The Notion of Security for Probabilistic Cryptosystems

(Extended Abstract) *

Silvio Micali† MIT

Charles Rackoff University of Toronto

Bob Sloan‡ MIT

Abstract

Three very different formal definitions of security for public-key cryptosystems have been proposed—two by Goldwasser and Micali and one by Yao. We prove all of them to be equivalent. This equivalence provides evidence that the right formalization of the notion of security has been reached.

1 Introduction

The key desideratum for any cryptosystem is that encrypted messages must be secure. Before one can discuss whether a cryptosystem has this property, however, one must first rigorously define what is meant by security. Three different rigorous notions of security have been proposed. Goldwasser and Micali[5] suggested two different definitions, polynomial security and semantic security, and proved that the first notion implies the second. Yao[8] proposed a third definition, one inspired by information theory, and suggested that it implies semantic security.

Not completely knowing the relative strength of these definitions is rather unpleasant. For instance, several protocols have been proved correct adopting the notion of polynomial security. Are these protocols that are secure with respect to a particular definition or are they secure protocols in a more general sense? In other words, a natural question arises: Which of the definitions is the "correct" one? Even better: How should we decide the "correctness" of a definition?

The best possible answer to these questions would be to find that the proposed definitions—each attempting to be as general as possible—are all equivalent. In this case, one obviously no longer has to decide which one definition is best. Moreover, the equivalence suggests that one has indeed found a strong, natural definition.

In this paper, we show that these notions are *essentially* equivalent. The three originally proposed definitions were not equivalent. However, as we point out, this inequivalence was caused only by some minor technical choices. After rectifying these marginal choices, we succeed in proving the desired equivalences, keeping the spirit of the definitions intact. We believe this to be an essential step in developing theory in the field of cryptography.

*The full version of this work will appear in the *SIAM Journal of Computing*. In the meantime, the full version is available from the authors.

†Supported by NSF grant DCR-8413577 and an IBM Faculty Development Award.

‡Supported by a General Electric Foundation Fellowship and NSF grant DCR-8509905.

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 381-392, 1987.

2 Public Key Scenarios

Let us briefly review what is meant by the notion of public-key cryptography, first proposed by Diffie and Hellman [4] in 1976. As with all cryptography, the goal is that A(lice), by using an encryption algorithm E, becomes able to securely send a message m to B(ob). What is meant by "securely" is that it is impossible for any party T who's tapped A and B's line to figure out information about m from $E(m)$. The distinguishing feature of public-key cryptography is that we require this security property to hold even when T knows the encryption algorithm E.

We believe that until now, the notion of public-key cryptography has not been fully understood. In fact, it is crucial to consider exactly how the communication between A and B establishes the algorithm E. Therefore, we introduce the fundamental notion of a *pass*. We will first explain what passes are, and then explain their implications for security.

2.1 Passes

Within the public-key model, A and B can alternate communicating back and forth as many times as they feel are necessary to achieve security. Call each alternation a *pass*.

Any number of passes are, of course, permissible. We concentrate on what we believe are the two most interesting and important cases, one and three passes. We do not consider more than three passes, because, if trapdoor permutations exist, a well designed probabilistic encryption scheme can achieve as much security as is possible using only three passes.

Three-pass systems

The three-pass case is, perhaps, the most natural to think about. It corresponds to a telephone conversation. A has a message m that she wants to securely communicate to B. A calls up B and says, "I have a message I'd like to send to you." B, so alerted, proceeds to generate an encryption/decryption algorithm pair, (E, D), and tells A, "Please use E to encrypt your message." A then uses E to encrypt her message and tells B "$E(m)$."

Notice the key property of a three-pass system: The message and the encryption algorithm are selected independently of one another. We are nevertheless in a public-key model, since anyone tapping the phone line gets to hear B tell E to A.

One-pass systems

A one-pass system corresponds to what is commonly called a public file system. In the one-pass model, A simply looks up B's public encryption algorithm, E, in a "phone book" and uses it to encrypt her message. (One pass is a slight misnomer. At some point, in what we may view as a preprocessing stage, B must have communicated his encryption algorithm, presumably by telling it to whomever publishes the phone book of encryption algorithms, and thus indirectly to A. "One and a half passes" might be more accurate. "Half" refers to the preprocessing stage that needs to be performed only once.) In this case, the choice of message *can* depend on E.

2.2 Passes and Security

The main result of this paper is:

> GM-security, semantic security, and Y-security (all formally defined in section 3) are equivalent both for one-pass and three-pass cryptosystems.

Interestingly, the equivalence still holds in the one-pass scenario, but the notions of security vary between the one-pass and three-pass scenarios. This point has not been given the proper attention, because people frequently confuse the notion of one-pass public-key cryptography with public key cryptography in general.

The distinction, however, is crucial for avoiding errors, particularly in cryptographic protocols. Let us informally state the two definitions of security that are achievable in the two scenarios if trapdoor permutations exist.

