CHAPTER FIVE

Encryption Schemes

Up to the 1970s, Cryptography was understood as the art of building encryption schemes, that is, the art of constructing schemes allowing secret data exchange over insecure channels. Since the 1970s, other tasks (e.g., signature schemes) have been recognized as falling within the domain of Cryptography (and even being at least as central to Cryptography). Yet the construction of encryption schemes remains, and is likely to remain, a central enterprise of Cryptography.

In this chapter we review the well-known notions of private-key and public-key encryption schemes. More importantly, we define what is meant by saying that such schemes are secure. This definitional treatment is a cornerstone of the entire area, and much of this chapter is devoted to various aspects of it. We also present several constructions of secure (private-key and public-key) encryption schemes. It turns out that using randomness during the encryption process (i.e., not only at the key-generation phase) is essential to security.

Organization. Our main treatment (i.e., Sections 5.1–5.3) refers to security under "passive" (eavesdropping) attacks. In contrast, in Section 5.4, we discuss notions of security under active attacks, culminating in robustness against chosen ciphertext attacks. Additional issues are discussed in Section 5.5.

Teaching Tip. We suggest to focus on the basic definitional treatment (i.e., Sections 5.1 and 5.2.1–5.2.4) and on the the feasibility of satisfying these definitions (as demonstarted by the simplest constructions provided in Sections 5.3.3 and 5.3.4.1). The overview to security under active attacks (i.e., Section 5.4.1) is also recommended. We assume that the reader is familiar with the material in previous chapters (and specifically with Sections 2.2, 2.4, 2.5, 3.2–3.4, and 3.6). This familiarity is important not only because we use some of the notions and results presented in these sections but also because we use similar proof techniques (and do so while assuming that this is *not* the reader's first encounter with these techniques).

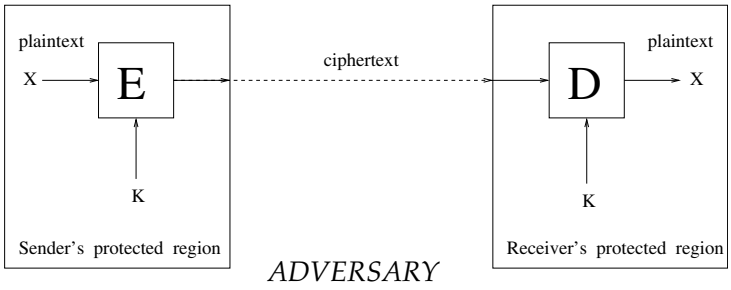
5.1. The Basic Setting

Loosely speaking, encryption schemes are supposed to enable private exchange of information 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 message(s) 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 corresponding ciphertext such that the receiver can retrieve the plaintext from the ciphertext, but the adversary cannot do so. Clearly, something must distinguish the receiver (who is able to retrieve the plaintext from the corresponding ciphertext) from the adversary (who cannot do so). Specifically, the receiver knows something that the adversary does not know. This thing is called a *key*.

An encryption scheme consists of a method of transforming plaintexts into ciphertexts and vice versa, using adequate keys. These keys are essential to the ability to effect these transformations. Formally, these transformations are performed by corresponding algorithms: an *encryption algorithm* that transforms a given plaintext and an adequate (encryption) key into a corresponding ciphertext, and a *decryption algorithm* that given the ciphertext and an adequate (decryption) key recovers the original plaintext. Actually, we need to consider a third algorithm, namely, a probabilistic algorithm used to generate keys (i.e., a *key-generation algorithm*). This algorithm must be probabilistic (or else, by invoking it, the adversary obtains the very same key used by the receiver). We stress that the encryption scheme itself (i.e., the aforementioned three algorithms) may be known to the adversary, and the scheme's security relies on the hypothesis that the adversary does not know the actual keys in use.[1]

In accordance with these principles, an encryption scheme consists of three algorithms. These algorithms are public (i.e., known to all parties). The two obvious algorithms are the *encryption algorithm,* which transforms plaintexts into ciphertexts, and the *decryption algorithm,* which transforms ciphertexts into plaintexts. By these principles, it is clear that the decryption 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 ciphertext that when fed to the decryption algorithm, together with the corresponding

[1] In fact, in many cases, the legitimate interest may be served best by publicizing the scheme itself, because this allows an (independent) expert evaluation of the security of the scheme to be obtained.



