replaces the hard-wiring of auxiliary information that was used in the non-uniform case (and is not possible in the uniform complexity model). Specifically, rather than using a hard-wired value of h (at some non-uniformly fixed sequence), the distinguishing algorithm will use the auxiliary information $Z_n = \overline{X}_n 1^{|\overline{X}_n|}$ in order to compute $h(\overline{X}_n)$, which it will pass to A. Indeed, we rely on the hypothesis that h is efficiently computable.

The actual proof is quite simple in case the function f is also polynomial-time computable (which is not the case in general). In this special case, on input $(e, z, \overline{E}_e(\overline{\alpha}))$, where $z = (\overline{x}, 1^{|\overline{x}|})$ and $\overline{\alpha} \in \{\overline{x}, 1^{|\overline{x}|}\}$, the distinguishing algorithm computes $u = h(\overline{x})$ and $v = f(\overline{x})$, invokes A, and outputs 1 if and only if $A(e, \overline{E}_e(\overline{\alpha}), 1^{|\overline{x}|}, u) = v$.

> (We comment that in case $\overline{\alpha} = 1^{|\overline{x}|}$, we actually mean that $\overline{\alpha}$ is a sequence of $t(n)$ strings of the form $1^{\ell(n)}$, where t and ℓ are as in $\overline{x} = (x^{(1)}, ..., x^{(t(n))}) \in (\{0,1\}^{\ell(n)})^{t(n)}$.)

The proof becomes more involved in case f is not polynomial-time computable.[3] Again, the solution is in realizing that indistinguishability of encryption postulates a similar output profile in both cases, and in particular no value can occur non-negligibly more in one case than in the other. To clarify the point, we define $\Delta_v(\overline{x}_n)$ to be the difference between $\mathsf{Pr}[A(G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{x}_n), h(\overline{x}_n)) = v]$ and $\mathsf{Pr}[A(G_1(1^n), \overline{E}_{G_1(1^n)}(1^{|\overline{x}_n|}), h(\overline{x}_n)) = v]$. We know that $\mathsf{E}[\Delta_{f(\overline{X}_n)}(\overline{X}_n)] > 1/p(n)$, but given $\overline{x}_n$ we cannot evaluate $\Delta_{f(\overline{x}_n)}(\overline{x}_n)$, since we do not have $f(\overline{x}_n)$. Instead, we let $\Delta(\overline{x}_n) \stackrel{\rm def}{=} \max_v\{\Delta_v(\overline{x}_n)\}$, and observe that $\mathsf{E}[\Delta(\overline{X}_n)] \geq \mathsf{E}[\Delta_{f(\overline{X}_n)}(\overline{X}_n)] > 1/p(n)$. Furthermore, given $\overline{x}_n$ we can approximate $\Delta(\overline{x}_n)$ in polynomial-time, and can find (in polynomial-time) a value v such that $\Delta_v(\overline{x}_n) > \Delta(\overline{x}_n) - (1/2p(n))$, with probability at least $1 - 2^{-n}$.

> On approximating $\Delta(\overline{x}_n)$ etc.: By invoking algorithm A on $O(n \cdot p(n)^3)$ samples of the distributions $(G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{x}_n), h(\overline{x}_n))$ and $G_1(1^n), \overline{E}_{G_1(1^n)}(1^{|\overline{x}_n|}), h(\overline{x}_n))$, we obtain (implicitly) an approximation of all $\Delta_v(\overline{x}_n)$'s upto an additive deviation of $1/4p(n)$ (with error probability at most 2^{-n}). The approximation to $\Delta_v(\overline{x}_n)$, denoted $\widetilde{\Delta}_v(\overline{x}_n)$ is merely the difference between the fraction of samples (from both distributions) on which algorithm A returned 1. (Indeed, most $\Delta_v(\overline{x}_n)$'s are approximated by 0, but some $\Delta_v(\overline{x}_n)$'s may approximated by non-zero values.) We just output v for which the approximated value $\widetilde{\Delta}_v(\overline{x}_n)$ is largest. Thus, if for some v_0 it holds that $\Delta_{v_0}(\overline{x}_n) = \Delta(\overline{x}_n)$, then with probability at least $1-2^{-n}$ we output v such that
>
> $$\begin{aligned} \Delta_v(\overline{x}_n) &\geq \widetilde{\Delta}_v(\overline{x}_n) - (1/4p(n)) \\ &\geq \widetilde{\Delta}_{v_0}(\overline{x}_n) - (1/4p(n)) \\ &\geq \Delta_{v_0}(\overline{x}_n) - (1/4p(n)) - (1/4p(n)) \end{aligned}$$
>
> Thus, $\Delta_v(\overline{x}_n) \geq \Delta(\overline{x}_n) - (1/2p(n))$.

[3] Unlike in the non-uniform treatment, here we cannot hardwire values (such as the values of h and f on good sequences) into the algorithm D' (which is required to be uniform).

Thus, on input $(e, z, \overline{E}_e(\overline{\alpha}))$, where $z = (\overline{x}, 1^{|\overline{x}|})$, the new algorithm, denoted D', operates in two stages.

1. In the first stage, D' ignores the ciphertext $\overline{E}_e(\overline{\alpha})$. Using z, algorithm D' recovers $\overline{x}$, and computes $u = h(\overline{x})$. Using $\overline{x}$ and u, algorithm D' estimates $\Delta(\overline{x})$, and finds a v as above.

2. In the second stage (using u and v found in the first stage), algorithm D' invokes A, and outputs 1 if and only if $A(e, \overline{E}_e(\overline{\alpha}), 1^{|x|}, u) = v$.

Let $V(\overline{x})$ be the value found in the first stage of algorithm A (i.e., obliviously of the ciphertext $\overline{E}_e(\overline{\alpha})$). The reader can easily verify that

$$\begin{aligned}
&\left|\mathsf{Pr}\left[D'(G_1(1^n), Z_n, \overline{E}_{G_1(1^n)}(\overline{X}_n))=1\right] - \mathsf{Pr}\left[D'(G_1(1^n), Z_n, \overline{E}_{G_1(1^n)}(1^{\overline{X}_n}))=1\right]\right| \\
&\quad = \mathsf{E}\left[\Delta_{V(\overline{X}_n)}(\overline{X}_n)\right] \\
&\quad \geq (1-2^{-n}) \cdot \mathsf{E}\left[\Delta(\overline{X}_n) - \frac{1}{2p(n)}\right] - 2^{-n} \\
&\quad > \mathsf{E}\left[\Delta(\overline{X}_n)\right] - \frac{2}{3p(n)} > \frac{1}{3p(n)}
\end{aligned}$$

Thus, we have derived a probabilistic polynomial-time algorithm (i.e., D') that distinguishes encryptions of $\overline{X}_n$ from encryptions of $\overline{Y}_n = 1^{|\overline{X}_n|}$, when getting auxiliary information $Z_n = \overline{X}_n 1^{|\overline{X}_n|}$. By hypothesis $\{\overline{X}_n\}$ is polynomial-time constructible, and it follows that so is $\{\overline{X}_n \overline{Y}_n Z_n\}$ Thus, we derive contradiction to Definition 5.2.14 (even under the special case postulated in the theorem), and the claim follows. □

Having established the important direction, we now turn to the opposite one. That is, we assume that (G, E, D) is (uniformly) semantically secure and prove that it has (uniformly) indistinguishable encryptions. Again, the proof is by contradiction. Suppose, without loss of generality, that there exists a probabilistic polynomial-time algorithm D', a polynomial-time constructible ensemble $\overline{T} \stackrel{\text{def}}{=} \{\overline{T}_n = \overline{X}_n \overline{Y}_n Z_n\}_{n\in\mathbb{N}}$ (as in Definition 5.2.14), a positive polynomial p and infinitely many n's so that

$$\begin{aligned}
&\mathsf{Pr}\left[D'(Z_n, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{X}_n))=1\right] \\
&\qquad > \mathsf{Pr}\left[D'(Z_n, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{Y}_n))=1\right] \mid + \frac{1}{p(n)}
\end{aligned}$$

Let $t(n)$ and $\ell(n)$ be such that $\overline{X}_n$ (resp., $\overline{Y}_n$) consists of $t(n)$ strings, each of length $\ell(n)$. Suppose, without loss of generality, that $|Z_n| = m(n) \cdot \ell(n)$, and parse Z_n into $\overline{Z}_n = (Z_n^{(1)}, ..., Z_n^{(m(n))}) \in (\{0,1\}^{\ell(n)})^{m(n)}$ such that $Z_n = Z_n^{(1)} \cdots Z_n^{(m(n))}$. We define an auxiliary polynomial-time constructible ensemble $\overline{Q} \stackrel{\text{def}}{=} \{\overline{Q}_n\}_{n\in\mathbb{N}}$ so that

$$\overline{Q}_n = \begin{cases} 0^{\ell(n)} \overline{Z}_n \overline{X}_n \overline{Y}_n & \text{with probability } \frac{1}{2} \\ 1^{\ell(n)} \overline{Z}_n \overline{Y}_n \overline{X}_n & \text{with probability } \frac{1}{2} \end{cases}$$

That is, $\overline{Q}_n$ is a sequence of $1+m(n)+2t(n)$ strings, each of length $\ell(n)$, that contains $\overline{Z}_n\overline{X}_n\overline{Y}_n$ in addition to a bit (provided in the $\ell(n)$-bit long prefix) indicating whether the order of $\overline{X}_n$ and $\overline{Y}_n$ is switched or not. We define the function f so that to equal this "switch" indicator bit, and the function h to provide all information in $\overline{Q}_n$ except this switch bit. That is, we define f and h as follows:

- The function $f:\{0,1\}^*\to\{0,1\}$ is defined so that f returns the first bit of its input; that is, $f(\sigma^{\ell(n)}abc)=\sigma$, for $(a,b,c)\in(\{0,1\}^{l(n)})^{m(n)+2t(n)}$.
- The function $h:\{0,1\}^*\to\{0,1\}$ is defined so that h provides the information in the suffix without yielding information on the prefix; that is, $h(\sigma^{\ell(n)}abc)=abc$ if $\sigma=0$ and $h(\sigma^{\ell(n)}abc)=acb$ otherwise. Thus, $h(\overline{Q}_n)=\overline{Z}_n\overline{X}_n\overline{Y}_n$; that is, it returns $\overline{T}_n$ to its original order (undoing the possible switch employed in $\overline{Q}_n$).

We stress that both h and f are polynomial-time computable.

We will show that the distinguishing algorithm D' (which distinguishes $\overline{E}(\overline{X}_n)$ from $\overline{E}(\overline{Y}_n)$, when also given $Z_n\equiv\overline{Z}_n$) can be transformed into a polynomial-size algorithm A that guesses the value of $f(\overline{Q}_n)$, from the encryption of $\overline{Q}_n$ (and the value of $h(\overline{Q}_n)$), and does so significantly better than with probability $\frac{1}{2}$. This violates semantic security, since no algorithm (regardless of its running-time) can guess $f(\overline{Q}_n)$ better than with probability $1/2$ when only given $h(\overline{Q}_n)$ and $1^{|\overline{Q}_n|}$ (since given $h(\overline{Q}_n)$ and $1^{|\overline{Q}_n|}$, the value of $f(\overline{Q}_n)$ is uniformly distributed over $\{0,1\}$).

