Fragments of a chapter on Encryption Schemes

(Extracts from Foundations of Cryptography – in preparation)

Oded Goldreich

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

December 25, 1999

to Dana

Preface

It is possible to build a cabin with no foundations,
but not a lasting building.

Eng. Isidor Goldreich (1906–1995)

Cryptography is concerned with the construction of schemes that withstand any abuse: Such schemes are constructed so to maintain a desired functionality, even under malicious attempts aimed at making them deviate from their prescribed functionality.

The design of cryptography schemes is a very difficult task. One cannot rely on intuitions regarding the typical state of the environment in which the system operates. For sure, the *adversary* attacking the system will try to manipulate the environment into untypical states. Nor can one be content with counter-measures designed to withstand specific attacks, since the adversary (which acts after the design of the system is completed) will try to attack the schemes in ways that are typically different from the ones the designer had envisioned. The validity of the above assertions seems self-evident, still some people hope that in practice ignoring these tautologies will not result in actual damage. Experience shows that these hopes rarely come true; cryptographic schemes based on make-believe are broken, typically sooner than later.

In view of the above, we believe that it makes little sense to make assumptions regarding the specific *strategy* that the adversary may use. The only assumptions that can be justified refer to the computational *abilities* of the adversary. Furthermore, it is our opinion that the design of cryptographic systems has to be based on *firm foundations*; whereas ad-hoc approaches and heuristics are a very dangerous way to go. A heuristic may make sense when the designer has a very good idea about the environment in which a scheme is to operate, yet a cryptographic scheme has to operate in a maliciously selected environment which typically transcends the designer's view.

This book is aimed at presenting firm foundations for cryptography. The foundations of cryptography are the paradigms, approaches and techniques used to conceptualize, define and provide solutions to natural "security concerns". We will present some of these paradigms, approaches and techniques as well as some of the fundamental results obtained using them. Our emphasis is on the clarification of fundamental concepts and on demonstrating the feasibility of solving several central cryptographic problems.

Solving a cryptographic problem (or addressing a security concern) is a two-stage process consisting of a *definitional stage* and a *constructive stage*. First, in the definitional stage, the functionality underlying the natural concern is to be identified, and an adequate cryptographic problem has to be defined. Trying to list all undesired situations is infeasible and prone to error. Instead, one should define the functionality in terms of operation in an imaginary ideal model, and require a candidate solution to emulate this operation in the real, clearly defined, model (which specifies the adversary's abilities). Once the definitional stage is completed, one proceeds to construct a system that satisfies the definition. Such a construction may use some simpler tools, and its security is proven relying on the features of these tools. In practice, of course, such a scheme may need to satisfy also some specific efficiency requirements.

This book focuses on several archetypical cryptographic problems (e.g., encryption and signature schemes) and on several central tools (e.g., computational difficulty, pseudorandomness, and zero-knowledge proofs). For each of these problems (resp., tools), we start by presenting the natural concern underlying it (resp., its intuitive objective), then define the problem (resp., tool), and finally demonstrate that the problem may be solved (resp., the tool can be constructed). In the latter step, our focus is on demonstrating the feasibility of solving the problem, not on providing a practical solution. As a secondary concern, we typically discuss the level of practicality (or impracticality) of the given (or known) solution.

Computational Difficulty

The specific constructs mentioned above (as well as most constructs in this area) can exist only if some sort of computational hardness exists. Specifically, all these problems and tools require (either explicitly or implicitly) the ability to generate instances of hard problems. Such ability is captured in the definition of one-way functions (see further discussion in Section 2.1). Thus, one-way functions is the very minimum needed for doing most sorts of cryptography. As we shall see, they actually suffice for doing much of cryptography (and the rest can be done by augmentations and extensions of the assumption that one-way functions exist).

Our current state of understanding of efficient computation does not allow us to prove that one-way functions exist. In particular, the existence of one-way functions implies that $\mathcal{NP}$ is not contained in $\mathcal{BPP} \supseteq \mathcal{P}$ (not even "on the average"), which would resolve the most famous open problem of computer science. Thus, we have no choice (at this stage of history) but to assume that one-way functions exist. As justification to this assumption we may only offer the combined believes of hundreds (or thousands) of researchers. Furthermore, these believes concern a simply stated assumption, and their validity follows from several widely believed conjectures which are central to some fields (e.g., the conjecture that factoring integers is hard is central to computational number theory).

As we need assumptions anyhow, why not just assume what we want (i.e., the existence of a solution to some natural cryptographic problem)? Well, first we need to know what we want: as stated above, we must first clarify what exactly we want; that is, go through the typically complex definitional stage. But once this stage is completed, can we just assume that the definition derived can be met? Not really: once a definition is derived how can we know that it can at all be met? The way to demonstrate that a definition is viable (and so the intuitive security concern can be satisfied at all) is to construct a solution based on a *better understood* assumption (i.e., one that is more common and widely believed). For example, looking at the definition of zero-knowledge proofs, it is not a-priori clear that such proofs exist at all (in a non-trivial sense). The non-triviality of the notion was first demonstrated by presenting a zero-knowledge proof system for statements, regarding Quadratic Residuosity, which are believed to be hard to verify (without extra information). Furthermore, in contrary to prior beliefs, it was later shown in that the existence of one-way functions implies that any NP-statement can be proven in zero-knowledge. Thus, facts which were not known at all to hold (and even believed to be false), where shown to hold by reduction to widely believed assumptions (without which most of modern cryptography collapses anyhow). To summarize, not all assumptions are equal, and so reducing a complex, new and doubtful assumption to a widely-believed simple (or even merely simpler) assumption is of great value. Furthermore, reducing the solution of a new task to the assumed security of a well-known primitive typically means providing a construction that, using the known primitive, solves the new task. This means that we do not only know (or assume) that the new task is solvable but rather have a solution based on a primitive that, being well-known, typically has several candidate implementations.

Structure and Prerequisites

Our aim is to present the basic concepts, techniques and results in cryptography. As stated above, our emphasis is on the clarification of fundamental concepts and the relationship among them. This is done in a way independent of the particularities of some popular number theoretic examples. These particular examples played a central role in the development of the field and still offer the most practical implementations of all cryptographic primitives, but this does not mean that the presentation has to be linked to them. On the contrary, we believe that concepts are best clarified when presented at an abstract level, decoupled from specific implementations. Thus, the most relevant background for this book 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 (yet, a short appendix presenting the most relevant facts is included in this volume so to support the few examples of implementations presented here).