The key K is known to both receiver and sender, but is unknown to the adversary. For example, the receiver may generate K at random and pass it to the sender via a perfectly-private secondary channel (not shown here).

Figure 5.1: Private-key encryption schemes: an illustration.

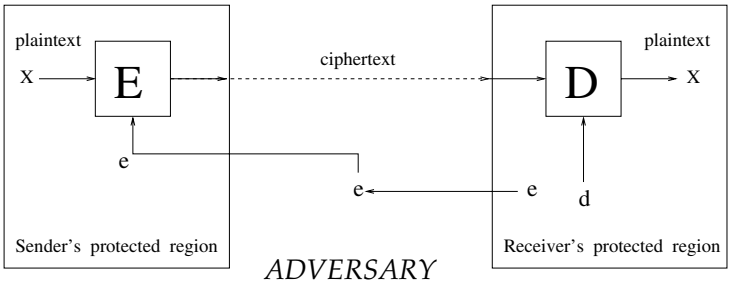
decryption-key, yields the original plaintext. We stress that knowledge of the decryption-key is essential for the latter transformation.

5.1.1. Private-Key Versus Public-Key Schemes

A fundamental distinction between encryption schemes refers to the relation between the aforementioned pair of keys (i.e., the encryption-key and the decryption-key). The simpler (and older) notion assumes that the encryption-key equals the decryption-key. Such schemes are called private-key (or symmetric).

Private-Key Encryption Schemes. 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 (secondary) channel that (unlike the main channel) is assumed to be secure (i.e., it cannot be tapped by the adversary). 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. Assuming that the legitimate parties have agreed on a (secret) key, they can secretly communicate by using this key (see illustration in Figure 5.1): The sender encrypts the desired plaintext using this key, and the receiver recovers the plaintext from the corresponding ciphertext (by using the same key). Thus, private-key encryption is a way of extending a private channel over time: If the parties can use a private channel today (e.g., they are currently in the same physical location) but not tomorrow, then they can use the private channel today to exchange a secret key that they may use tomorrow for secret communication.

A simple example of a private-key encryption scheme is the *one-time pad*. The secret key is merely a uniformly chosen sequence of n bits, and an n-bit long ciphertext is produced by XORing the plaintext, bit-by-bit, with the key. The plaintext is recovered from the ciphertext in the same way. Clearly, the one-time pad provides



The key-pair (e, d) is generated by the receiver, who posts the encryption-key e on a public media, while keeping the decryption-key d secret.

Figure 5.2: Public-key encryption schemes: an illustration.

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 (or information contents) of the data being communicated. By contrast, the rest of this chapter will focus on encryption schemes in which n-bit long keys allow for the secure communication of data having an a priori unbounded (albeit polynomial in n) length. In particular, n-bit long keys allow for significantly more than n bits of information to be communicated securely.

Public-Key Encryption Schemes. A new type of encryption schemes emerged in the 1970s. In these so-called public-key (or asymmetric) encryption schemes, 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 the use of a secure channel. Instead, each party applies the key-generation algorithm to produce a pair of keys. The party (denoted 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. (See illustration in Figure 5.2.)

5.1.2. The Syntax of Encryption Schemes

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

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

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

(encryption) *and* D (decryption) *satisfy*

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

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

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

We stress that Definition 5.1.1 says nothing about security, and so trivial (insecure) algorithms may satisfy it (e.g., $E(e, \alpha) \stackrel{\text{def}}{=} \alpha$ and $D(d, \beta) \stackrel{\text{def}}{=} \beta$). Furthermore, Definition 5.1.1 does not distinguish private-key encryption schemes from public-key ones. The difference between the two types is introduced in the security definitions: In a public-key scheme the "breaking algorithm" gets the encryption-key (i.e., e) as an additional input (and thus $e \neq d$ follows), while in private-key schemes e is not given to the "breaking algorithm" (and thus, one may assume, without loss of generality, that $e = d$).