On input $(e,\overline{E}_e(\overline{\alpha}),1^{|\overline{\alpha}|},h(\overline{\alpha}))$, where $\overline{\alpha}=\sigma^{\ell(n)}abc\in(\{0,1\}^{l(n)})^{1+m(n)+2t(n)}$ equals either $(0^{\ell(n)},\overline{z},\overline{x},\overline{y})$ or $(1^{\ell(n)},\overline{z},\overline{y},\overline{x})$, algorithm A proceeds in two stages:

1. In the first stage, algorithm A ignores the ciphertext $\overline{E}_e(\overline{\alpha})$. It first extracts $\overline{x},\overline{y}$ and $z\equiv\overline{z}$ out of $h(\overline{\alpha})=\overline{z}\,\overline{x}\,\overline{y}$, and approximates $\Delta(z,\overline{x},\overline{y})$, which is defined to equal
$$\Pr\left[D'(z,G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{x}))=1\right]-\Pr\left[D'(z,G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{y}))=1\right] \tag{5.9}$$
Specifically, using $O(n\cdot p(n)^2)$ samples, algorithm A obtains an approximation, denoted $\widetilde{\Delta}(z,\overline{x},\overline{y})$, such that $|\widetilde{\Delta}(z,\overline{x},\overline{y})-\Delta(z,\overline{x},\overline{y})|<1/3p(n)$ with probability at least $1-2^{-n}$.
Algorithm A sets $\xi=1$ if $\widetilde{\Delta}(z,\overline{x},\overline{y})>1/3p(n)$, sets $\xi=-1$ if $\widetilde{\Delta}(z,\overline{x},\overline{y})<-1/3p(n)$, and sets $\xi=0$ otherwise (i.e., $|\widetilde{\Delta}(z,\overline{x},\overline{y})|\le 1/3p(n)$).
In case $\xi=0$, algorithm A halts with an arbitrary reasonable guess (say a randomly selected bit). (We stress that all this is done obliviously of the ciphertext $\overline{E}_e(\overline{\alpha})$, which is only used next.)
2. In the second stage, algorithm A extracts the last block of ciphertexts (i.e., $\overline{E}_e(c)$) out of $\overline{E}_e(\overline{\alpha})=\overline{E}_e(\sigma^{\ell(n)}abc)$, and invokes D' on input $(z,e,\overline{E}_e(c))$, where z is as extracted in the first stage. Using the value of ξ as determined in the first stage, algorithm A decides as follows:

- In case $\xi = 1$, algorithm A outputs 1 if and only if the output of D' is 1.
- In case $\xi = -1$, algorithm A outputs 0 if and only if the output of D' is 1.

Claim 5.2.15.2: Let p, $\overline{Q}_n$, h, f and A be as above.

$$\mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{Q}_n), h(\overline{Q}_n)) = f(\overline{Q}_n)\right] \;>\; \frac{1}{2} + \frac{1}{10 \cdot p(n)^2}$$

Proof sketch: We focus on the case in which the approximation of $\Delta(z, \overline{x}, \overline{y})$ computed by (the first stage of) A is within $1/3p(n)$ of the correct value. Thus, in case $\xi \neq 0$, the sign of ξ concurs with the sign of $\Delta(z, \overline{x}, \overline{y})$. It follows that, for every possible $(z, \overline{x}, \overline{y})$ such that $\xi = 1$ (it holds that $\Delta(z, \overline{x}, \overline{y}) > 0$ and) the following holds

$$\begin{aligned}
&\mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{Q}_n), h(\overline{Q}_n)) = f(\overline{Q}_n) \,|\, (Z_n, \overline{X}_n, \overline{X}_n) = (z, \overline{x}, \overline{y})\right] \\
&\quad = \frac{1}{2} \cdot \mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(0^{\ell(n)}, z, \overline{x}, \overline{y}), h(0^{\ell(n)}, z, \overline{x}, \overline{y})) = 0\right] \\
&\qquad + \frac{1}{2} \cdot \mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(1^{\ell(n)}, z, \overline{y}, \overline{x}), h(1^{\ell(n)}, z, \overline{y}, \overline{x})) = 1\right] \\
&\quad = \frac{1}{2} \cdot \mathsf{Pr}\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{y})) = 0\right] \\
&\qquad + \frac{1}{2} \cdot \mathsf{Pr}\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{x})) = 1\right] \\
&\quad = \frac{1}{2} \cdot (1 + \Delta(z, \overline{x}, \overline{y}))
\end{aligned}$$

Similarly, for every possible $(z, \overline{x}, \overline{y})$ such that $\xi = -1$ (it holds that $\Delta(z, \overline{x}, \overline{y}) < 0$ and) the following holds

$$\begin{aligned}
&\mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{Q}_n), h(\overline{Q}_n)) = f(\overline{Q}_n) \,|\, (Z_n, \overline{X}_n, \overline{X}_n) = (z, \overline{x}, \overline{y})\right] \\
&\quad = \frac{1}{2} \cdot \mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(0^{\ell(n)}, z, \overline{x}, \overline{y}), h(0^{\ell(n)}, z, \overline{x}, \overline{y})) = 0\right] \\
&\qquad + \frac{1}{2} \cdot \mathsf{Pr}\left[A(G_1(1^n), \overline{E}_{G_1(1^n)}(1^{\ell(n)}, z, \overline{y}, \overline{x}), h(1^{\ell(n)}, z, \overline{y}, \overline{x})) = 1\right] \\
&\quad = \frac{1}{2} \cdot \mathsf{Pr}\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{y})) = 1\right] \\
&\qquad + \frac{1}{2} \cdot \mathsf{Pr}\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{x})) = 0\right] \\
&\quad = \frac{1}{2} \cdot (1 - \Delta(z, \overline{x}, \overline{y}))
\end{aligned}$$

Thus, in both cases where $\xi \neq 0$, algorithm A succeeds with probability

$$\frac{1 + \xi \cdot \Delta(z, \overline{x}, \overline{y})}{2} \;=\; \frac{1 + |\Delta(z, \overline{x}, \overline{y})|}{2}$$

and in case $\xi = 0$ it succeeds with probability 1/2. Recall that if $\Delta(z, \overline{x}, \overline{y}) > \frac{2}{3p(n)}$ then $\xi = 1$. Using the contradiction hypothesis that asserts that $\mathsf{E}[\Delta(Z_n, \overline{X}_n, \overline{Y}_n)] > \frac{1}{p(n)}$, we lower bound $\mathsf{Pr}[\Delta(Z_n, \overline{X}_n, \overline{X}_n) > \frac{2}{3p(n)}]$ by $\frac{1}{3p(n)}$. Thus, the overall success probability of algorithm A is at least

$$\frac{1}{3p(n)} \cdot \frac{1 + (2/3p(n))}{2} + \left(1 - \frac{1}{3p(n)}\right) \cdot \frac{1}{2} \;=\; \frac{1}{2} + \frac{1}{(3p(n))^2}$$

and the claim follows. $\square$

This completes the proof of the opposite direction. $\blacksquare$

Discussion: The proof of the first (i.e., important) direction holds also in the single-message setting. In general, for any function t, in order to prove that semantic security holds with respect to t-long sequences of ciphertexts, we just use the hypothesis that t-long message-sequences have indistinguishable encryptions. In contrast, the proof of the second (i.e., opposite) direction makes an essential use of the multiple-message setting. In particular, in order to prove that t-long message-sequences have indistinguishable encryptions, we use the hypothesis that semantic security holds with respect to $(1 + m + 2t)$-long sequences of ciphertexts, where m depends on the length of the auxiliary input in the claim of ciphertext-indistinguishability. Thus, even if we only want to establish ciphertext-indistinguishability in the single-message setting, we do so by using semantic security in the multiple-message setting. Furthermore, we use the fact that given a sequence of ciphertexts, we can extract a certain subsequence of ciphertexts.

5.2.5.3 Single-message versus multiple-message definitions

As in the non-uniform case, for the public-key model, single-message security implies multiple-message security. Again, this implication does NOT hold in the private-key model. The proofs of both statements are analogous to the proofs provided in the non-uniform case. Specifically:

1. For the public-key model, single-message uniform-indistinguishability of encryptions imply multiple-message uniform-indistinguishability of encryptions, which in turn implies multiple-message uniform-semantic security.

   In the proof of this result, we use the fact that all hybrids are polynomial-time constructible, and that we may select a random pair of neighboring hybrids (cf. the proof of Theorem 3.2.6). We also use the fact that an ensemble of triplets, $\{\overline{T}_n = \overline{X}_n \overline{Y}_n Z'_n\}_{n \in \mathbb{N}}$, with $\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})$, $\overline{Y}_n = (Y_n^{(1)}, ..., Y_n^{(t(n))})$, as in Definition 5.2.14, induces an ensemble of triplets, $\{T_n = X_n Y_n Z_n\}_{n \in \mathbb{N}}$, for the case $t \equiv 1$. Specifically, we shall use $X_n = X_n^{(i)}$, $Y_n = Y_n^{(i)}$, and $Z_n = (\overline{X}_n, \overline{Y}_n, Z'_n, i)$, where i is uniformly distributed in $\{1, ..., t(n)\}$.

2. For the private-key model, single-message uniform-indistinguishability of encryptions does NOT imply multiple-message uniform-indistinguishability of encryptions. The proof is exactly as in the non-uniform case.

5.2.5.4 The gain of a uniform treatment

Suppose that one is content with the uniform-complexity level of security, which is what we advocate below. Then the gain in using the uniform-complexity treatment is that a uniform-complexity level of security can be obtained using only uniform complexity assumptions (rather than non-uniform complexity assumptions). Specifically, the results presented in the next section are based on non-uniform assumptions such as the existence of functions that cannot be inverted by polynomial-size circuits (rather than by probabilistic polynomial-time algorithms). These non-uniform assumption are used in order to satisfy the non-uniform definitions presented in the main text (above). Using any of these constructions, while making the analogous uniform assumptions, yields encryption schemes with the analogous uniform-complexity security. (We stress that this is no coincidence, but is rather an artifact of these results being proven by a uniform reducibility argument.)

However, something is lost when relying on these (seemingly weaker) uniform complexity assumptions. Namely, the security we obtain is only against the (seemingly weaker) uniform adversaries. We believe that this loss in security is immaterial. Our belief is based on the thesis that uniform complexity is the right model of "real world" cryptography. We believe that it is reasonable to consider only objects (i.e., inputs) generated by uniform and efficient procedures and the effect that these objects have on uniformly and efficient observers (i.e., adversaries). In particular, schemes secure against probabilistic polynomial-time adversaries can be used in any setting consisting of probabilistic polynomial-time machines with inputs generated by probabilistic polynomial-time procedures. We believe that the cryptographic setting is such a case.

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

Probabilistic Encryption: Before starting, we recall 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.[4] 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. Thus, the common practice of using pseudorandom permutations as "block-ciphers" (see definition below) is NOT secure (again, one can distinguish two encryptions of the same message from encryptions of two different messages). This explains the linkage between the above robust security definitions and *randomized* (a.k.a *probabilistic*) encryption schemes. Indeed, all our encryption schemes will employ randomized encryption algorithms.[5]

5.3.1 * Stream–Ciphers

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 of it that 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 Section 3.3). Thus, we obtain a *private-key stream cipher*, that allows to encrypt a stream of plaintext bits. Note that such a stream cipher does not conform with our formulation of an encryption scheme (i.e., as in Definition 5.1.1), since for encrypting several messages one is required to maintain a counter. In other words, we obtain a encryption scheme with a variable state that is modified after the encryption of each message.