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

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

Volume 3: Beyond the Basics

. . .

Figure 0.1: Organization of this book

Organization of the book. The book is organized in three parts (see Figure 0.1): *Basic Tools*, *Basic Applications*, and *Beyond the Basics*. The first volume contains an introductory chapter as well as the first part (Basic Tools). This part contains chapters on computational difficulty (one-way functions), pseudorandomness and zero-knowledge proofs. These basic tools will be used for the Basic Applications of the second part, which consist of Encryption, Signatures, and General Cryptographic Protocols.

The partition of the book 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 years, and publish it without waiting for the third part.

The current manuscript. The current manuscript consists of fragments of a chapter on encryption schemes. 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*), fragments for the Miscellaneous section of this chapter, the above extracts from the preface of Volume 1, and a table of contents that includes Volume 1.

Contents

List of Figures

Part II

Basic Applications

Chapter 5

Encryption Schemes

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

In this chapter we review the well-known notions of private-key and public-key encryption schemes. More importantly, we define what is meant by saying that such schemes are secure. It turns out that using randomness throughout the encryption process (i.e., not only during key-generation) is essential to security. We present some basic constructions of secure encryption schemes. Finally, we discuss "dynamic" notions of security culminating in robustness against chosen ciphertext attacks and non-malleability.

> Author's Note: *Currently the write-up contains only a rough draft for the first 3 sections of this chapter.*

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 retreive the plaintext from the ciphertext, but the adversary cannot do so. Clearly, something must distinguish the receiver (who is able to retreive the plaintext from the corresponding ciphertext) from the adversary (who cannot do so). Specifically, the receiver know something 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. Formally, we need to consider a third algorithm; namely, a probabilistic algorithm used to generate keys. This algorithm must be probabilistic (or else, by invoking it the adversary obtains the very same key used by the receiver).

5.1.1 Overview

In accordance with the above, an encryption scheme consists of three algorithms. These algorithms are public (i.e., known to all parties). The obvious algorithms are the *encryption algorithm*, which transforms plaintexts to ciphertexts, and the *decryption algorithm*, which transforms ciphertexts to plaintexts. By the discussion above, it is clear that the decription algorithm must employ a *key* 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 which when fed to the decryption algorithm, with the corresponding decryption key, returns the original plaintext. We stress that knowledge of the decryption key is essential for the latter transformation.

A fundamental distiction between encryption schemes refers to 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 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 tommorow, 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 communicated data of length *greater than* n (but still polynomial in n).

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

5.1.2 A Formulation of Encryption Schemes

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

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

1. *On input* 1^n, *algorithm* G (*called the* key generator) *outputs a pair of bit strings.*

2. *For every pair* (e, d) *in the range of* $G(1^n)$, *and for every* $\alpha \in \{0,1\}^*$, *algorithms* E (encryption) *and* D (decryption) *satisfy*

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

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

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

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

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 scheme mentioned above.)

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

Comments: The above definition may be relaxed in several ways without significantly harming its usefulness. For example, we may relax Condition (2) and allow a negligible decryption error (e.g., $\mathsf{Pr}[D_d(E_e(\alpha)) \neq \alpha] < 2^{-n}$). 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 all popular suggestions of encryption schemes.

Another relaxation consists of restricting the domain of possible plaintexts (and ciphertexts). For example, one may restrict Condition (2) to α's of length $\ell(n)$, where $\ell : \mathbb{N} \to \mathbb{N}$ 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 separetly to each block. For more details see Section 5.2.4.

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. Loosely speaking, an encryption scheme is semantically secure if the encryption of a message does not yield any information on the message to an adversary that is computationally restricted (e.g., to polynomial-time). The second definition has a more technical flavour. It interprets security as the infeasibility of distinguishing between encryptions of a given pair of messages. This definition is useful in demonstrating the security of a proposed encryption scheme, and for arguments concerning properties of cryptographic protocols 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 learnt from the ciphertext can be learnt within related complextity also without the ciphertext.

To be somewhat more accurate, semantic security means that whatever can be efficiently computed from the ciphertext, can be efficiently computed given only the length of the plaintext. Note that this formulation does not role out the possibility that the length of the plaintext can be inferred from the ciphertext. Indeed, some information about the length of the plaintext must be revealed by the ciphertext (see Exercise 3). We stress that other than information about the length of the plaintext, the ciphertext is required to yield nothing about the plaintext.

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 $\{X_n\}_{n\in\mathbb{N}}$.

Secrurity holds only for plaintexts of length polynomial in the security parameter. This is captured below by the restriction $|X_n| = \text{poly}(n)$. Note that we cannot hope to provide computational security for plaintexts of unbounded length in the security parameter (see Exercise 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 manisfested in the definition of security. In the latter the adversary, trying to obtain information on the plaintext, is given the encryption key whereas in the former it is not. Thus, the difference between these schemes amounts to a difference in the adversary model (considered in the definition of security). We start by presenting the definition for private-key encryption schemes.

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

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

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

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

Furthermore, we require that the latter circuit family is efficiently constructable from the former one. That is, we require the existence of a probabilistic polynomial-time transformation, T, that for every n, given the description of C_n returns a description of C'_n as above (i.e., $C'_n = T(C_n)$ for every n). *In case C'_n is a random variable, the probability in Eq. (5.2) is also taken over its distribution.*

The function h provides both algorithms with partial information on the plaintext X_n. In addition both algorithms get the length of X_n. These algorithms then try to guess the value $f(X_n)$; namely, they try to infer information about the plaintext X_n. Loosely speaking, in semantically secure encryption scheme the ciphertext does not help in this inference task. That is, the success probability of any efficient algorithm (i.e., the circuit family $\{C_n\}$) that is given the ciphertext, can be matched, upto a negligible fraction, by the success probability of an efficient algorithm (i.e., the circuit family $\{C'_n\}$) that is not given the ciphertext at all. The extra requirement that C_n can be efficiently transformed into C'_n makes the definition stronger, and is done out of philosophical reasons – see discussion below.

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 there exists a polynomial-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, and for every $\{X_n\}_{n\in\mathbb{N}}$, f, h, $p(\cdot)$ and n as in Definition 5.2.1*

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

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

We comment that the encryption-key can be omitted from the input to C'_n, since C'_n 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.