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

Notation. In the rest of this text, we write $E_e(\alpha)$ instead of $E(e, \alpha)$ and $D_d(\beta)$ instead of $D(d, \beta)$. Sometimes, when there is little risk of confusion, we drop these subscripts. Also, we let $G_1(1^n)$ (resp., $G_2(1^n)$) denote the first (resp., second) element in the pair $G(1^n)$. That is, $G(1^n) = (G_1(1^n), G_2(1^n))$. Without loss of generality, we may assume that $|G_1(1^n)|$ and $|G_2(1^n)|$ are polynomially related to n, and that each of these integers can be efficiently computed from the other. (In fact, we may even assume that $|G_1(1^n)| = |G_2(1^n)| = n$; see Exercise 6.)

Comments. Definition 5.1.1 may be relaxed in several ways without significantly harming its usefulness. For example, we may relax Condition (2) and allow a negligible decryption error (e.g., $\Pr[D_d(E_e(\alpha)) \neq \alpha] < 2^{-n}$). Alternatively, one may postulate that Condition (2) holds for all but a negligible measure of the key-pairs generated by $G(1^n)$. At least one of these relaxations is essential for some suggestions of (public-key) 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 separately to each block. (Note that security of the resulting scheme requires that the security of the length-restricted scheme be preserved under multiple encryptions with the same key.) For more details see Sections 5.2.4 and 5.3.2.

5.2. Definitions of Security

In this section we present two fundamental definitions of security and prove their equivalence. The first definition, called *semantic security,* is the most natural one. Semantic security is a computational-complexity analogue of Shannon's definition of perfect privacy (which requires that the ciphertext yield no information regarding the plaintext). Loosely speaking, an encryption scheme is semantically secure if it is *infeasible* to learn anything about the plaintext from the ciphertext (i.e., impossibility is replaced by infeasibility). The second definition has a more technical flavor. It interprets security as the infeasibility of distinguishing between encryptions of a given pair of messages. This definition is useful in demonstrating the security of a proposed encryption scheme and for the analysis of cryptographic protocols that utilize an encryption scheme.

We stress that the definitions presented in Section 5.2.1 go far beyond saying that it is infeasible to recover the plaintext from the ciphertext. The latter statement is indeed a minimal requirement for a secure encryption scheme, but we claim that it is far too weak a requirement. For example, one should certainly not use an encryption scheme that leaks the first part of the plaintext (even if it is infeasible to recover the entire plaintext from the ciphertext). In general, an encryption scheme is typically used in applications where even obtaining partial information on the plaintext may endanger the security of the application. The question of which partial information endangers the security of a specific application is typically hard (if not impossible) to answer. Furthermore, we wish to design application-independent encryption schemes, and when doing so it is the case that each piece of partial information may endanger some application. Thus, we require that it be infeasible to obtain any information about the plaintext from the ciphertext. Moreover, in most applications the plaintext may not be uniformly distributed, and some a priori information regarding it may be available to the adversary. We thus require that the secrecy of all partial information be preserved also in such a case. That is, given any 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.

Security of Multiple Plaintexts. Continuing the preceding discussion, the definitions are presented first in terms of the security of a single encrypted plaintext. However, in many cases, it is desirable to encrypt many plaintexts using the same encryption-key, and security needs to be preserved in these cases, too. Adequate definitions and discussions are deferred to Section 5.2.4.

A Technical Comment: Non-Uniform Complexity Formulation. To simplify the exposition, we define security in terms of non-uniform complexity (see Section 1.3.3 of Volume 1). Namely, in the security definitions we expand the domain of efficient adversaries (and algorithms) to include (explicitly or implicitly) non-uniform polynomial-size circuits, rather than only probabilistic polynomial-time machines. Likewise, we make

no computational 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 complexity formulation (rather than a uniform one) may only strengthen the definitions, yet it does weaken the implications proven between the definitions because these (simpler) proofs make free usage of non-uniformity. A uniform-complexity treatment is provided in Section 5.2.5.