We comment that constructions of secure and stateless encryption schemes (i.e., conforming with Definition 5.1.1) are known (and presented in Sections 5.3.3 and 5.3.4). The traditional interest in stream ciphers is due to efficiency considerations. We discuss this issue at the end of Section 5.3.3. But before doing so, let us formalize the above discussion.

[4] 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 public-key setting and in the multi-message private-key setting?)

[5] The (private-key) stream-ciphers discussed in Section 5.3.1 are an exception, but– as we point out– they do not adhere to our (basic) formulation of encryption schemes (as in Definition 5.1.1).

Definitions: The following formalization extends Definition 5.1.1. The key-generation algorithm remains unchanged, but both the encryption and decryption algorithm take an additional input and emit an additional output, corresponding to their state before and after the operation. The length of the state is not allowed to grow by too much during each application of the encryption algorithm (see Item 3 below), or else efficiency of the entire "repeated encryption" process can not be guaranteed. The initial state (of both algorithms) is the empty string. For clarity, the reader may consider of the special case in which the state counts the number of times the scheme was invoked (or the total number of plaintext bits in such invocations).

Definition 5.3.1 (state-based cipher – the mechanism): *A* state-based 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 outputs a pair of bit strings.*
2. *For every pair (e, d) in the range of $G(1^n)$, every string s (representing a possible state), every $\alpha \in \{0,1\}^*$, and every pair (s', β) in the range of $E(e, s, \alpha)$ it holds that $D(d, s, \beta) = \alpha$.*

   (That is, given ciphertext and the decryption-key along with the state of the encryption process before encrypting the current plaintext, the decryption algorithm retrieves the plaintext.)[6]
3. *There exists a polynomial p so that for every pair (e, d) in the range of $G(1^n)$, every string s (representing a possible state), every $\alpha \in \{0,1\}^*$, and every pair (s', β) in the range of $E(e, s, \alpha)$ it holds that $|s'| \leq |s| + |\alpha| \cdot p(n)$.*

That is, as in Definition 5.1.1, the encryption-decryption process operates properly (i.e., the decrypted message equals the plaintext), provided that the corresponding algorithm get the corresponding keys along with the same start state. However, since the above holds also for the empty string s, it is not clear in what sense the above yields something novel. Indeed, the novelty is in the security definition that refers to the encryption of multiple messages, and holds only in case the state is properly maintained throughout the multiple-message encryption process. Below we present only the semantic security definition for private-key schemes.

Definition 5.3.2 (semantic security – state-based cipher): *For a state-based encryption scheme, (G, E, D), and any $\overline{x} = (x^{(1)}, ..., x^{(t)})$, we let $\overline{E}_e(\overline{x}) = (S^{(0)}, C^{(1)}), ..., (S^{(t-1)}, C^{(t)})$ be the result of the following t-step random process, where $S^{(0)}$ is the empty string. For $i = 1, ..., t$, we let $(S^{(i)}, C^{(i)}) \leftarrow E_e(S^{(i-1)}, x^{(i-1)})$, where each of the t invocations E_e utilizes independently chosen random coins. The scheme (G, E, D) is* semantically secure in the state-based

[6] A stronger requirement, which is achieved by the construction below, is that $D(d, s, \beta) = (s', \alpha)$. That is, that the decryption algorithm can figure out the updated state of the encryption process. This extra property is useful in applications where decryption is performed in the same order as encryption (e.g., in FIFO communication).

private-key model *if for every polynomial $t(\cdot)$ and every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that for every ensemble $\{\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})\}_{n\in\mathbb{N}}$, with $|X_n^{(i)}| = \mathrm{poly}(n)$, every pair of functions $f, h : \{0,1\}^* \to \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n*

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

Note that Definition 5.3.2 differs from Definition 5.2.8 (only) in the preamble defining the random variable $\overline{E}_e(\overline{x})$. Furthermore, Definition 5.3.2 guarantees nothing regarding an encryption process in which the plaintext sequence $(x^{(1)}, ..., x^{(t)})$ is encrypted by $E_e(S^{(0)}, x^{(1)}), E_e(S^{(0)}, x^{(2)}), ..., E_e(S^{(0)}, x^{(t)})$ (i.e., the state is reset to $S^{(0)}$ after each encryption). Finally, note that the security is preserved also when the adversary knows all intermediate values of the encryption state. (This is required in applications, since we need to pass these intermediate states to the receiver in order to allow proper decryption.)

A sound version of a common practice: Loosely speaking, using any pseudorandom generator, one can easily construct a secure state-based private-key encryption scheme. The state will hold the total number of bits encrypted so far, and the i^{th} bit to be encrypted is encrypted by XORing it with the i^{th} bit of the pseudorandom generator. A minor technicality arises, since pseudorandom generators were defined to have an a-priori fixed number of output bits (as a function of the seed length), whereas here we need an a-priori unbounded (polynomial) number of bits. This technicality is resolved by using variable-output pseudorandom generators as defined and studied in Section 3.3.3.

Construction 5.3.3 (how to construct state-based private-key encryption schemes): *Let $G:\{0,1\}^*\times 1^{\mathbb{N}} \to \{0,1\}^*$ so that $|G(r, 1^t)| = t$, for every $r \in \{0,1\}^*$ and $t \in \mathbb{N}$.*

key-generation and initial state: *Uniformly select $r \in \{0,1\}^n$, and output the key-pair (r, r). The initial state is viewed as $t = 0$.*

encrypting plaintext x with key r and state t: *Let $\ell = |x|$ and p be the ℓ-bit suffix of $G(r, 1^{t+\ell})$. Then the ciphertext is $x \oplus p$, and the new state is set to $t + \ell$.*

decrypting ciphertext y with key r and state t: *Let $\ell = |y|$ and p be the ℓ-bit suffix of $G(r, 1^{t+\ell})$. Then the ciphertext is $y \oplus p$.*[7]

The reader may easily verify the following:

[7] The decryption algorithm can infer the state of the encryption algorithm (i.e., $t + \ell$). For motivation, see Footnote 6.

Proposition 5.3.4 *Suppose that G is a variable-output pseudorandom generator with respect to polynomial-size circuits. Then Construction 5.3.3 constitutes a secure state-based private-key encryption scheme.*

5.3.2 Preliminaries: Block–Ciphers

Many encryption schemes are more conveniently presented by first presenting a restricted type of encryption scheme that we call a *block-cipher*.[8] 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 5.3.5 (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 $\ell : \mathbb{N} \to \mathbb{N}$, 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*

$$\mathsf{Pr}[D_d(E_e(\alpha)) = \alpha] = 1$$

Typically, we use 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 are essentially as before with the modification that we only consider plaintexts of length $\ell(n)$. For example, the analogue of Definition 5.2.1 reads

Definition 5.3.6 (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 for every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that 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*

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

[8] In using the term *block-cipher*, 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. We comment that the latter cannot be semantically secure; see Exercise 19.

Transforming block-ciphers into a general encryption schemes: There are 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 that is not an integer multiple of the block-length (i.e., $\ell(n)$). This is easily resolved by padding the last block (while indicating the end of the actual plaintext).[9]

Construction 5.3.7 (from block-ciphers to general encryption schemes): *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 consecutive blocks of length $\ell(n)$, while possibly augmenting the last block. Let $\alpha_1,...,\alpha_t$ be the resulting blocks. Then*

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

To decrypt the ciphertext $(m,\beta_1,...,\beta_t)$ (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 concatenated 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, and that we cannot hope to entirely hide the length. However, we can easily construct encryption schemes that hide some information about the length of the plaintext; see examples in Exercise 4. Also, note that the above construction applies even to the special case where ℓ is identically 1.

Proposition 5.3.8 *Let (G,E,D) and (G',E',D') be as in Construction 5.3.7. Suppose that the former a secure private-key*[10] (resp., public-key) *block-cipher. Then the latter is a secure private-key* (resp., public-key) *encryption scheme.*

Proof Sketch: The proof is by a reducibility argument. Assuming towards the contradiction that the encryption scheme (G',E',D') is not secure, we conclude that neither is (G,E,D), contradicting our hypothesis. Note that in case the security of (G',E',D') is violated via $t(n)$ messages of length $L(n) = \mathrm{poly}(n)$, the security of (G,E,D) is violated via $t(n) \cdot \lceil L(n)/\ell(n) \rceil$ messages of length $\ell(n)$. Also, the argument may utilize any of the two notions of security (i.e., semantic security or ciphertext-indistinguishability). ■

[9] We choose to use a very simple indication of the end of the actual plaintext (i.e., include its length in the ciphertext). In fact, it suffices to include the length of the plaintext modulo $\ell(n)$. Another natural alternative is to use a padding of the form $10^{(\ell(n)-|\alpha|-1) \bmod \ell(n)}$, while observing that no padding is ever required in case $\ell(n)=1$.

[10] Recall that throughout this section *security* means security in the multiple-message setting.

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). Specifically, we 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, we have

Construction 5.3.9 (a private-key block-cipher based on pseudorandom functions): *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 $\ell(n) = n$ as follows*

key-generation: $G(1^n) = (i, i)$, *where* $i \leftarrow I(1^n)$.

encrypting plaintext $x \in \{0,1\}^n$: $E_i(x) = (r, V(i,r) \oplus x)$, *where r is uniformly chosen in* $\{0,1\}^n$.

decrypting ciphertext (r, y): $D_i(r, y) = V(i, r) \oplus y$

Below we assume that F is *pseudorandom with respect to polynomial-size circuits*, meaning that no polynomial-size circuit having "oracle gates" can distinguish the case the answers are provided by a random function from the case in which the answers are provided by a function in F. Alternatively, one may consider probabilistic polynomial-time oracle machines that obtain a non-uniform polynomially-long auxiliary input. That is,

> *for every probabilistic polynomial-time oracle machine M for every pair of positive polynomial p and q, for all sufficiently large n's and all $z \in \{0,1\}^{p(n)}$,*
>
> $$\left|\Pr\left[M^f(z)=1\right] - \Pr\left[M^{f_{I(1^n)}}(z)=1\right]\right| < \frac{1}{q(n)}$$
>
> *where f is a uniformly selected function mapping $\{0,1\}^n$ to $\{0,1\}^n$.*

Recall, that such (non-uniformly strong) pseudorandom functions can be constructed using any non-uniformly strong one-way function.

Proposition 5.3.10 *Let F and (G, E, D) be as in Construction 5.3.9, and suppose that F is pseudorandom with respect to polynomial-size circuits. Then (G, E, D) is secure.*

Combining Propositions 5.3.8 and 5.3.10 (with the above), we obtain

Theorem 5.3.11 *If there exist (non-uniformly strong) one-way functions then there exist secure private-key encryption schemes.*

The converse holds too; see Exercise 1.

Proof of Proposition 5.3.10: 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 $f:\{0,1\}^n \to \{0,1\}^n$, 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).