Discussion of some definitional choices

We discuss some subtle issues regarding Definitions 5.2.1 and 5.2.2. The first comment is important, the others can be skipped with little loss. The interested reader is also referred to Exercises 11 and 12 that discuss additional variants of the definition of semantic security.

Effecient transformation of adversaries. Our definitions require that adversaries capturing what can be inferred from the ciphertext be *effectively* transformed into "equivalent" adversaries that operate without being given the ciphertext. This is stronger than only requiring that corresponding "equivalent" adversaries exist. The strenthening seems especially appropriate since we are using a non-uniform model of adversary strategies. Merely saying that polynomial-size circuits that operate without being given the ciphertext do exist is not reassuring enough, since they may be hard to find whereas circuits that operate while being given the ciphertext may be easy to find. The extra requirement guarantess that this cannot be the case: if circuits that operates on the ciphertext are easy to find then so are the "equivalent" circuits that operate without the ciphertext.

Deterministic versus randomized adversaries. Our definitions refer implicitly to deterministic adversaries (modelled by non-uniform families of circuits which are typically assumed to be deterministic). This is in accordance with the general thesis by which the harm of non-uniform adversaries may be maximized by deterministic ones (i.e., by fixing the "worst" coin-sequence). However, we need to verify that a transformation of adversaries (as discussed above) referring to deterministic adversaries can be extended to randomized ones. This is indeed the case; see Exercise 6.

We comment that the above non-uniform formulation is equivalent to a uniform formulation in which the adversaries are given identical auxiliary input: See Exercise 5.

Lack of restrictions on the functions f and g. We do not require that these functions are even computable. This seems strange at first glance. However, as we shall see in the sequel (see also Exercise 9), the meaning of semantic security is essentially that the distribution ensembles $(E(X_n), 1^{|X_n|}, h(X_n))$ and $(E(1^{|X_n|}), 1^{|X_n|}, h(X_n))$ are computationally indistinguishable (and so whatever C_n can compute can be computed by C'_n).

Other modifications of no impact. Actually, inclusion of a-priori information (captured by the function h) does not affect the definition of semantic security: Definition 5.2.1 remains intact if we omit h from the formulation (or consider a constant function). (This can be shown in various ways; e.g., see Exercise 10.) 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.

5.2.2 Indistinguishability of Encryptions

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

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

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

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 7.

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.

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.

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. Similar results hold for public-key encryption schemes (see Exercise 8).

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. Both propositions are in fact stronger than the corresponding direction stated in Theorem 5.2.5. The more useful direction is stated first: it asserts that the technical interpration of security, in terms of ciphertext-indistinguishability, implies the natural notion of sematic security. Thus, the following proposition yields a methodology for designing sematically secure encryption schemes: design and prove your scheme to be ciphertext-indistinguishable, and conclude (by the following) that it is sematically 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 sematically 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 circuit C'_n produced by the transformation T captures the computation of a probabilistic polynomial-time oracle machine that is given oracle access to C_n.*

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 where h is a constant function, X_n is uniformly distributed over a set containing two strings, the function f is Boolean, and the transformation T is not even required to be computable.*

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 polynomial-size circuit family $\{C_n\}$, a polynomial-size circuit family $\{C'_n\}$ so that for every $\{X_n\}_{n\in\mathbb{N}}$, f and h, the circuit C'_n guesses $f(X_n)$ from $(1^{|X_n|}, h(X_n))$ essentially as good as C_n guesses $f(X_n)$ from $(E(X_n), 1^{|X_n|}, h(X_n))$.

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

Indistinguishability of encryptions will be used to prove that C'_n performs essentially as well as C_n. Note that the construction of C'_n does not depend on the functions h and f or on the distribution of messages to be encrypted. Furthermore, C'_n consists of a probabilistic polynomial-time machine that uses C_n as a black-box.

Claim 5.2.6.1: Let $\{C'_n\}$ be as above. Then, for any polynomial p, and all sufficiently large n's

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

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

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

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

Assume, to 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

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

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

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

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

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

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

Proposition 5.2.6 follows. ■

Discussion: 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 proof requires is the values of f and h on a single string, and such values can be incorporated in the description of the circuit D_n.

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

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

We define a random variable X_n which is uniformly distributed over $\{x_n, y_n\}$, and $f:\{0,1\}^* \to \{0,1\}$ so that $f(x_n) = 1$ and $f(y_n) = 0$. Note that $f(X_n) = 1$ with probability 1/2 and is 0 otherwise. (The function h is defined as a constant function.)

We will show that D_n can be transformed into a polynomial-size circuit C_n that guesses the value of $f(X_n)$, from the encryption of X_n, and does so significantly better that with probability $\frac{1}{2}$. This violates (even the restricted form of) semantic security, since no circuit (regardless of its size) can guess $f(X_n)$ better than with probability 1/2 when only given $1^{|X_n|}$ (since given the constant value $1^{|X_n|}$, the value of $f(X_n)$ is uniformly distributed over $\{0,1\}$).

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

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

Claim 5.2.7.1: There exists a polynomial-size circuit family $\{C_n\}$ so that for infinitely many n's

$$\mathsf{Pr}\left[C_n(E_{G_1(1^n)}(X_n))=f(X_n)\right] \;>\; \frac{1}{2} + \frac{1}{2p(n)}$$

Proof: The circuit C_n uses D_n in a straightforward manner: On input $\beta = E(\alpha)$, the new circuit C_n feeds D_n with input β and outputs 1 if D_n outputs 1 (otherwise, C_n outputs 0).[1]

It is left to analyze the success probability of C_n:

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

where the inequality is due to Eq. (5.5). □

In contrast, as observed above, no circuit (regardless of its size) can guess $f(X_n)$ with success probability above 1/2, when given only $1^{|X_n|}$ and $h(X_n)$

[1] We comment that the '1' output by D_n is an indication that α is more likely to be x_n, whereas the output of C_n 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 C_n output x_n if C_n outputs 1 and output y_n otherwise.

(which are both fixed strings that can be incorporated in the circuit). Thus, we have

Fact 5.2.7.2: For every n and every circuit C'_n

$$\mathsf{Pr}\left[C'_n(1^{|X_n|}, h(X_n)) = f(X_n)\right] \le \frac{1}{2}$$

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, the circuit C_n just passes the encryption-key, given as part of its input, to the circuit D_n. The rest of the proof remains intact.