5.2.1. Semantic Security

A good disguise should not reveal the person's height.
Shafi Goldwasser and Silvio Micali, 1982

Loosely speaking, semantic security means that nothing can be gained by looking at a ciphertext. Following the simulation paradigm, this means that whatever can be efficiently learned from the ciphertext can also be efficiently learned from scratch (or from nothing).

5.2.1.1. The Actual Definitions

To be somewhat more accurate, semantic security means that whatever can be efficiently computed from the ciphertext can be efficiently computed when *given only the length of the plaintext*. Note that this formulation does not rule out the possibility that the length of the plaintext can be inferred from the ciphertext. Indeed, some information about the length of the plaintext must be revealed by the ciphertext (see Exercise 4). We stress that other than information about the length of the plaintext, the ciphertext is required to yield nothing about the plaintext.

In the actual definitions, we consider only information regarding the plaintext (rather than information regarding the ciphertext and/or the encryption-key) that can be obtained from the ciphertext. Furthermore, we restrict our attention to functions (rather than randomized processes) applied to the plaintext. We do so because of the intuitive appeal of this special case, and are comfortable doing so because this special case implies the general one (see Exercise 13). We augment this formulation by requiring that the infeasibility of obtaining information about the plaintext remain valid even in the presence of other auxiliary partial information about the same 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 definition that follows, the information regarding the plaintext that the adversary tries to obtain is represented by the function f, whereas the a priori partial information about the plaintext is represented by the function h. The infeasibility of obtaining information about the plaintext is required to hold for any distribution of plaintexts, represented by the probability ensemble $\{X_n\}_{n\in\mathbb{N}}$.

Security holds only for plaintexts of length polynomial in the security parameter. This is captured in the following definitions by the restriction $|X_n| \leq \text{poly}(n)$, where "poly" represents an arbitrary (unspecified) polynomial. Note that we cannot hope to provide computational security for plaintexts of unbounded length or for plaintexts of length

that is exponential in the security parameter (see Exercise 3). Likewise, we restrict the functions f and h to be *polynomially-bounded,* that is, $|f(z)|, |h(z)| \leq \text{poly}(|z|)$.

The difference between private-key and public-key encryption schemes is manifested in the definition of security. In the latter case, the adversary (which is trying to obtain information on the plaintext) is given the encryption-key, whereas in the former case 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 probabilistic polynomial-time algorithm* A *there exists a probabilistic polynomial-time algorithm* A' *such that for every probability ensemble* $\{X_n\}_{n\in\mathbb{N}}$*, with* $|X_n| \leq \text{poly}(n)$*, every pair of polynomially bounded functions* $f, h : \{0,1\}^* \to \{0,1\}^*$*, every positive polynomial* p *and all sufficiently large* n

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

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

We stress that all the occurrences of X_n in each of the probabilistic expressions refer to the same random variable (see the general convention stated in Section 1.2.1 in Volume 1). The security parameter 1^n is given to both algorithms (as well as to the functions h and f) for technical reasons.[2] The function h provides both algorithms with partial information regarding the plaintext X_n. Furthermore, h also makes the definition implicitly non-uniform; see further discussion in Section 5.2.1.2. In addition, both algorithms get the length of X_n. These algorithms then try to guess the value $f(1^n, X_n)$; namely, they try to infer information about the plaintext X_n. Loosely speaking, in a semantically secure encryption scheme the ciphertext does not help in this inference task. That is, the success probability of any efficient algorithm (i.e., algorithm A) that is given the ciphertext can be matched, up to a negligible fraction, by the success probability of an efficient algorithm (i.e., algorithm A') that is not given the ciphertext at all.

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

[2] The auxiliary input 1^n is used for several purposes. First, it allows smooth transition to fully non-uniform formulations (e.g., Definition 5.2.3) in which the (polynomial-size) adversary depends on n. Thus, it is good to provide A (and thus also A') with 1^n. Once this is done, it is natural to allow also h and f to depend on n. In fact, allowing h and f to explicitly depend on n facilitates the proof of Proposition 5.2.7. In light of the fact that 1^n is given to both algorithms, we may replace the input part $1^{|X_n|}$ by $|X_n|$, because the former may be recovered from the latter in poly(n)-time.