Specifically, in the ideal version the messages $x^{(1)}, ..., x^{(t)}$ are encrypted by $(r^{(1)}, f(r^{(1)}) \oplus x^{(1)}), ..., (r^{(t)}, f(r^{(t)}) \oplus x^{(t)})$, where the $r^{(i)}$'s are independently and uniformly selected, and f is a random function. Thus, with probability greater than $1 - t^2 \cdot 2^{-n}$, the $r^{(i)}$'s are all distinct and so the values $f(r^{(i)}) \oplus x^{(i)}$ are independently and uniformly distributed, regardless of the $x^{(i)}$'s. It follows that the ideal version is ciphertext-indistinguishable. Now, if the actual scheme is not ciphertext-indistinguishable, then for some sequence of $r^{(i)}$'s a polynomial-size circuit can distinguish the $f(r^{(i)}) \oplus x^{(i)}$'s from the $f_s(r^{(i)}) \oplus x^{(i)}$'s, where f is random and f_s is pseudorandom. But this contradicts the hypothesis that polynomial-size circuits cannot distinguish between the two. ■

Discussion: Note that we could have gotten rid of the randomization if we had allowed the encryption algorithm to be history dependent (as discussed in Section 5.3.1 above). Specifically, in such a case, we could have used 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. However, in the later case Construction 5.3.3 is preferable (because the adequate pseudorandom generator may be more efficient than a pseudorandom function as used in Construction 5.3.9). We note that in case the encryption scheme is not used for FIFO communication and one may need to decrypt messages with arbitrary varying counter values, it is typically better to use Construction 5.3.9. Furthermore, in many cases it may be preferable to select a value (i.e., r) at random rather than rely on a counter that must stored in a reliable manner between applications (of the encryption algorithm).

The ciphertexts produced by Construction 5.3.9 are longer than the corresponding plaintexts. This is unavoidable in case of secure (history-independent) encryption schemes (see Exercise 19). In particular, the common practice of using pseudorandom permutations as block-ciphers[11] is NOT secure (e.g., one

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

can distinguish two encryptions of the same message from encryptions of two different messages).

5.3.4 Public-key encryption schemes

As mentioned above, randomization during the encryption process can be avoided in private-key encryption schemes that employ a varying state (not allowed in our basic Definition 5.1.1). In case of public-key encryption schemes, randomization during the encryption process is essential (even if the encryption scheme employs a varying state). Thus, the randomized encryption paradigm plays an even more pivotal role in the construction of public-key encryption scheme. To demonstrate this paradigm we start with a very simple (and quite wasteful) construction. But before doing so, we recall the notion of trapdoor permutations.

Trapdoor permutations: All our constructions employ a collection of trapdoor permutations, as in Definition 2.4.5. Recall that such a collection, $\{p_\alpha\}_\alpha$, comes with four probabilistic polynomial-time algorithms, denoted here by I, S, F and B (for *index*, *sample*, *forward* and *backward*), such that the following (syntactic) conditions hold

1. On input 1^n, algorithm I selects a random n-bit long *index* α of a permutation p_α, along with a corresponding trapdoor τ;
2. On input α, algorithm S *samples* the domain of p_α, returning a random element in it;
3. For x in the domain of p_α, given α and x, algorithm F returns $p_\alpha(x)$ (i.e., $F(\alpha, x) = p_\alpha(x)$);
4. For y in the range of p_α if (α, τ) is a possible output of $I(1^n)$ then, given τ and y, algorithm B returns $p_\alpha^{-1}(y)$ (i.e., $B(\tau, y) = p_\alpha^{-1}(y)$).

The hardness condition refers to the difficulty of inverting p_α on a random element of its range, when given only the range-element and α. That is, let $I_1(1^n)$ denote the first element in the output of $I(1^n)$ (i.e., the index), then for every polynomial-size circuit family $\{C_n\}$, every polynomial p and all sufficiently large n's

$$\mathsf{Pr}[C_n(I_1(1^n), p_{I_1(1^n)}(S(I_1(1^n))) = S(I_1(1^n))] \; < \; \frac{1}{p(n)}$$

Namely, C_n fails to invert p_α on $p_\alpha(x)$, where α and x are selected by I and S as above. Recall the above collection can be easily modified to have a hard-core predicate (cf. Theorem 2.5.2). For simplicity, we continue to refer to the collection as $\{p_\alpha\}$, and *let b denote the corresponding hard-core predicate.*

5.3.4.1 Simple schemes

We are now ready to present a very simple (alas quite wasteful) construction of a secure public-key encryption scheme. It is a block-cipher with $\ell \equiv 1$.

Construction 5.3.12 (a simple public-key block-cipher scheme): *Let $\{p_\alpha\}$, I, S, F, B and b be as above.*

key-generation: *The key generation algorithm consists of selecting at random a permutation p_α together with a trapdoor τ for it: The permutation (or rather its description) serves as the public-key, whereas the trapdoor serves as the private-key. That is, $G(1^n) = I(1^n)$, which means that the index-trapdoor pair generated by I is associated with the key-pair of G.*

encryption: *To encrypt a bit σ (using the encryption-key α), the encryption algorithm randomly selects an element, r, in the domain of p_α and produces the ciphertext $(p_\alpha(r), \sigma \oplus b(r))$. That is, $E_\alpha(\sigma) = (F(\alpha, r), \sigma \oplus b(r))$, where $r \leftarrow S(\alpha)$.*

decryption: *To decrypt the ciphertext (y, ς) (using the decryption-key τ), the decryption algorithm just computes $\varsigma \oplus b(p_\alpha^{-1}(y))$, where the inverse is computed using the trapdoor τ of p_α. That is, $D_\tau(y, \varsigma) = b(B(\tau, y)) \oplus \varsigma$.*

Clearly, for every possible (α, τ) output of G and for every $\sigma \in \{0,1\}$, it holds that

$$\begin{aligned} D_\tau(E_\alpha(\sigma)) &= b(B(\tau, F(\alpha, S(\alpha)))) \oplus (\sigma \oplus b(S(\alpha))) \\ &= b(p_\alpha^{-1}(p_\alpha(S(\alpha)))) \oplus (\sigma \oplus b(S(\alpha))) \\ &= b(S(\alpha)) \oplus \sigma \oplus b(S(\alpha)) \;=\; \sigma \end{aligned}$$

The security of the above public-key encryption scheme follows from the (non-uniform) one-way feature of the collection $\{p_\alpha\}$ (or rather from the hypothesis that b is a corresponding hard-core predicate).

Proposition 5.3.13 *Suppose that b is a (non-uniformly strong) hard-core of the collection $\{p_\alpha\}$. Then Construction 5.3.12 constitute a secure public-key block-cipher (with block-length $\ell \equiv 1$).*

Proof: Recall that by the equivalence theorems (i.e., Theorems 5.2.5 and 5.2.11), it suffices to show single-message ciphertext-indistinguishability. Furthermore, by Proposition 5.2.7 and the fact that here there are only two plaintexts (i.e., 0 and 1), it suffices to show that one cannot predict which of the two plaintexts (selected at random) is being encrypted (significantly better than with success probability 1/2). We conclude by noting that a guess σ' for the plaintext σ, given the encryption-key α and the ciphertext $E_\alpha(\sigma) = (f_\alpha(r), \sigma \oplus b(r))$, where $r \leftarrow S(\alpha)$, yields a guess $\sigma' \oplus (\sigma \oplus b(r))$ for $b(r)$ given $(\alpha, f_\alpha(r))$. The latter guess is correct with probability equal to the probability that $\sigma' = \sigma$, and contradicts the hypothesis that b is a hard-core of $\{p_\alpha\}$. Details follow.

Recall that by saying that b is a hard-core of $\{p_\alpha\}$ we mean that for every polynomial-size circuit family $\{C_n\}$, every polynomial p and all sufficiently large n's

$$\mathsf{Pr}[C_n(I_1(1^n), p_{I_1(1^n)}(S(I_1(1^n))) = b(S(I_1(1^n)))] \ < \ \frac{1}{2} + \frac{1}{p(n)} \tag{5.10}$$

By Proposition 5.2.7, it suffices to show that for randomly chosen α (i.e., $\alpha \leftarrow I_1(1^n)$) and uniformly distributed $\sigma \in \{0,1\}$, no polynomial-size circuit given the encryption-key α and the ciphertext $E_\alpha(\sigma)$, can predict σ non-negligibly better than with success probability $1/2$. Suppose towards the contradiction that there exists a polynomial-size circuit family $\{C'_n\}$, a polynomial p' and infinitely many n's such that

$$\mathsf{Pr}[C'_n(I_1(1^n), E_{I_1(1^n)}(\sigma)) = \sigma] \ > \ \frac{1}{2} + \frac{1}{p'(n)} \tag{5.11}$$

where σ is uniformly distributed in $\{0,1\}$. Recall that $E_\alpha(\sigma) = (p_\alpha(r), \sigma \oplus b(r))$, where $r \leftarrow S(\alpha)$ is a random sample in p_α's domain, and consider the following probabilistic circuit C''_n: On input α and y (in the range of p_α), the circuit C''_n uniformly selects $\varsigma \in \{0,1\}$, invokes C'_n on input $(\alpha, (y, \varsigma))$, and outputs $C'_n(\alpha, (y, \varsigma)) \oplus \varsigma$. In the following analysis of the behavior of C''_n, we let $\alpha \leftarrow I_1(1^n)$, $r \leftarrow S(\alpha)$, and consider uniformly distributed $\varsigma, \sigma \in \{0,1\}$:

$$\begin{aligned}
\mathsf{Pr}[C''_n(\alpha, p_\alpha(r)) = b(r)] &= \mathsf{Pr}[C'_n(\alpha, (p_\alpha(r), \varsigma)) \oplus \varsigma = b(r)] \\
&= \mathsf{Pr}[C'_n(\alpha, (p_\alpha(r), \varsigma)) = \varsigma \oplus b(r)] \\
&= \mathsf{Pr}[C'_n(\alpha, (p_\alpha(r), \sigma \oplus b(r)) = (\sigma \oplus b(r)) \oplus b(r)] \\
&= \mathsf{Pr}[C'_n(\alpha, E_\alpha(\sigma)) = \sigma] \\
&> \frac{1}{2} + \frac{1}{p'(n)}
\end{aligned}$$

where the inequality is due to Eq. (5.11). Removing the randomization from C''_n (i.e., by fixing the best possible choice), we derive a contradiction to Eq. (5.10). The proposition follows. ■

Using Propositions 5.3.8 and 5.3.13, and recalling that Theorem 2.5.2 applies also to collections of one-way functions and to the non-uniform setting, we obtain

Theorem 5.3.14 *If there exist collections of* (*non-uniformly hard*) *trapdoor permutations then there exist secure public-key encryption schemes.*

A generalization: As admitted above, Construction 5.3.12 is quite wasteful. Specifically, it is wasteful in *bandwidth*; that is, the relation between the length of the plaintext and the length of the ciphertext. In Construction 5.3.12 the relation between these lengths equals the security parameter (i.e., the length of description of individual elements in the domain of the permutation). However, the idea underlying Construction 5.3.12 can yield efficient public-key schemes, provided we use trapdoor permutations having hard-core functions with large range (see Section 2.5.3). To demonstrate the point, we use the following assumption relating to the RSA collection of trapdoor permutations (cf. Subsections 2.4.3 and 2.4.4).