5.2.4 Multiple Messages

The above definitions only refer to the security of a scheme that is used to encrypt a single plaintext (per key generated). Since the plaintext may be longer than the key, these definitions are already non-trivial, and a scheme satisfying them (even in the private-key model) implies the existence of one-way functions (see Exercise 1). Still, in reality, we want to use encryption schemes to encrypt many messages with the same key. We show that in the public-key model, security in the single-message setting (discussed above) implies security in the multiple-message setting (defined below). This is not necessarily true for the private-key model.

5.2.4.1 Definitions

For $\overline{x} = (x^{(1)}, ..., x^{(t)})$, we let $\overline{E}_e(\overline{x})$ denote the concatanation of the results of applying the randomized process E_e to $x^{(1)}, ..., x^{(t)}$. That is, $\overline{E}_e(\overline{x}) = E_e(x^{(1)}), ..., E_e(x^{(t)})$. We stress that in each of the t invocations E_e utilizes independently chosen random coins.

Definition 5.2.8 (semantic security – mulitple messages):

For private-key: *An encryption scheme, (G, E, D), is* semantically secure for multiple messages in the private-key model *if there exists a polynomial-time transformation, T, so that for every polynomial $t(\cdot)$ and every polynomial-size circuit family $\{C_n\}$, 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*

$$\mathsf{Pr}\left[C_n(\overline{E}_{G_1(1^n)}(\overline{X}_n), 1^{|\overline{X}_n|}, h(\overline{X}_n)) = f(\overline{X}_n)\right]$$
$$< \mathsf{Pr}\left[C'_n(1^{|\overline{X}_n|}, h(\overline{X}_n)) = f(\overline{X}_n)\right] + \frac{1}{p(n)}$$

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

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

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

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 infromation about the missing plaintexts (see Exercise 14).

Definition 5.2.9 (indistinguishability of encryptions – mulitple 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* $x_1, ..., x_{t(n)}, y_1, ..., y_{t(n)} \in \{0,1\}^{\text{poly}(n)}$

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

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 In 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 $x_1, ..., x_{t(n)}, y_1, ..., y_{t(n)} \in \{0,1\}^{\text{poly}(n)}$ 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

$$\left|\Pr\left[C_n(H_n^{(i)})=1\right] - \Pr\left[C_n(H_n^{(i+1)})=1\right]\right| > \frac{1}{t(n)\cdot p(n)} \qquad (5.6)$$

We now construct a circuit D_n that, on input e and β, operates as follows.[2] For every $j \leq i$, the circuit D_n generates an encryption of x_j using the encryption key e. Similarly, for every $j > i+1$, the circuit D_n generates an encryption of y_j using the encryption key e. Let us denote the resulting ciphertexts by $\beta_1, ..., \beta_i, \beta_{i+2}, ..., \beta_{t(n)}$. Finally, D_n invokes C_n on input e and $\beta_1, ..., \beta_i, \beta, \beta_{i+2}, ..., \beta_{t(n)}$, and outputs whatever C_n does.

Now, suppose that β is a (random) encryption of x_{i+1} with key e; that is, $\beta = E_e(x_{i+1})$. Then, $D_n(e,\beta) \equiv C_n(e, \bar{h}^{(i+1)}) = C_n(H_n^{(i+1)})$, 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 $D_n(e,\beta) \equiv C_n(e, \bar{h}^{(i)}) = C_n(H_n^{(i)})$. Thus, by Eq. (5.6),

[2] The construction relies on D_n's knowledge of the encryption-key and hence the public-key model is essential for it.

we have

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

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, the analogous result does not hold in the private-key model.

5.2.4.3 In 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 $|x| = |y| = m = \mathrm{poly}(n)$. On the other hand,

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

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 retreive 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 retreive x_2. On input the ciphertexts β_1, β_2, knowing that the first plaintext is x_1, first retreives the pseudorandom sequence (i.e., it is just $r \stackrel{\text{def}}{=} \beta_1 \oplus x_1$), and next retreives 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 truely 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 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 the same part 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 modelling of the adversary. The honest (legitimate) parties are always modelled by uniform complexity classes (most commonly probabilistic polynomial-time).

The notion of *efficiently constructible ensembles*, defined in Section 3.2.3, is centeral 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.

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 probabilistic polynomial-time algorithm A there exists a probabilistic polynomial-time algorithm A' so that for every polynomial t, every polynomial-time constructible ensemble $\{\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})\}_{n\in\mathbb{N}}$, with $|X_n^{(i)}| = \text{poly}(n)$, every polynomial-time computable $h : \{0,1\}^* \rightarrow \{0,1\}^*$, every $f : \{0,1\}^* \rightarrow \{0,1\}^*$, 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 $\overline{E}_e(\overline{x}_n) \stackrel{\text{def}}{=} E_e(x_n^{(1)}), ..., E_e(x_n^{(t(n))})$ 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 $\overline{T} \stackrel{\text{def}}{=} \{\overline{T}_n = \overline{X}_n\overline{Y}_n Z_n\}_{n\in\mathbb{N}}$, with $\overline{X}_n = (X_n^{(1)}, ..., X_n^{(t(n))})$, $\overline{Y}_n = (Y_n^{(1)}, ..., Y_n^{(t(n))})$, and $|X_n^{(i)}| = |Y_n^{(i)}| = \text{poly}(n)$,*

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

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 $Z_n = \overline{X}_n\overline{Y}_n$.

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 also if Definition 5.2.14 is restricted to the special case where $Z_n = \overline{X}_n\overline{Y}_n$, and 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 conventions (of semantic security) in order to account for auxiliary information (required in ciphertext-indistinguishability).

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,

$$A'(1^{|\overline{\alpha}_n|}, h(\overline{\alpha}_n)) = A(G(1^n), \overline{E}_{G(1^n)}(1^{|\overline{\alpha}_n|}), 1^{|\overline{\alpha}_n|}, h(\overline{\alpha}_n)) \tag{5.7}$$

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

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

Proof sketch: Again, assuming towards the contradiction that the claim does not hold, yields an algorithm that distinguishes encryptions of $\overline{X}_n$ from encryptions of $1^{|\overline{X}_n|}$. This algorithm will use auxiliary information $h(\overline{X}_n)$, which is efficiently computable from $Z_n = \overline{X}_n 1^{|\overline{X}_n|}$. Thus, we derive contradiction to Definition 5.2.14 (even under the special case postulated in the theorem).