> A 3-pass cryptosystem is secure if, for every message m in the message space, it is impossible to efficiently distinguish an encryption of m from random noise.

> A 1-pass cryptosystem is secure if, for every message m that is efficiently computable on input the encryption algorithm alone, it is impossible to efficiently distinguish an encryption of m from random noise.

In other words, in the one-pass scenario one cannot just blithely write, "For all messages m." For instance, if one closely analyzes all known public-key cryptosystems, it is *conceivable* that if (E, D) is an encryption/decryption pair, then D can be easily computed from $E(D)$. For instance, the constructive reduction of security to quadratic residuosity given by Goldwasser and Micali [5] for their cryptosystem would *vanish* if the encrypted message is allowed to be D itself.[1]

Such problems cannot arise in the three-pass scenario because the encryption algorithm E is selected after and independently of the message m.

In this paper we concentrate on providing the details for the three pass case, and sketch the results for the one pass case in the final section. The reason for this choice is that the definitions of security are much more easily stated for three-pass systems. It is much more convenient to say, "For all messages m," than "For all messages m that are efficiently computable given the encryption algorithm as an input."

3 Notions of security for three-pass systems

In this section we will formally specify our cryptographic scenario, and define the three notions of security. These definitions are the same in spirit as those originally chosen by Goldwasser and Micali and Yao; therefore, we will use either the names they chose or their initials. We will point out explicitly at the end of this section the minor changes we needed to make to reach the right level of generality.

[1] Notice that if Bob publishes an encryption algorithm E in the public file while keeping its associated decryption algorithm D secret, then any other user, being limited to efficient computation and ignorant of D, necessarily selects her message m efficiently from the input E—maybe without even looking at E—and perhaps other inputs altogether independent of (E, D). However, in designing cryptographic protocols, one would often like to be able to transmit things like $E(D)$. For instance, if that type of message were allowed, one would have a trivial solution to the problem of verifiable secret sharing [3].

3.1 Notation and Conventions for Probabilistic Algorithms.

We introduce some generally useful notation and conventions for discussing probabilistic algorithms. (We make the natural assumption that all parties may make use of probabilistic methods.)

We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write "$A(\cdot,\cdot)$" and so on.

"PS" will stand for "probability space"; in this paper we only consider countable probability spaces. In fact, we deal almost exclusively with probability spaces arising from probabilistic algorithms.

If $A(\cdot)$ is a probabilistic algorithm, then for any input i, the notation $A(i)$ refers to the PS which assigns to the string σ the probability that A, on input i, outputs σ. Notice the special case where A takes no inputs; in this case the notation A refers to the algorithm itself, whereas the notation $A()$ refers to the PS defined by running A with no input. If S is a PS, denote by $\mathrm{Pr}_S(e)$ the probability that S associates with element e. Also, we denote by $[S]$ the set of elements which S gives positive probability. In the case that $[S]$ is a singleton set $\{e\}$ we will use S to denote the value e; this is in agreement with traditional notation. (For instance, if $A(\cdot)$ is an algorithm that, on input i, outputs i^3, then we may write $A(2) = 8$ instead of $[A(2)] = \{8\}$.)

If $f(\cdot)$ and $g(\cdot,\cdots)$ are probabilistic algorithms then $f(g(\cdot,\cdots))$ is the probabilistic algorithm obtained by composing f and g (i.e. running f on g's output). For any inputs $x, y, \ldots$ the associated probability space is denoted $f(g(x, y, \ldots))$.

If S is any PS, then $x \leftarrow S$ denotes the algorithm which assigns to x an element randomly selected according to S; that is, x is assigned the value e with probability $\mathrm{Pr}_S(e)$. If F is a finite set, then the notation $x \leftarrow F$ denotes the algorithm which assigns to x an element randomly selected from the PS which has sample space F and the uniform probability distribution on the sample points. Thus, in particular, $x \leftarrow \{0,1\}$ means x is assigned the result of a coin toss.

The notation $\mathrm{Pr}(p(x, y, \ldots) \mid x \leftarrow S; y \leftarrow T; \ldots)$ denotes the probability that the predicate $p(x, y, \ldots)$ will be true, after the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, etc. We use analogous notation for expected value—$\mathrm{Ex}(f(x, y, \ldots) \mid x \leftarrow S; y \leftarrow T; \ldots)$—where now f is a function which takes numerical values.

Let $\mathcal{RA}$ denote the set of probabilistic polynomial-time algorithms. We assume that a natural representation of these algorithms as binary strings is used.

By 1^n we denote the unary representation of integer n, i.e.

$$\underbrace{11\ldots1}_{n}$$

3.2 Cryptographic Scenario

Here we specify those elements that are necessary for all public-key cryptography.

A *cryptographic scenario* consists of the following components:

- A *security parameter* n which is chosen by the user when he creates his encryption and decryption algorithms. The parameter n will determine a number of quantities (length of plaintext messages, overall security, etc.).