Large hard-core conjecture for RSA: *The first $n/2$ least significant bits of the argument constitute a* (non-uniformly strong) *hard-core function of the RSA function when applied with n-bit long moduli.*

We stress that the conjecture is NOT know to follow from the assumption that the RSA collection is (non-uniformly) hard to invert. What can be proved under the latter assumption is only that the first $O(\log n)$ least significant bits of the argument constitute a (non-uniformly strong) hard-core function of RSA (with n-bit long moduli). Still, if the above conjecture holds then one obtains a secure public-key encryption scheme with efficiency comparable to that of "plain RSA" (see discussion below). Furthermore, this scheme coincides with the common practice of randomly padding messages (using padding equal in length to the message) before encrypting them applying the RSA function. That is, we consider the following scheme:

Construction 5.3.15 (Randomized RSA – a public-key block-cipher scheme): *This scheme employs the RSA collection of trapdoor permutations (cf. Subsections 2.4.3 and 2.4.4). The following description is, however, self-contained.*

key-generation: *The key generation algorithm consists of selecting at random two n-bit primes, P and Q, setting $N = P \cdot Q$, selecting at random a pair (e, d) so that $e \cdot d \equiv 1 \pmod{(P-1)\cdot(Q-1)}$, and outputting the tuple $((N, e), (N, d))$, where (N, e) is the encryption-key and (N, d) is the decryption-key. That is, $((N, e), (N, d)) \leftarrow G(1^n)$, where N, e and d are as specified above.*

(Note that N is $2n$-bit long.)

encryption: *To encrypt an n-bit string σ (using the encryption-key (N, e)), the encryption algorithm randomly selects an element, $r \in \{1, ..., N-1\}$, and produces the ciphertext $(r^e \bmod N, \sigma \oplus \text{LSB}(r))$, where $\text{LSB}(r)$ denotes the n least significant bits of r. That is, $E_{(N,e)}(\sigma) = (r^e \bmod N, \sigma \oplus \text{LSB}(r))$.*

decryption: *To decrypt the ciphertext $(y, \varsigma) \in \{0, ..., N-1\} \times \{0,1\}^n$ (using the decryption-key (N, d)), the decryption algorithm just computes $\varsigma \oplus \text{LSB}(y^d \bmod N)$, where $\text{LSB}(\cdot)$ is as above. That is, $D_{(N,d)}(y, \varsigma) = \varsigma \oplus \text{LSB}(y^d \bmod N)$.*

The bandwidth of the above scheme is much better than in Construction 5.3.12: a plaintext of length n is encrypted via a ciphertext of length $2n$. On the other hand, Randomized RSA is almost as efficient as "plain RSA" (or the RSA function itself).

To see that Randomized RSA satisfies the syntactic requirements of an encryption scheme, consider any possible output of $G(1^n)$, denoted, $((N,e),(N,d))$, and any $\sigma \in \{0,1\}^n$. Then, it holds that

$$\begin{aligned} D_{(N,d)}(E_{(N,e)}(\sigma)) &= D_{(N,d)}((r^e \bmod N), \sigma \oplus \text{LSB}(r)) \\ &= (\sigma \oplus \text{LSB}(r)) \oplus \text{LSB}((r^e \bmod N)^d \bmod N) \\ &= \sigma \oplus \text{LSB}(r) \oplus \text{LSB}(r^{ed} \bmod N) \ = \ \sigma \end{aligned}$$

where the last equality is due to $r^{ed} \equiv r \pmod N$. The security of Randomized RSA (as a public-key encryption scheme) follows from the large hard-core conjecture for RSA, analogously to the proof of Proposition 5.3.13.

Proposition 5.3.16 *Suppose that the large hard-core conjecture for RSA does hold. Then Construction 5.3.15 constitute a secure public-key block-cipher (with block-length $\ell(n) = n$).*

Proof: Recall that by the equivalence theorems (i.e., Theorems 5.2.5 and 5.2.11), it suffices to show single-message ciphertext-indistinguishability. Considering any two strings x and y, we need to show that $(r^e \bmod N, x \oplus \text{LSB}(r))$ and $(r^e \bmod N, y \oplus \text{LSB}(r))$ are indistinguishable, where N, e and r are selected at random as in the construction. It suffices to show that, for every x, the distributions $(r^e \bmod N, x \oplus \text{LSB}(r))$ and $(r^e \bmod N, x \oplus s)$ are indistinguishable, where $s \in \{0,1\}^n$ is uniformly distributed, independently of anything else. The latter claim follows from the hypothesis that the n least significant bits are a hard-core function for RSA with moduli of length $2n$. ■

Discussion: We wish to stress that *encrypting messages by merely applying the RSA function to them (without randomization), yields an insecure encryption scheme.* Unfortunately, this procedure (referred to about as 'plain RSA'), is quite common in practice. The fact that plain RSA is definitely insecure is a special case of the fact that any public-key encryption scheme that employs a deterministic encryption algorithm is insecure. We warn that the fact that in such deterministic encryption schemes one can distinguish encryptions of two specific messages (e.g., the all-zero message and the all-one message) is not "merely of theoretical concern" – it may seriously endanger some applications! In contrast, Randomized RSA (as defined in Construction 5.3.15) *may be* secure, provided a quite reasonable conjecture (i.e., the large hard-core conjecture for RSA) holds. Thus, the common practice of applying the RSA function to a randomly-padded version of the plaintext is way superior to using the RSA function directly (i.e., without randomization): *the randomized version is likely to be secure, whereas the non-randomized (or plain) version is definitely insecure.*

Recall that Construction 5.3.15 (or alternatively Construction 5.3.12) generalizes to any collection of trapdoor permutations having a corresponding large hard-core function. Suppose that $\{p_\alpha\}$ is such a collection, and h (or rather $\{h_\alpha\}$) is a corresponding hard-core function (resp., a corresponding collection of hard-core functions) such that any element in the domain of p_α is mapped to an $\ell(|\alpha|)$-bit long string. Then we can encrypt an $\ell(|\alpha|)$-bit long plaintext, x, by $(p_\alpha(r), h(r) \oplus x)$ (resp., $(p_\alpha(r), h_\alpha(r) \oplus x)$), where $r \leftarrow S(\alpha)$ (as in Construction 5.3.12). This yields a secure public-key encryption scheme with bandwidth that relates to the relation between $\ell(|\alpha|)$ and the length of a description of individual elements in the domain of p_α.

5.3.4.2 An alternative scheme

An alternative construction of a public-key encryption scheme is presented below. Rather than encrypting each plaintext bit (or block of bits) by an independently selected element in the domain of the trapdoor permutation (as done in Construction 5.3.12), we select only one such element (for the entire plaintext), and generate an additional bit per each bit of the plaintext. These additional bits are determine by successive applications of the trapdoor permutation, and only the last result is included in the ciphertext. In a sense, the construction of the encryption scheme (below) augments the construction of a pseudorandom generator based on one-way permutations (i.e., Construction 3.4.4).

Construction 5.3.17 (a public-key encryption scheme): *Let $\{p_\alpha\}$, I, S, F, B and b be as in Construction 5.3.12. We use the notation $p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x))$ and $p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x))$.*

key-generation: *The key-generation algorithm consists of selecting at random a permutation p_α together with a trapdoor, exactly as in Construction 5.3.12. That is, $G(1^n) = I(1^n)$, which means that the index-trapdoor pair generated by I is associated with the key-pair of G.*

encryption: *To encrypt a string σ (using the encryption-key α), the encryption algorithm randomly selects an element, r, in the domain of p_α and produces the ciphertext $(p_\alpha^{|\sigma|}(r), \sigma \oplus G_\alpha(r))$, where*

$$G_\alpha(r) \stackrel{\rm def}{=} b(r) \cdot b(p_\alpha(r)) \cdots b(p_\alpha^{|\sigma|-1}(r)) \tag{5.12}$$

That is, $E_\alpha(\sigma) = (p_\alpha^{|\sigma|}(S(\alpha)), \sigma \oplus G_\alpha(S(\alpha)))$.

decryption: *To decrypt the ciphertext (y, ς) (using the decryption-key τ), the decryption algorithm just computes $\varsigma \oplus G_\alpha(p_\alpha^{-|\varsigma|}(y))$, where the inverse is computed using the trapdoor τ of p_α. That is, $D_\tau(y, \varsigma) = \varsigma \oplus G_\alpha(p_\alpha^{-|\varsigma|}(y))$.*

We stress that the above encryption scheme is a full-fledged one (rather than a block-cipher). Its bandwidth tends to 1 with the length of the plaintext; that is, a plaintext of length $\ell = \text{poly}(n)$ is encrypted via a ciphertext of length $m + \ell$, where m denotes the length of the description of individual elements in the domain of p_α. Clearly, for every possible (α, τ) output of G, it holds that $D_\tau(E_\alpha(\sigma)) = \sigma$. The security of the above public-key encryption scheme follows from the (non-uniform) one-way feature of the collection $\{p_\alpha\}$, but here we restrict the sampling algorithm S to produce almost uniform distribution over the domain (so that this distribution is preserved under successive applications of p_α).

Proposition 5.3.18 *Suppose that b is a (non-uniformly strong) hard-core of the trapdoor collection $\{p_\alpha\}$. Furthermore, suppose that this trapdoor collection utilizes a domain sampling algorithm S so that the statistical difference between $S(\alpha)$ and the uniform distribution over the domain of p_α is negligible in terms of $|\alpha|$. Then Construction 5.3.17 constitute a secure public-key encryption scheme.*

Proof: Again, we prove single-message ciphertext-indistinguishability. As in the proof of Proposition 5.3.16, it suffices to show that, for every σ, the distributions $(p_\alpha^{|\sigma|}(S(\alpha)), \sigma \oplus G_\alpha(S(\alpha)))$ and $(p_\alpha^{|\sigma|}(S(\alpha)), \sigma \oplus s)$ are indistinguishable, where $s \in \{0,1\}^{|\sigma|}$ is uniformly distributed, independently of anything else. The latter claim holds by a minor extension to Proposition 3.4.6: the latter refers to the case $S(\alpha)$ is uniform over the domain of p_α, but can be extended to the case in which there is a negligible statistical difference between the distributions. The proposition follows. ■

An instantiation: 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 Section 2.4.3). This yields a secure public-key encryption scheme (presented below) with efficiency comparable to that of plain RSA. Recall that plain RSA itself is not secure (as it employs a deterministic encryption algorithm), whereas Randomized RSA (i.e., Construction 5.3.15) is not known to be secure under standard assumption such as intractability of factoring (or of inverting the RSA function).[12]

Construction 5.3.19 (The Blum-Goldwasser Public-Key Encryption Scheme): *For simplicity, we present a block-cipher with arbitrary block-length $\ell(n) = \mathrm{poly}(n)$.*

key-generation: *The key generation algorithm consists of selecting at random two n-bit primes, P and Q, each congruent to 3 mod 4, and outputting the pair $(N,(P,Q))$, where $N = P \cdot Q$.*

Actually, for sake of efficiency, the key-generator also computes

$$\begin{aligned}
d_P &= ((P+1)/4)^{\ell(n)} \bmod P-1 && (\mathit{in}\ \{0,...,P-2\})\\
d_Q &= ((Q+1)/4)^{\ell(n)} \bmod Q-1 && (\mathit{in}\ \{0,...,Q-2\})\\
c_P &= Q \cdot (Q^{-1} \bmod P) && (\mathit{in}\ \{0,...,N-Q\})\\
c_Q &= P \cdot (P^{-1} \bmod Q) && (\mathit{in}\ \{0,...,N-P\})
\end{aligned}$$

[12] Recall that Randomized RSA is secure assuming that the $n/2$ least significant bits constitute a hard-core function for n-bit RSA moduli. Even when assuming the intractability of factoring, we only know that the $O(\log n)$ least significant bits constitute a hard-core function for n-bit moduli.