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 $z = (\overline{x}, 1^{|\overline{x}|})$, the new algorithm computes $u = h(\overline{x})$ and $v = f(\overline{x})$, invokes A, and outputs 1 if and only if $A(e, \overline{E}_e(\overline{\alpha}), 1^{|\overline{x}|}, u) = v$.

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-negligiblly more in one case than in the other. To clarify the point, we define $\Delta_v(\overline{x}_n)$ to be the difference between $\Pr[A(G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{x}_n),h(\overline{x}_n))=v]$ and $\Pr[A(G_1(1^n),\overline{E}_{G_1(1^n)}(1^{|\overline{x}_n|}),h(\overline{x}_n))=v]$. 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 $\Delta(\overline{x}_n) \stackrel{\rm def}{=} \max_v\{\Delta_v(\overline{x}_n)\}$. Again, $\mathsf{E}[\Delta(\overline{X}_n)] > 1/p(n)$, yet given $\overline{x}_n$ we can approximate $\Delta(\overline{x}_n)$ in polynomial-time. Furthermore, we may find a value v so that $\Delta_v(\overline{x}_n) > \Delta(\overline{x}_n) - (1/2p(n))$, with probability at least $1-2^{-n}$. Thus, on input $(e,z,\overline{E}_e(\overline{\alpha}))$, where $z=(\overline{x},1^{|\overline{x}|})$, the new algorithm, denoted D', first computes $u=h(\overline{x})$, estimates $\Delta(\overline{x})$, and finds a v as above. (This is done obliviously of the ciphertext $\overline{E}_e(\overline{\alpha})$, which is only used next.) Next, 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

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

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 $\overline{T} \stackrel{\rm def}{=} \{\overline{T}_n = \overline{X}_n\overline{Y}_nZ_n\}_{n\in\mathbb{N}}$ (as in Definition 5.2.14), a positive polynomial p and infinitely many n's so that

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

Assume, without loss of generality, that $m \stackrel{\rm def}{=} |Z_n| = |\overline{X}_n|$. We define an auxiliary polynomial-time constructible ensemble $\overline{Q} \stackrel{\rm def}{=} \{\overline{Q}_n\}_{n\in\mathbb{N}}$ so that

$$\overline{Q}_n = \begin{cases} 0^mZ_n\overline{X}_n\overline{Y}_n & \text{with probability } \frac{1}{2} \\ 1^mZ_n\overline{Y}_n\overline{X}_n & \text{with probability } \frac{1}{2} \end{cases}$$

[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).

That is, $\overline{Q}_n$ contains $Z_n\overline{X}_n\overline{Y}_n$ in addition to a bit (provided in the m-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. Specifically, define $f:\{0,1\}^*\to\{0,1\}$ so that f returns the first bit of its input; that it $f(\sigma^m abc)=\sigma$, for $a,b,c\in\{0,1\}^m$. Define $h:\{0,1\}^*\to\{0,1\}$ so that h provides the information in the suffix without yielding information on the prefix; that is $h(\sigma^m abc)=abc$ if $\sigma=0$ and $h(\sigma^m abc)=acb$ otherwise. Thus, $h(\overline{Q}_n)=Z_n\overline{X}_n\overline{Y}_n$. We stress that both h and f are polynomial-time computable.

We will show that D' 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 $h(\overline{Q}_n)$), and does so significantly better that 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 $(e,\overline{E}_e(\overline{\alpha}),1^{|\overline{\alpha}|},z\overline{x}\,\overline{y})$, where $\overline{\alpha}=\sigma^m abc$ equals either $0^m z\overline{x}\,\overline{y}$ or $1^m z\overline{y}\,\overline{x}$, algorithm A first extracts $\overline{x},\overline{y}$ and z out of $h(\overline{\alpha})=z\overline{x}\,\overline{y}$, and approximates

$$\Delta(z,\overline{x},\overline{y})\stackrel{\rm def}{=}\Pr\left[D'(z,G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{x}))=1\right]-\Pr\left[D'(z,G_1(1^n),\overline{E}_{G_1(1^n)}(\overline{y}))=1\right]$$

Let $\widetilde{\Delta}(z,\overline{x},\overline{y})$ denote this approximation, and assume the parameters are 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., the estimate is in between). In case $\xi=0$, algorithm A halts with an arbitrary reasonable guess (say a randomly selected bit). (This is done obliviously of the ciphertext $\overline{E}_e(\overline{\alpha})$, which is only used next.) Next, algorithm A extracts the last block of ciphertexts (i.e., $\overline{E}_e(c)$) out of $\overline{E}_e(\overline{\alpha})=\overline{E}_e(\sigma^m abc)$, and invokes D' on input $(z,e,\overline{E}_e(c))$. 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.

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

Proof sketch: We focus on the case in which the approximation of $\Delta(Z_n,\overline{X}_n,\overline{X}_n)$ provided by 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 $\Delta(Z_n,\overline{X}_n,\overline{X}_n)$. It follows that, for every possible $(z,\overline{x},\overline{y})$, so that $\xi=1$ it holds that $\Delta(z,\overline{y},\overline{x})>0$ and

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

$$\begin{aligned}
&= \frac{1}{2} \cdot \Pr\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{y})) = 0\right] \\
&\quad + \frac{1}{2} \cdot \Pr\left[D'(z, G_1(1^n), \overline{E}_{G_1(1^n)}(\overline{x})) = 1\right] \\
&= \frac{1}{2} \cdot (1 + \Delta(z, \overline{y}, \overline{x}))
\end{aligned}$$

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

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

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

$$\frac{1}{3p(n)} \cdot \frac{1 + (2/3p(n))}{2} + (1 - \frac{1}{3p(n)}) \cdot \frac{1}{2}$$

and the claim follows. □

This completes the proof of the opposite direction. ■

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-unform 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 which output a constant fraction of the bits of each resulting number). However, this common practice becomes sound provided one uses pseudorandom generators (as defined in 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, 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. To obtain a stateless encryption scheme, as in our definitions above, we may use a pseudorandom function (see below). But before doing so, let us formalize the above discussion.

Author's Note: *DO IT (i.e., 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 multi-message private-key setting? Same for the public-key setting).

[5] The (private-key) stream-ciphers discussed below are an execption, but– as we point out– they do not adherse to our formulation of encryption schemes.