Definition 5.2.2 (semantic security – public-key): *An encryption scheme,* (G, E, D)*, is* semantically secure (in the public-key model) *if for every probabilistic polynomial-time algorithm* A*, there exists a probabilistic polynomial-time algorithm* A' *such that for every* $\{X_n\}_{n\in\mathbb{N}}$*,* f*,* h*,* p*, and* n *as in Definition 5.2.1*

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

Recall that (by our conventions) both occurrences of $G_1(1^n)$, in the first probabilistic expression, refer to the same random variable. We comment that it is pointless to give the random encryption-key (i.e., $G_1(1^n)$) to algorithm A' (because the task as well as the main inputs of A' are unrelated to the encryption-key, and anyhow A' could generate a random encryption-key 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 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. See further discussion following Definition 5.2.4.

5.2.1.2. Further Discussion of Some Definitional Choices

We discuss several secondary issues regarding Definitions 5.2.1 and 5.2.2. The interested reader is also referred to Exercises 16, 17, and 19, which present additional variants of the definition of semantic security.

Implicit Non-Uniformity of the Definitions. The fact that h is not required to be computable makes these definitions non-uniform. This is the case because both algorithms are given $h(1^n, X_n)$ as auxiliary input, and the latter may account for arbitrary (polynomially bounded) advice. For example, letting $h(1^n, \cdot) = a_n \in \{0,1\}^{\mathrm{poly}(n)}$ means that both algorithms are supplied with (non-uniform) advice (as in one of the common formulations of non-uniform polynomial-time; see Section 1.3.3). In general, the function h can code both information regarding its main input and non-uniform advice depending on the security parameter (i.e., $h(1^n, x) = (h'(x), a_n)$). We comment that these definitions are equivalent to allowing A and A' to be *related* families of non-uniform circuits, where by *related* we mean that the circuits in the family $A' = \{A'_n\}_{n\in\mathbb{N}}$ can be efficiently computed from the corresponding circuits in the family $A = \{A_n\}_{n\in\mathbb{N}}$. For further discussion, see Exercise 9.

Lack of Computational Restrictions Regarding the Function f. We do not even require that the function f be computable. This seems strange at first glance because (unlike the situation with respect to the function h, which codes a priori information

given to the algorithms) the algorithms are asked to guess the value of f (at a plaintext implicit in the ciphertext given only to A). However, as we shall see in the sequel (see also Exercise 13), the actual technical content of semantic security is that the probability ensembles $\{(1^n, E(X_n), 1^{|X_n|}, h(1^n, X_n))\}_n$ and $\{(1^n, E(1^{|X_n|}), 1^{|X_n|}, h(1^n, X_n))\}_n$ are computationally indistinguishable (and so whatever A can compute can also be computed by A'). Note that the latter statement does not refer to the function f, which explains why we need not make any restriction regarding f.

Other Modifications of No Impact. Actually, inclusion of a priori information regarding the plaintext (represented by the function h) does not affect the definition of semantic security: Definition 5.2.1 remains intact if we restrict h to only depend on the security parameter (and so only provide plaintext-oblivious non-uniform advice). (This can be shown in various ways; e.g., see Exercise 14.1.) Also, the function f can be restricted to be a Boolean function having polynomial-size circuits, and the random variable X_n may be restricted to be very "dull" (e.g., have only two strings in its support): See proof of Theorem 5.2.5. On the other hand, Definition 5.2.1 implies stronger forms discussed in Exercises 13, 17 and 18.

5.2.2. Indistinguishability of Encryptions

A good disguise should not allow a mother to distinguish her own children.
Shafi Goldwasser and Silvio Micali, 1982

The following technical interpretation of security states that it is infeasible to distinguish the encryptions of two plaintexts (of the same length). That is, such ciphertexts are computationally indistinguishable as defined in Definition 3.2.7. Again, we start with the private-key variant.

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

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

The probability in these 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 11.

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