It outputs the pair (N, T), *where* N *serves as the encryption-key and* $T = (P, Q, N, c_P, d_P, c_Q, d_Q)$ *serves as decryption-key.*

encryption: *To encrypt the message* $\sigma \in \{0,1\}^{\ell(n)}$, *using the encryption-key* N:

1. *Uniformly select* $s_0 \in \{1, ..., N\}$.
2. *For* $i = 1, .., \ell(n)+1$, *compute* $s_i \leftarrow s_{i-1}^2 \bmod N$ *and* $b_i = \mathrm{lsb}(s_i)$, *where* $\mathrm{lsb}(s)$ *is the least significant bit of* s.

The ciphertext is $(s_{\ell(n)+1}, \varsigma)$, *where* $\varsigma = \sigma \oplus b_1 b_2 \cdots b_{\ell(n)}$.

decryption: *To decrypt of the ciphertext* (r, ς) *using the decryption-key* $T = (P, Q, N, c_P, d_P, c_Q, d_Q)$, *one first retrieves* s_1 *and then computes the* b_i*'s as above. Instead of extracting modular square roots successively* $\ell(n)$ *times, we extract the* $2^{\ell(n)}$*-th root, which can be done as efficiently as extracting a single square root:*

1. *Let* $s' \leftarrow r^{d_P} \bmod P$, *and* $s'' \leftarrow r^{d_Q} \bmod Q$.
2. *Let* $s_1 \leftarrow c_P \cdot s' + c_Q \cdot s'' \bmod N$.
3. *For* $i = 1, .., \ell(n)$, *compute* $b_i = \mathrm{lsb}(s_i)$ *and* $s_{i+1} \leftarrow s_i^2 \bmod N$.

The plaintext is $\varsigma \oplus b_1 b_2 \cdots b_{\ell(n)}$.

Again, one can easily verify that the above construction constitutes an encryption scheme: the main fact to verify is that the value of s_1 as reconstructed in the decryption stage equals the value used in the encryption stage. This follows by combining the Chinese Reminder Theorem with the fact that for every quadratic residue $s \bmod N$ it holds that $s \equiv (s^{2^\ell} \bmod N)^{d_P} \pmod P$ (and similarly, $s \equiv (s^{2^\ell} \bmod N)^{d_Q} \pmod Q$).

Details: Recall that for a prime $P \equiv 3 \pmod 4$, and every integer i, we have $i^{(P+1)/2)} \equiv i \pmod P$. Thus, for every integer j, we have

$$\begin{aligned}(j^{2^\ell} \bmod N)^{d_P} &\equiv \left(j^{2^\ell} \bmod N\right)^{((P+1)/4)^\ell} \pmod P \\ &\equiv j^{((P+1)/2)^\ell} \pmod P \\ &\equiv j \pmod P\end{aligned}$$

Similarly, $j \equiv (j^{2^\ell} \bmod N)^{d_Q} \pmod Q$. Observing that c_P and c_Q are as in the Chinese Reminder Theorem (i.e., $i \equiv c_P \cdot (i \bmod P) + c_Q \cdot (i \bmod Q) \pmod N$, for every integer i), we conclude that s_1 as recovered in Step 2 of the decryption equals s_1 as first computed in Step 2 of the encryption.

Encryption amounts to $\ell(n)+1$ modular multiplications, whereas decryption amounts to $\ell(n)+2$ such multiplications and 2 modular exponentiations (relative to half-sized moduli). Counting modular exponentiations with respect to n-bit moduli as $O(n)$ (i.e., at least n and at most $2n$) modular multiplications (with respect to n-bit moduli), we conclude that the entire encryption-decryption process requires work comparable to $2\ell(n) + 2n$ modular multiplications. For comparison to (Randomized) RSA, note that encrypting/decrypting $\ell(n)$-bit messages amounts to $\lceil \ell(n)/n \rceil$ modular exponentiations, and so the total work is comparable to $2 \cdot (\ell(n)/n) \cdot 1.5n = 3\ell(n)$ (for general exponent e, or half that much in case $e = 3$).

The security of the Blum-Goldwasser scheme (i.e., Construction 5.3.19) follows immediately from Proposition 5.3.18 and the fact that lsb is a hard-core for the modular squaring function. Recalling that inverting the latter is computationally equivalent to factoring, we get:

Corollary 5.3.20 *Suppose that factoring is infeasible in the sense that for every polynomial-size circuit $\{C_n\}$, every positive polynomial p and all sufficiently large n's*

$$\mathsf{Pr}[C_n(P_n \cdot Q_n) = P_n] \; < \; \frac{1}{p(n)}$$

where P_n and Q_n are uniformly distributed n-bit long primes. Then Construction 5.3.19 constitutes a secure public-key encryption scheme.

5.4 * Beyond eavesdropping security

Our treatment so far refers only to a "passive" attack in which the adversary merely eavesdrops on the line over which ciphertexts are being sent. Stronger types of attacks, culminating in the so-called Chosen Ciphertext Attack, may be possible in various applications. Specifically, 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* and to *chosen ciphertext attacks*, respectively, which are not covered by the above security definitions. Thus, our main goal in this section is to provide a treatment to such types of attacks. In addition, we also discuss the related notion of non-malleable encryption schemes. We start with an overview of the type of attacks considered in this section.

Types of attacks. The following mini-taxonomy of attacks is certainly not exhaustive.

1. Passive attacks as captured in the definitions above. In case of public-key schemes we distinguish two sub-cases:
   (a) A *key-oblivious*, passive attack, as captured in the definitions above. By 'key-obliviousness' we refer to the fact that the choice of plaintext does not depend on the public-key.
   (b) A *key-dependent*, passive attack, in which the choice of plaintext may depend on the public-key.

(In Definition 5.2.8 the choice of plaintext means the random variable $\overline{X}_n$, whereas in Definition 5.2.9 it means the pair of sequences $(\overline{x}_n, \overline{y}_n)$. In both these definitions, the choice of the plaintext is key-oblivious.)

2. *Chosen Plaintext Attacks.* Here the attacker may obtain the encryption of any plaintext of its choice (under the key being attacked).

   Indeed, such an attack does not add power in case of public-key schemes.

3. *Chosen Ciphertext Attacks.* Here the attacker may obtain the decryption of any ciphertext of its choice (under the key being attacked). That is, the attacker is given oracle access to the decryption function corresponding to the decryption-key in use. We distinguish two types of such attacks.

   (a) In an *a-priori chosen* ciphertext attack, the attacker is given this oracle access prior to being presented the ciphertext that it should attack (i.e., the ciphertext for which it has to learn partial information). That is, the attack consists of two stages: in the first stage the attacker is given the above oracle access, and in the second stage the oracle is removed and the attacker is given a 'test ciphertext' (i.e., a target to be learned).

   (b) In an *a-posteriori chosen* ciphertext attack, after being given the target ciphertext, the oracle is not removed but the adversary's access to it is restricted in that it is not allowed to make a query equal to the target ciphertext.

   In both cases, the adversary may make queries that do not correspond to a legitimate ciphertext, and the answer will be accordingly (i.e., a special 'failure' symbol). Furthermore, in both cases the adversary may effect the selection of the target ciphertext.

Formal definitions of all types of attacks listed above are given in the following corresponding subsections. Before presenting the actual definitions, we provide an overview of the known results.

Some known constructions. As in the basic case, the (strongly secure) private-key encryption schemes can be constructed based on the existence of one-way functions, whereas the (strongly secure) public-key encryption schemes are based on the existence of trapdoor permutations. In both cases, withstanding a-posteriori chosen ciphertext attacks is harder than withstanding a-priori chosen ciphertext attacks. We will present the following schemes.

Private-key schemes: The private-key encryption scheme based on pseudorandom functions (i.e., Construction 5.3.9), is secure also *against a-priori chosen ciphertext attacks.*[13]

[13] Note that this scheme is not secure under an a-posteriori chosen ciphertext attack: on input a ciphertext $(r, x \oplus f_s(r))$, we obtain $f_s(r)$ by making the query (r, y'), where $y' \neq x \oplus f_s(r)$. (This query is answered with x' so that $y' = x' \oplus f_s(r)$.)

It is easy to turn any passively secure private-key encryption scheme into a scheme *secure under (a-posteriori) chosen ciphertext attacks*, by using a message authentication scheme[14] on top of the basic encryption.

Public-key schemes: Public-key encryption schemes secure against a-priori chosen ciphertext attacks can be constructed, assuming the existence of trapdoor permutations and utilizing non-interactive zero-knowledge proofs. (Recall that the latter proof systems can be constructed under the former assumption.)

Public-key encryption schemes secure against a-posteriori chosen ciphertext attacks can also be constructed under the same assumption, but this construction is even more complex.

5.4.1 Key-dependent passive attacks

Author's Note: *Applicable only to public-key schemes.*

Author's Note: *Plan: define, and show that above constructions satisfy the definition.*

5.4.2 Chosen plaintext attack

Author's Note: *No affect in case of public-key schemes.*

Author's Note: *Plan: define, and show that above constructions satisfy the definition.*

5.4.3 Chosen ciphertext attack

Author's Note: *For private-key, refer also to a combined plaintext+ciphertext attack.*

Author's Note: *Plan:*

1. *Define the two types (i.e., a-priori and a-posteriori CCA).*
2. *Prove that the PRF-based private-key scheme remains secure under a-priori CCA.*
3. *Using MAC, transform any passively-secure private-key scheme into an a scheme secure under a-posteriori CCA.*
4. *The NY-framework for constructing CCA-secure public-key schemes: double-encryption + use of NIZK. [172]*
   (a) *Apply the framework to obtain security under a-priori CCA. [NY]*

---

[14] See definition in Section 6.1.

(b) *Apply the framework to obtain security under a-posteriori CCA. [DDN, Amit] [62, ?]*

Author's Note: *Indeed, my plans were modified due to a recent result of Amit Sahai (further simplified by Yehuda Lindell). This result makes it feasible to present (in the context of the current book) a public-key encryption scheme that is secure under a-posteriori CCA.*

5.4.4 Non-malleable encryption schemes

Author's Note: *Tentative introductory test follows.*

So far, our treatment has referred to an adversary that tries to extract explicit information about the plaintext. A less explicit attempt, captured by the so-called notion of *malleability*, is to generate an encryption of a related plaintext (possibly without learning anything about the original plaintext).

Thus, we have a "matrix" of adversaries, with one dimension (parameter) being the *type of attack* and the second being its *purpose*. So far, we have discussed the first dimension (i.e., the type of the attack). We now turn to the second (i.e., the purpose of the attack). We make a distinction between the following two notions (or purposes of attack):

1. Standard *security*: the infeasibility of *obtaining information regarding the plaintext*. As defined above, such information must be a function (or a randomized process) applied to the bare plaintext, and may not depend on the encryption (or decryption) key.

2. In contrast, the notion of *non-malleability* refers to generating a string depending on both the plaintext and the current encryption-key. Specifically, one requires that it should be infeasible for an adversary, given a ciphertext, to produce a valid ciphertext for a related plaintext. For example, given a ciphertext of a plaintext of the form $1x$, it should be infeasible to produce a ciphertext to the plaintext $0x$.