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

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

1. *On input 1^n, algorithm G outputs a pair of bit strings.*
2. *There exists a polynomially-bounded function $\ell : \mathbb{N} \rightarrow \mathbb{N}$, 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*

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

All conventions are as in Definition 5.1.1.

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 remain as they were above with the modification that we only consider plaintexts of length $\ell(n)$. For example, the analogue of Definition 5.2.1 reads

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

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

where $C'_n \stackrel{\rm def}{=} T(C_n)$ is the circuit produced by T on input C_n.

There are several obvious ways of transforming a block-cipher into a general encryption scheme. The basic idea is to break the plaintexts (for the resulting scheme) into blocks and encode each block separately by using the block-cipher.

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

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.

Construction 5.3.3 (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 consequetive blocks of length $\ell(n)$, while possibly augmenting the last block. Let $\alpha_1,...,\alpha_t$ be the resulting blocks. Then*

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

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

The above construction yields ciphertexts which reveal the exact length of the plaintext. Recall that this is not prohibited by the definitions of security, 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 13. Also, note that the above construction applies even to the special case where ℓ is identically 1.

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

Proof: Assuming towards the contradiction that the encryption scheme (G',E',D') is not secure, we obtain 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 $L(n)$, the security of (G,E,D) is violated by $t(n)\cdot\lceil L(n)/\ell(n)\rceil$. Also, the argument may utilize any of the two notions of security (i.e., semantic security or ciphertext-indistinguishability). ■

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 5.3.5 (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 polynomail-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 orcale machines that obtain a non-uniform poly(n)-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 $z \in \{0,1\}^{p(n)}$,

$$\left|\mathsf{Pr}\left[M^{f}(z)=1\right] - \mathsf{Pr}\left[M^{f_{I(1^n)}}(z)=1\right]\right| < \frac{1}{q(n)}$$

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.

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

Proof: 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 $f:\{0,1\}^n \rightarrow \{0,1\}^n$, 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 $(r_1, f(r_1) \oplus x_1), ..., (r_1, f(r_t) \oplus x_t)$, 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 $f(r_i) \oplus x_i$'s are independently and uniformly distributed, regardless of the x_i's. Now, if the actual scheme is not ciphertext-indistinguishable then for some sequence of distinct 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. ■

Comments. Note that we could have gotten rid of the randomization if we had allowed the encryption algorithm to be history dependent (as discussed 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.

On the other hand, recall that the common practice of using pseudorandom permutations as block-ciphers[7] is NOT secure (e.g., one 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, in a sense, 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.

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*), so that

1. $I(1^n)$ selects a random n-bit long *index* α of a permutation p_α, along with a corresponding trapdoor τ;

2. $S(\alpha)$ randomly *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., $B(\tau, y) = p_\alpha^{-1}(y)$);

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

Let $I_1(1^n)$ denote the first element in the output of $I(1^n)$ (i.e., the index). It is guaranteed that for every polynomial-size circuit family $\{C_n\}$, every polynomial p and all sufficiently large n's

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

That is, C_n fails to invert p_α on $p_\alpha(x)$, wherein 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 5.3.7 (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, it holds that

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

The security of the above public-key encryption scheme follows from the (non-uniform) one-way feature of the collection $\{p_\alpha\}$. We comment that the proof of Theorem 2.5.2 implies that the corresponding hard-core predicate is non-uniformly strong; that is, for randomly chosen α and r, no polynomial-size circuit can predict $b(r)$ given $p_\alpha(r)$ and α, non-negligiblly better than with success probability $1/2$.

Proposition 5.3.8 *Suppose that b is a (non-uniformly strong) hard-core of the collection $\{p_\alpha\}$. Then Construction 5.3.7 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, it suffices to show that one cannot predict which of the two plaintexts is being encrypted significantly better than by a random guess. We conclude by noting that a guess σ' for the plaintext σ, given the ciphertext $(\alpha, E_\alpha(\sigma)) = (\alpha, (f_\alpha(r), \sigma \oplus b(r))$, 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 the probability that $\sigma' = \sigma$, and so the proposition follows. ■

As admitted above, Construction 5.3.7 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 5.3.7 the relation between these lengths equals the security parameter (i.e., n). However, the idea underlying Construction 5.3.7 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 RSA 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, the above conjecture implies that the common practice of randomly padding messages (using padding equal in length to the message) before encrypting them using RSA, results in a secure public-key encryption scheme. That is, we consider the following

Construction 5.3.9 (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 $N = P \cdot Q$, and selecting at random a pair (e, d) so that $e \cdot d \equiv 1 \pmod{(P-1)\cdot(Q-1)}$. 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, ς) (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 5.3.7: a plaintext of length n is encrypted via a ciphertext of length $2n$. Clearly, for every possible $((N,e),(N,d))$ output of G, it holds that

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

The security of the above public-key encryption scheme follows from the large hard-core conjecture for RSA, analogously to the proof of Proposition 5.3.8.

Proposition 5.3.10 *Suppose that the large hard-core conjecture for RSA does hold. Then Construction 5.3.9 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.* This is a special case of the fact that no public-key encryption scheme that employs a deterministic encryption algorithm may be secure. 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!

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 by an independently selected element in the domain of the trapdoor permutation (as done in Construction 5.3.7), we select only one such element per a plaintext string, and provide an additional bit per each bit of the plaintext. These 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 5.3.11 (a public-key encryption scheme): *Let $\{p_\alpha\}$, I, S, F, B and b be as in Construction 5.3.7. 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 5.3.7. 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 $(p_\alpha^{|\sigma|}(r), \oplus G_\alpha(r))$, where*

$$G_\alpha(r) \stackrel{\text{def}}{=} b(r) \cdot b(p_\alpha(r)) \cdots b(p_\alpha^{|\sigma|-1}(r)) \tag{5.8}$$

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 $n + \ell$. 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.