We shall show below that, with the exception of passive attacks on private-key schemes, non-malleability always implies security against attempts to obtain information on the plaintext. We shall also show that security and non-malleability are equivalent under a-posteriori chosen ciphertext attack.

Author's Note: *Plan:*

1. *discuss and define the notion of non-malleable encryption,*
2. *prove the following (relations between definitions):*
   (a) *With the exception of passive attacks on private-key schemes, non-malleability always implies security against attempts to obtain information on the plaintext.*

(b) Under a-posteriori chosen ciphertext attacks, security and non-malleability are equivalent. (This is very intuitive and the intuition can be presented already in the definitional discussion above!)

Derive the existence of non-malleable encryption schemes as a corollary to the above.

5.5 Miscellaneous

Author's Note: *The entire section 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, it seems that 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 [196]: In his work, 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. He thus concluded that 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 relaxation 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 [61]. First concrete candidates were suggested by Rivest, Shamir and Adleman [187] and by Merkle and Hellman [159]. However, satisfactory definitions of security were presented only a few years afterwards, by Goldwasser and Micali [121]. The two definitions presented in Section 5.2 originate in [121], where it was shown that ciphertext-indistinguishability implies semantic security. The converse direction is due to [160].

Regarding the seminal paper of Goldwasser and Micali [121], a few additional comments are due. Arguably, this paper is the basis of the entire rigorous approach to cryptography (presented in the current book): It introduced general notions such as computational indistinguishability, definitional approaches such as the simulation paradigm, and techniques such as the hybrid argument. The paper's title ("Probabilistic Encryption") is due to the authors' 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). Finally, the term "ciphertext-indistinguishability" used here replaces the (generic) term "polynomial-security" used in [121]. Some of our modifications have already appeared in [93], which is also the main source of our uniform-complexity treatment.

The first construction of a secure public-key encryption scheme based on a simple complexity assumption was given by Goldwasser and Micali [121].[15] Specifically, they constructed a public-key encryption scheme assuming that deciding Quadratic Residiousity modulo composite numbers is intractable. The condition was weaken by Yao [205] who prove that any trapdoor permutation will do. The efficient public-key encryption scheme of Construction 5.3.19 is due to Blum and Goldwasser [33]. 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 [5].

For decades, it has been common practice to use "pseudorandom generators" in the design of stream ciphers. As pointed out by Blum and Micali [34], this practice is sound *provided* that one uses pseudorandom generators (as defined in Chapter 3). The construction of private-key encryption schemes based on pseudorandom functions is due to [101].

We comment that it is indeed peculiar that the rigorous study of (the security of) private-key encryption schemes has legged behind the corresponding study of public-key encryption schemes. This historical fact may be explained by the very thing that makes it peculiar; that is, private-key encryption schemes are less complex that public-key ones, and hence the problematics of their security (when applied to popular candidates) is less obvious. In particular, the need for a rigorous study of (the security of) public-key encryption schemes arose from observations regarding their concrete applications (e.g., doubts raised by Lipton concerning the security of the "mental poker" protocol of [195], which used "plain RSA" as an encryption scheme). In contrast, the need for a rigorous study of (the security of) private-key encryption schemes arose later and by analogy to the public-key case.

Author's Note: *The rest of this subsection is yet to be written. The following notes are merely place-holders.*

Author's Note: *The NY-framework for constructing public-key encryption schemes secure under Chosen Ciphertext Attacks: double-encryption + NIZK (Naor and Yung [172]). Its (original) application to the case of a-priori Chosen Ciphertext Attacks [172]. Its application to the case of a-priori Chosen Ciphertext Attacks [?] (Sahai following Dolev, Dwork and Noar [62]). Refer to NIZK works such as [75, 189].*

[15] Recall that plain RSA is not secure, whereas Randomized RSA is based on the Large Hard-Core Conjecture for RSA (which is less appealing that the standard conjecture referring to the intractability of inverting RSA).

Author's Note: *The study of* non-malleability *of the encryption schemes, was initiated by Dolev, Dwork and Noar [62]. Security and non-malleability are equivalent under a-posteriori chosen ciphertext attack (cf. [62, 15]).*

5.5.2 Suggestion for Further Reading

Author's Note: *This subsection is yet to be written. The following notes are merely place-holders.*

Author's Note: *On the "gap" between private-key and public-key encryption; the former is possible under OWF whereas Impagliazzo and Rudich indicate the this is unlikely for the latter [131].*

Author's Note: *For discussion of Non-Malleable Cryptography, which actually transcends the domain of encryption, see [62].*

Author's Note: *For a detailed discussion of the relationship among the various notions of secure private-key and public-key encryption, the reader is referred to [138]* and *[15], respectively.*

5.5.3 Open Problems

Secure public-key encryption schemes exist if there exist collections of (non-uniformly hard) trapdoor permutations (cf. Theorem 5.3.14). It is not known whether the converse holds (although secure public-key encryption schemes easily imply one-way function). (The few-to-1 feature of the function collection is important; see [17].)

Randomized RSA (i.e., Construction 5.3.15) is commonly believed to be a secure public-key encryption scheme. It would be of great practical importance to gain additional support for this belief. As shown in Proposition 5.3.16, the security of Randomized RSA follows from the *Large Hard-Core Conjecture for RSA*, but the latter is not known to follow from a more standard assumption such as that RSA is hard to invert. This is indeed the third place in this book where we suggest the establishment of the latter implication as an important open problem.

Both constructions of *public-key* encryption schemes *secure against chosen ciphertext attacks* (presented in Section 5.4) are to be considered as plausibility results (which also offer some useful construction paradigms). Presenting "reasonably-efficient" public-key encryption schemes that are secure against (a-posteriori) chosen ciphertext attacks, under widely believed assumptions, is an important open problem. (We comment that the "reasonably-efficient" scheme of [55] is based on a very strong assumption regarding the *Diffie-Hellman Key Exchange*. Specifically, it is assumed that for a prime P and primitive element g, given $(P, g, (g^x \bmod P), (g^y \bmod P), (g^z \bmod P))$, it is infeasible to decide whether $z \equiv xy \pmod{P-1}$.)

5.5.4 Exercises

Author's Note: *The following are but a tentative collection of exercises that occurred to me while writing the main text.*

Exercise 1: *Encryption schemes imply one-way function* [129]: Show that the existence of a secure private-key encryption scheme (i.e., as in Definition 5.2.1) implies the existence of one-way functions.

> **Guideline:** Recall that, by Exercise 11 of Chapter 3, it suffices to prove that the former implies the existence of a pair of polynomial-time constructible probability ensembles that are statistically far apart and still are computationally indistinguishable. To prove the existence of such ensembles consider the encryption of $n+1$-bit plaintexts relative to a random n-bit long key, denoted K_n. Specifically, let the first ensemble be $\{(U_{n+1}, E(U_{n+1}))\}_{n\in\mathbb{N}}$, where $E(x) = E_{K_n}(x)$, and the second ensemble be $\{(U^{(1)}_{n+1}, E(U^{(2)}_{n+1}))\}_{n\in\mathbb{N}}$, where $U^{(1)}_{n+1}$ and $U^{(2)}_{n+1}$ are independently distributed. It is easy to show that these ensembles are computationally indistinguishable and are both polynomial-time constructible. The more interesting part is to show that these ensembles are statistically far apart. To prove this fact, assume towards the contradiction that for all but a negligible fraction of the 2^{n+1} possible x's, the distribution of $E(x)$ is statistically close to a single distribution Y, and show that this does not allow correct decryption (since there are only 2^n possible keys).

Exercise 2: *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 semantically secure public-key encryption scheme. (Hint: A plaintext of length exponential in the security parameter allows the adversary to find the decryption key by exhaustive search.)

Exercise 3: *Encryption schemes 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 semantically secure encryption scheme.

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

Exercise 4: *Hiding partial information about the length of the plaintext*: Using an arbitrary encryption scheme, construct an encryption scheme that hides the exact length of the plaintext. In particular, construct an encryption scheme that reveals only the following function h' of the length of the plaintext:

1. $h'(m) = \lceil m/n \rceil \cdot n$, where n is the security parameter.
2. $h'(m) = 2^{\lceil \log_2 m \rceil}$

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

Exercise 5: *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. Furthermore, show that (without loss of generality) the length of ciphertexts may be a fixed polynomial in the length of the plaintext.

Exercise 6: *Deterministic encryption schemes*: Prove that in order to be semantically 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 7: Prove that the following definition, in which we use non-uniform families of polynomial-size circuits (rather than probabilistic polynomial-time algorithms) is equivalent to Definition 5.2.1.

> There exists a probabilistic polynomial-time transformation T such that for every polynomial-size circuit family $\{C_n\}_{n\in\mathbb{N}}$, and for every $\{X_n\}_{n\in\mathbb{N}}$, $f,h : \{0,1\}^* \to \{0,1\}^*$, $p(\cdot)$ and n as in Definition 5.2.1
>
> $$\begin{aligned} &\mathsf{Pr}\left[C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right] \\ &< \ \mathsf{Pr}\left[C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right] + \frac{1}{p(n)} \end{aligned}$$
>
> where $C'_n \leftarrow T(C_n)$ and the probability is also taken over the internal coin tosses of T.

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 above definition states that advices for a machine that gets the ciphertext can be efficiently transformed into advices for a machine that does not get the ciphertext. However, we can incorporate the (probabilistic) transformation program into the second universal algorithm (which then become probabilistic). Consequently, the advices are identical for both machines (and can be incorporated in the auxiliary input $h(X_n)$ used in Definition 5.2.1). Viewed this way, the above definition is equivalent to asserting that for some (universal) deterministic polynomial-time algorithm U there exists a probabilistic polynomial-time algorithm U' and for every $\{X_n\}_{n\in\mathbb{N}}$, $f,h : \{0,1\}^* \to \{0,1\}^*$, $p(\cdot)$ and n as in Definition 5.2.1
>
> $$\begin{aligned} &\mathsf{Pr}\left[U(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right] \\ &< \ \mathsf{Pr}\left[U'(1^{|X_n|}, h(X_n)) = f(X_n)\right] + \frac{1}{p(n)} \end{aligned}$$
>
> Still, a gap remains between the above definition and Definition 5.2.1: the above condition refers only to one possible deterministic algorithm

U, whereas Definition 5.2.1 refers to all probabilistic polynomial-time algorithms. To close the gap, we first observe that (by Propositions 5.2.7 and 5.2.6) Definition 5.2.1 is equivalent to a form in which one only quantifies over deterministic polynomial-time algorithms A. We conclude by observing that one can code any algorithm A (and polynomial time-bound) referred to by Definition 5.2.1 in the auxiliary input (i.e., $h(X_n)$) given to U.

Exercise 8: In continuation to Exercise 7, consider a definition in which the transformation T (of the circuit family $\{C_n\}_{n\in\mathbb{N}}$ to the circuit family $\{C'_n\}_{n\in\mathbb{N}}$) is not required to (even) be computable.[16] Clearly, the new definition is not stronger than the one in Exercise 7. Show that the two definitions are in fact equivalent.

> **Guideline:** Use the furthermore-clause of Proposition 5.2.7 to show that the new definition implies indistinguishability of encryptions, and conclude by applying Proposition 5.2.6 and invoking Exercise 7.

Exercise 9: Prove that Definition 5.2.3 remains unchanged when supplying the circuit with auxiliary-input. That is, an encryption scheme satisfies Definition 5.2.3 if and only if

> for every polynomial-size circuit family $\{C_n\}$, every polynomial p, all sufficiently large n and every $x, y \in \{0,1\}^{\text{poly}(n)}$ (i.e., $|x| = |y|$) and $z \in \{0,1\}^{\text{poly}(n)}$,
>
> $$|\Pr\left[C_n(z, E_{G_1(1^n)}(x))=1\right] - \Pr\left[C_n(z, E_{G_1(1^n)}(y))=1\right]| < \frac{1}{p(n)}$$

(Hint: incorporate z in the circuit C_n.)

Exercise 10: *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 11: *The technical contents of semantic security*: The following explains the lack of computational requirements regarding the function f, in Definition 5.2.1. 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 probabilistic polynomial-time algorithm A' so that for every $\{X_n\}_{n\in\mathbb{N}}$ and $h : \{0,1\}^* \to \{0,1\}^*$ as in Definition 5.2.1, the following two ensembles are computationally indistinguishable.
>
> 1. $\{E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)\}_{n\in\mathbb{N}}$.

[16] Equivalently, one may require that for any polynomial-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ there exists a polynomial-size circuit family $\{C'_n\}_{n\in\mathbb{N}}$ satisfying the above inequality.

2. $\{A'(1^{|X_n|}, h(X_n))\}_{n\in\mathbb{N}}$.

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

> **Guideline:** We care mainly about the (easy to establish) fact by which the above implies semantic security. The other direction can be proven analogously to the proof of Proposition 5.2.6.

Exercise 12: Prove that Definition 5.2.1 remains unchanged if we may restrict the function h to depend only on the length of its input (i.e., $h(x) = h'(|x|)$ for some $h' : \mathbb{N} \to \{0,1\}^*$).

> **Guideline:** It suffices to prove that this special case (i.e., obtained by the restriction on h) is equivalent to the general one. This follows by combining Propositions 5.2.7 and 5.2.6.

Exercise 13: *A variant on Exercises 11 and 12*: Prove that an encryption scheme, (G, E, D), is (semantically) secure (in the private-key model) if and only if the following holds.

> For every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, and polynomially bounded h' the following two ensembles are computationally indistinguishable.
>
> 1. $\{A(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h'(|X_n|))\}_{n\in\mathbb{N}}$.
> 2. $\{A'(1^{|X_n|}, h'(|X_n|))\}_{n\in\mathbb{N}}$.

(Indeed, since $|X_n|$ is constant, so is $h'(|X_n|)$. So an equivalent form is obtained by replacing $h'(|X_n|)$ with a $\text{poly}(n)$-bit long string v_n.)

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

> **Guideline:** Again, we care mainly about the fact that the above implies semantic security. The easiest proof of this direction is by applying Propositions 5.2.7 and 5.2.6. A more interesting proof is obtained by using Exercise 11: Indeed, the current formulation is a special case of the formulation in Exercise 11, and so we need to prove that it implies the general case. The latter is proven by observing that otherwise – using an averaging argument – we derive a contradiction in one of the residual probability spaces defined by conditioning on $h(X_n)$ (i.e., $(X_n|h(X_n) = v)$ for some v).

Exercise 14: *Another equivalent definition of security*: The following exercise is interesting mainly for historical reasons. In the definition of semantic security appearing in [121], the term $\max_{u,v}\{\mathsf{Pr}[f(X_n) = v|h(X_n) = u]\}$ appears instead of the term $\mathsf{Pr}[A'(1^{|X_n|}, h(X_n)) = f(X_n)]$. That is, it is required that for every probabilistic polynomial-time algorithm A every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \rightarrow \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

$$\begin{aligned} &\Pr\left[A(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n)) = f(X_n)\right] \\ &\quad < \max_{u,v}\{\Pr\left[f(X_n) = v | h(X_n) = u\right]\} + \frac{1}{p(n)} \end{aligned}$$

Prove that the above formulation is in fact equivalent to Definition 5.2.1.

Guideline: First, note that the above definition implies Definition 5.2.1 (since $\max_{u,v}\{\Pr[f(X_n) = v | h(X_n) = u]\} \geq \Pr[A'(h(X_n), 1^n, |X_n|) = f(X_n)]$, for every algorithm A'). Next note that in the *special case*, in which X_n satisfies $\Pr[f(X_n) = 0 | h(X_n) = u] = \Pr[f(X_n) = 1 | h(X_n) = u] = \frac{1}{2}$, for all u's, the above terms are equal (since A' can easily achieve success probability 1/2 by simply always outputting 1). Finally, combining Propositions 5.2.7 and 5.2.6. infer that it suffices to consider only the latter special case.

Exercise 15: *Yet another equivalent definition of security*: The following syntactic strengthening of semantic security is important in some applications. Its essence is in considering information *related* to the plaintext, in the form of a related random variable, rather than partial information about the plaintext (in the form of a function of it). Prove that an encryption scheme, (G, E, D), is (semantically) secure (in the private-key model) if and only if the following holds.

For every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that for every $\{(X_n, Z_n)\}_{n\in\mathbb{N}}$, with $|(X_n, Z_n)| = \text{poly}(n)$, where Z_n may dependent arbitrarily on X_n, and f, $p(\cdot)$ and n as in Definition 5.2.1

$$\begin{aligned} &\Pr\left[A(E_{G_1(1^n)}(X_n), 1^{|X_n|}, Z_n) = f(X_n)\right] \\ &\quad < \Pr\left[A'(1^{|X_n|}, Z_n) = f(X_n)\right] + \frac{1}{p(n)} \end{aligned}$$

That is, the auxiliary input $h(X_n)$ of Definition 5.2.1 is replaced by the random variable Z_n. Formulate and prove an analogous claim for the public-key model.

Guideline: Definition 5.2.1 is clearly a special case of the above. On the other hand, the proof of Proposition 5.2.6 extends easily to the above (seemingly stronger) formulation of semantic security.

Exercise 16: *Extended Semantic Security* (suggested by Boaz Barak): Consider an extended definition of semantic security in which, in addition to the regular inputs, the algorithms have oracle access to a function

$H_x : \{0,1\}^* \to \{0,1\}^*$ (instead of being given the value $h(x)$). The function H_x's have to be restricted to have polynomial (in $|x|$) size circuit. That is, *an encryption scheme,* (G,E,D)*, is* extended-semantically secure (in the private-key model) *For every probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' such that for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \mathrm{poly}(n)$, every polynomially-bounded function $f : \{0,1\}^* \to \{0,1\}^*$, every family of polynomial-sized circuits $\{H_x\}_{x\in\{0,1\}^*}$, every polynomial $p(\cdot)$ and all sufficiently large n*

$$\begin{aligned} &\mathsf{Pr}\left[A^{H_{X_n}}(E_{G_1(1^n)}(X_n), 1^{|X_n|}) = f(X_n)\right] \\ &\quad < \mathsf{Pr}\left[A'^{H_{X_n}}(1^{|X_n|}) = f(X_n)\right] + \frac{1}{p(n)} \end{aligned}$$

The definition of public-key security is analogous.

1. Show that if (G,E,D) has indistinguishable encryptions then it is extended-semantically secure.
2. Show that if no restriction are placed on the H_x's then no scheme can be extended-semantically secure (in this unrestricted sense).

Guideline (for Part 1): The proof is almost identical to the proof of Proposition 5.2.6: The algorithm A' forms an encryption of $1^{|X_n|}$, and invokes A on it. Indistinguishability of encryptions is used in order to establish that $A'^{H_{X_n}}(1^{|X_n|})$ performs essentially as well as $A^{H_{X_n}}(E(X_n))$. Otherwise, we obtain a distinguisher of $E(x_n)$ from $E(1^{|x_n|})$, for some infinite sequence of x_n's. In particular, the oracle H_{x_n} (being implementable by a small circuit) can be incorporated into a distinguisher.

Guideline (for Part 2): In such a case, H_x may be defined so that, when queried about a ciphertext, it reveals the decryption-key in use. partial information about the corresponding plaintext. This is obvious in case of public-key schemes, but is also doable in some private-key schemes (e.g., suppose that the ciphertext always contains a commitment to the private-key). Such an oracle allows A (which is given a ciphertext) to recover the corresponding plaintext, but does not help A' (which is given $1^{|X_n|}$) find any information about the value of X_n.

Exercise 17: *Multiple messages of varying lengths*: In continuation to Section 5.2.4, generalize the treatment to encryption of multiple messages of varying lengths. Provide adequate definitions, and analogous results.

Guideline: For example, a generalization of the first item of Definition 5.2.8 postulates that for every pair of polynomials $t(\cdot)$ and $\ell(\cdot)$, and every probabilistic polynomial-time algorithm A, there exists a probabilistic polynomial-time algorithm A' such that for every ensemble $\{\overline{X}_n = (X_n^{(1)},...,X_n^{(t(n))})\}_{n\in\mathbb{N}}$, with $|X_n^{(i)}| \leq \ell(n)$, every pair of functions $f,h : \{0,1\}^* \to \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

$$\begin{aligned} &\mathsf{Pr}\left[A(\overline{E}_{G_1(1^n)}(\overline{X}_n), (1^{|X_n^{(1)}|},...,1^{|X_n^{(t(n))}|}), h(\overline{X}_n)) = f(\overline{X}_n)\right] \\ &\quad < \mathsf{Pr}\left[A'((1^{|X_n^{(1)}|},...,1^{|X_n^{(t(n))}|}), h(\overline{X}_n)) = f(\overline{X}_n)\right] + \frac{1}{p(n)} \end{aligned}$$

Exercise 18: *Known plaintext attacks*: Loosely speaking, in a known plaintext 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 that is secure in the multiple-message setting is also secure in the presence of known plaintext attack.

Guideline (for Part 3): Consider a function h in the multiple-message setting that reveals some of the plaintexts.

Exercise 19: *The standard term of block-cipher*: A standard block-cipher is a triple, (G, E, D), of probabilistic polynomial-time algorithms that satisfies Definition 5.3.5 as well as $|E_e(\alpha)| = \ell(n)$ for every pair (e, d) in the range of $G(1^n)$ and every $\alpha \in \{0,1\}^{\ell(n)}$.

1. Prove that a standard block-cipher cannot be semantically secure (in the multiple-message model). Furthermore, show that any semantically secure encryption scheme must employ ciphertexts that are longer than the corresponding plaintexts.
2. Present a state-based version of a standard block-cipher and note that Construction 5.3.3 satisfies it.

Guideline (for Part 1): Consider the encryption of a pair of two identical messages versus the encryption of a pair of two different messages, and use the fact that E_e must be a permutation of $\{0,1\}^{\ell(n)}$. Extend the argument to any encryption scheme in which plaintexts of length $\ell(n)$ are encrypted by ciphertexts of length $\ell(n)+O(\log n)$, observing that otherwise most plaintexts have only poly(n)-many ciphertexts under E_e.

Exercise 20: The Blum-Goldwasser public-key encryption scheme was presented in Construction 5.3.19 as a block-cipher (with arbitrary block length). Provide an alternative presentation of this scheme as a full-fledged encryption scheme (rather than a block-cipher), and prove its security (under the factoring assumption).

Author's Note: *First draft written mainly in 1997. Major revision completed in Dec. 1999. Further significant revisions were conducted in May–June 2001.*

Index