Proposition 5.3.12 *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 samplying 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 5.3.11 constitute a secure public-key encryption scheme.*

Proof: Again, we prove single-message ciphertext-indistinguishability. As in the proof of Proposition 5.3.10, 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 extention 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 RSA. Recall that RSA itself is not secure (as it employs a deterministic encryption algorithm), whereas Randomized RSA (defined above) is not known to be secure under standard assumption such as intractability of factoring (or of inverting the RSA function).[8]

Construction 5.3.13 (The Blum-Goldwasser Public-Key Encryption Scheme): *For simplicity, we present a block-cipher with arbitrary block-length $\ell(n) = \text{poly}(n)$.*

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 outputing the pair $(N,(P,Q))$, where $N = P \cdot Q$.*

Actually, for sake of efficiency, the key-generator also computes $d_P = ((P+1)/4)^{\ell(n)} \bmod P-1$, $d_Q = ((Q+1)/4)^{\ell(n)} \bmod Q-1$, $c_P = Q \cdot (Q^{-1} \bmod P)$, and $c_Q = P \cdot (P^{-1} \bmod Q)$. 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 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 $b_i = \text{lsb}(s_i)$, where $\text{lsb}(s)$ 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 decryption-key $T = (P,Q,N,c_P,d_P,c_Q,d_Q)$, one first retreives 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 $s_1 \leftarrow c_P \cdot s' + c_Q \cdot s'' \bmod N$.*

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

3. *For $i = 1, .., \ell(n)$, compute $b_i = \text{lsb}(s_i)$ 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 mod N it holds that $s \equiv (s^{2^\ell} \bmod N)^{d_P} \pmod{P}$ (and similarly, $s \equiv (s^{2^\ell} \bmod N)^{d_Q} \pmod{Q}$). Encryption amounts to $\ell(n)$ modular multiplications, whereas decryption amounts to $2 + \ell(n)$ such multiplications and 2 modular exponentiations (relative to half-sized moduli). For comparison to Randomized RSA, consider the setting $\ell(n) = n$. The security of the above scheme follows immediately from Proposition 5.3.12 and the fact that lsb is a hard-core for the modular squaring function (and that inverting the latter is computationally equivalent to factoring). Thus we get:

Corollary 5.3.14 *Suppose that factoring is infeasible in the sense that for every polynomial-size circuit $\{C_n\}$, every positive polynomial p and all succiently 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 5.3.13 consititutes a secure public-key encryption scheme.

5.4 * Beyond eavesdropping security

Author's Note: *The following text includes an introduction that is reproduced with little change from Appendix B.1.3, and a plan.*

The above definitions refer 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. Furthermore, the above definitions refer 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 dimention (parameter) being the *type of attack* and the second being its *purpose*.

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 $(\overline{x}_n, \overline{y}_n)$.)

2. *Chosen Plaintext Attacks.* Here the attacker may obtain the encryption of any plaintext of its choice (under the key being attacked). 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 or form a related ciphertext). 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 or modified in violation of non-malleablity).
   (b) In an *a-posteriori chosen* ciphertext attack, the attacker is given the target ciphertext first, but its access to the oracle 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).

Formal definitions of all types of attacks listed above (as well as the purposes listed below) will follow.

Purpose of attacks. Again, the following is not claimed to be exhaustive.

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.

Some known constructions. Before presenting the actual definitions, let us provide an overview on the known results. 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.

Private-key schemes: The private-key encryption scheme based on pseudorandom functions (i.e., Construction 5.3.5), is secure also against a-priori chosen ciphertext attacks.[9]

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[10] 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.*

[9] 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 $y' = x' \oplus f_s(r)$.)

[10] See definition in Section B.2.

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.*
2. *Prove that the PRF-based private-key scheme remains secure under a-priori CMA.*
3. *Discuss the NIZK construction for a-priori CMA.*

Postpone construction of a-posteriori CMA to next subsection.

5.4.4 Non-malleable encryption schemes

Author's Note: *Plan:*

1. *discuss and define,*
2. *prove that*
   (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) *security and non-malleability are equivalent under a-posteriori chosen ciphertext attack.*
3. *Present and analyze the construction for private-key secure under a-posteriori CMA.*
4. *Sketch the solution for public-key (following DDN+Amit).*

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 [188]: 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 relaxion of the secrecy condition. That is, rather than requiring that the ciphertext yields no information on the plaintext, one has to require that such information cannot be efficiently computed from the ciphertext. The latter requirement indeed coincides with the above definition of semantic security.

The notion of public-key encryption scheme was introduced by Diffie and Hellman [61]. First concrete candidates were suggested by Rivest, Shamir and Adleman [179] and by Merkle and Hellman [153]. However, satisfactory definitions of security were presented only a few years afterwards, by Goldwasser and Micali [118]. The two definitions presented in Section 5.2 originate in [118], where it was shown that ciphertext-indistinguishability implies semantic security. The converse direction is due to [154].

Regarding the seminal paper of Goldwasser and Micali [118], 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, definitioanl approaches such as the simulation paradigm, and techniques such as the hybrid argument. The paper's title ("Probabilistic Encryption") is due to the author's realization that public-key encryption schemes in which the encryption algorithm is deterministic cannot be secure in the sense defined in their paper. Indeed, this led the authors to (explicitly) introduce and justify the paradigm of "randomizing the plaintext" as part of the encryption process. Technically speaking, the paper only presents security definitions for public-key encryption schemes, and furthermore some of these definitions are syntactically different from the ones we have presented above (yet, all these definitions are equivalent). Finaly, the term "ciphertext-indistinguishability" used here replaces the (generic) term "polynomial-security" used in [118]. Some of our modifications have already appeared in [92], 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 [118]. Specifically, they constructed a public-key encryption scheme assuming that deciding Quadratic Residiousity modulo composite numbers is intractable. The condition was weaken by Yao [197] who prove that any trapdoor permutation will do. The efficient public-key encryption scheme of Construction 5.3.13 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 [99].

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

*** 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 [164] (which can be constructed under this assumption [75]).

*** The study of *non-malleability* of the encryption schemes, was initiated in [62]. Non-malleable public-key encryption schemes are known to exist assuming the existence of trapdoor permutation [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 paragraphs are merely place-holders.*

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

*** For a detailed discussion of the relationship among the various notions of secure public-key encryption, the reader is referred to [15].

5.5.3 Open Problems

Author's Note: *Incorporate the following text...*

Both constructions of *public-key* encryption schemes secure against chosen ciphertext attacks (mentioned above) are to be considered as plausibility results (which also offer some useful construction paradigms). Presenting "reasonablly-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 "reasonablly-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* [125]: 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 $\{(U_{n+1}^{(1)}, E(U_{n+1}^{(2)}))\}_{n\in\mathbb{N}}$, where $U_{n+1}^{(1)}$ and $U_{n+1}^{(2)}$ 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 sematically secure public-key encryption scheme. (Hint: A plaintext of length exponential in the security parameter allows the adversary to find the decryption key by exhuastive search.)

Exercise 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 sematically secure encryption scheme.

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

Exercise 4: *Deterministic encryption schemes*: Prove that in order to be sematically secure a public-key encryption scheme must have a probabilistic encryption algorithm. (Hint: Otherwise, one can distinguish the encryptions of two candidate plaintexts by computing the unique ciphertext for each of them.)

Exercise 5: Prove that the following definition, in which we use probabilistic polynomial-time algorithms with auxiliary inputs (rather than polynomial-size non-uniform circuits), is equivalent to Definition 5.2.1.

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

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

Same for public-key encryption.

Guideline: The alternative view of non-uniformity, discussed in Section 1.3, is useful here. That is, we can view a circuit family as a sequence of advices given to a universal machine. Thus, the original definition states that advices for a machine that gets the ciphertext can be efficiently transformed into advices for a machine that does not get the ciphertext. However, we can incorporate the transformation program into the second universal algorithm, and so the advices are identical for both machines (and can be viewed as the auxiliary string z in the new formulation). Thus, the original definition is implied by the new definition. To close the gap between the two definitions, one only needs to observe that it suffices to consider one fixed universal machine, A, in the new definition (as any adversarial strategy can be coded in the auxiliary input to this universal machine).

Exercise 6: Prove that a sematically-secure (private-key) encryption scheme satisfies the same requirements with respect to randomized circuits. That is, there exists a polynomail-time transformation, T, so that for every polynomial-size randomized circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \mathrm{poly}(n)$, every pair of polynomially-bounded functions $f, h : \{0,1\}^* \to \{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

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

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

Guideline: Given a randomized family $\{C_n\}$ as above, consider all possible families of deterministic circuits derived by fixing a sequence of coins for each C_n. Note that you should provide one family of randomized circuits, $\{C'_n\}$, to match the randomized family $\{C_n\}$. The alternative formulation of Exercise 5 is useful here (as one may incorporate and extract the coin-sequence in the auxiliary input).

Exercise 7: 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 $z \in \{0,1\}^{\text{poly}(n)}$,

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

(Hint: incorporate z in the circuit C_n.)

Exercise 8: *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 9: *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 polynomail-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, for every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n| = \text{poly}(n)$, and every polynomially-bounded function $h : \{0,1\}^* \to \{0,1\}^*$, the following two ensembles are computationally indistinguishable.

1. $\{C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|}, h(X_n))\}_{n\in\mathbb{N}}$.
2. $\{C'_n(1^{|X_n|}, h(X_n))\}_{n\in\mathbb{N}}$, where $C'_n = T(C_n)$.

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 10: *A variant on Exercise 9*: The current exercise shows that we may drop the auxiliary information provided by the function h without weakeening the definition. Prove that an encryption scheme, (G, E, D), is (semantically) secure (in the private-key model) if and only if the following holds.

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

1. $\{C_n(E_{G_1(1^n)}(X_n), 1^{|X_n|})\}_{n\in\mathbb{N}}$.
2. $\{C'_n(1^{|X_n|})\}_{n\in\mathbb{N}}$, where $C'_n = T(C_n)$.

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

Guideline: Again, we care mainly about the (easier to establish) fact by which 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 combining Exercises 5 and 9: Starting from the above formulation, and using the alternative presentation of Exercise 5, we establish the formulation of Exercise 9 for the special case in which h is constant on X_n. The general case follows, since 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 11: *Another equivalent definition of security*: The following exercise is interesting mainly for historical reasons. In the definition of semantic security appearing in [118], the term $\max_{u,v}\{\mathsf{Pr}[f(X_n)=v|h(X_n)=u]\}$ appears instead of the term $\mathsf{Pr}[C'_n(1^{|X_n|},h(X_n))=f(X_n)]$. That is, it is required that

for every polynomial-size circuit family $\{C_n\}_{n\in\mathbb{N}}$, every ensemble $\{X_n\}_{n\in\mathbb{N}}$, with $|X_n|=\text{poly}(n)$, every pair of polynomially-bounded functions $f,h:\{0,1\}^*\to\{0,1\}^*$, every polynomial $p(\cdot)$ and all sufficiently large n

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

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 $\max_{u,v}\{\mathsf{Pr}[f(X_n)=v|h(X_n)=u]\}\geq\mathsf{Pr}[C'_n(h(X_n),1^n,|X_n|)=f(X_n)]$, for every circuit C'_n). Next note that in the *special case*, in which X_n satisfies $\mathsf{Pr}[f(X_n)=0|h(X_n)=u]=\mathsf{Pr}[f(X_n)=1|h(X_n)=u]=\frac{1}{2}$, for all u's, the above terms are equal (since C'_n 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 12: *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.

There exists a polynomial-time transformation, T, so that for every polynomial-size circuit family $\{C_n\}$, for every $\{(X_n,Z_n)\}_{n\in\mathbb{N}}$, where Z_n may dependent arbitrarily on X_n, and f, $p(\cdot)$ and n as in Definition 5.2.1

$$\mathsf{Pr}\left[C_n(E_{G_1(1^n)}(X_n),1^{|X_n|},Z_n)=f(X_n)\right]$$

$$< \ \Pr\left[C'_n(1^{|X_n|}, Z_n) = f(X_n)\right] + \frac{1}{p(n)}$$

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

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 13: *Hiding partial information about the length of the plaintext*: Using an arbitrary block cipher, construct an encryption scheme that

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

Prove that the resulting encryption scheme inherents the security of the original block-cipher.
(Hint: Just use an adequate padding convention, making sure that it always yields correct decoding.)

Exercise 14: *Known plaintext attacks*: Loosely speaking, in a known palintext attack on a private-key (resp., public-key) encryption scheme the adversary is given some plaintext/ciphertext pairs in addition to some extra ciphertexts (without corresponding plaintexts). Semantic security in this setting means that whatever can be efficiently computed about the missing plaintexts, can be also efficiently computed given only the length of these plaintexts.

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

Exercise 15: *Length parameters*: Assuming the existence of a secure public-key (resp., private-key) encryption scheme, prove the existence of such scheme in which the length of keys equal the security parameter. Show that the length of ciphertexts may be a fixed polynomial in the length of the plaintext.

Author's Note: *First draft written mainly in 1997. Major revision completed in Dec. 1999.*