- A sequence of *message spaces*, $M = \{M_n\}$ from which all plaintext messages will be drawn. M_n consists of all messages allowed to be sent if the security parameter has been set equal to n. In order to make our notation simpler, (but without loss of generality), we'll assume that $M_n = \{0,1\}^n$. There is a probability distribution on each message space, $\Pr_n : M_n \to [0,1]$ such that $\sum_{m \in M_n} \Pr_n(m) = 1$.

- A *public-key cryptosystem* is an algorithm $\mathcal{C} \in \mathcal{RA}$ that on input 1^n outputs the description of two polynomial-size circuits E and D such that:

  1. E has n inputs and $l(n)$ outputs, and D has $l(n)$ inputs and n outputs. (l is some polynomial that gives the length of the ciphertext.)
  2. E is probabilistic; D is deterministic.
  3. For all $m \in \Sigma^n$, $\Pr(D(\alpha) = m \,|\, (E,D) \leftarrow \mathcal{C}(1^n);\ \alpha \leftarrow E(m)) = 1$.

Notice that $[E(m)]$ is a set which is typically quite large. Our notation requires us to write $\alpha \in [E(m)]$ to refer to α, particular encryption of m. Nevertheless, we will sometimes sloppily write $E(m)$ for a particular encryption of m when the meaning is clear.

3.3 GM-security

This definition is essentially what Goldwasser and Micali [5] called polynomial security.

A *line tapper* is a family of polynomial-size probabilistic circuits $T = \{T_n\}$. Each T_n takes four strings as input and outputs either 0 or 1. However, to make our next equation more readable, we will treat T_n's output as being either its second or third input (0 or 1 respectively).

Definition Let $\mathcal{C}$ be a public-key cryptosystem. $\mathcal{C}$ is *GM-secure* if for all line tappers T and $c > 0$, for all sufficiently large n, for every $m_0, m_1 \in \{0,1\}^n$

$$\Pr(T_n(E, m_0, m_1, \alpha) = m \mid m \leftarrow \{m_0, m_1\}\,; E \leftarrow \mathcal{C}(1^n);\ \alpha \leftarrow E(m)) < \frac{1}{2} + n^{-c}. \qquad (1)$$

Remark: In reading the above definition, one should pay close attention to our notation. Upon casual consideration of Equation 1, one might conclude that there aren't any GM-secure cryptosystems! After all, the definition says that the encryption E must be secure for *any* m_0 and m_1, both of which are given as inputs to the line tapper. What happens if we put $m_0 = D$, a description of the decryption algorithm? The answer to this question is that our notation specifies that *first* we choose m from $\{m_0, m_1\}$ (and thus m_0 and m_1 already had been set), and *then* we choose our encryption algorithm. If $\mathcal{C}$ is GM-secure, then the probability that $\mathcal{C}(1^n)$ assigns to any given output is quite small, say $O(2^{-n})$. Thus there's little worry that $\mathcal{C}$ will just happen to output a decryption algorithm $D = m_0$. Notice how the above definition (via our notation) models the three-pass scenario.

3.4 Semantic Security (3-pass)

Again, this definition is essentially the same as in [5]. It can be viewed as a polynomial-time bounded version of Shannon's "perfect secrecy" [7]. Informally, let f be *any* function defined on a

message space sequence. $f(m)$ constitutes information about the message m. Intuitively, f should be thought of as some particular information about the plaintext that the adversary is going to try to compute from the ciphertext—say the first seventeen bits of the plaintext. A cryptosystem is semantically secure if no adversary, on input $E(m)$ can compute $f(m)$ more accurately than by random guessing (taking into account the probability distribution on the message space).

Definition Let $\mathcal{C}$ be a public-key cryptosystem, and let $M = \{M_n\}$ a sequence of message spaces. Let $\mathcal{F} = \{f_E : M_n \to \Sigma^* \mid E \in [\mathcal{C}(1^n)], n \in \mathbf{N}\}$ be any set of functions on the message spaces. For any value $v \in \Sigma^*$, we denote by $f_E^{-1}(v)$ the inverse image of v; that is, the set $\{m \in M_n \mid f_E(m) = v\}$. Then the probability of the most probable value for $f_E(m)$ is $p_E = \max\left\{\sum_{m \in f_E^{-1}(v)} \Pr_n(m) \mid v \in \Sigma^*\right\}$. p_E is the maximum probability with which one could guess $f_E(m)$ knowing only the probability distribution from which m has been drawn.

$\mathcal{C}$ is *semantically secure* if for all message space sequences M, for all families of functions $\mathcal{F}$, for every family of polynomial-size probabilistic circuits $A = \{A_n(\cdot,\cdot)\}$, for all $c > 0$, and for all sufficiently large n

$$\Pr(A_n(E,\alpha) = f_E(m) \mid m \leftarrow M_n;\ E \leftarrow \mathcal{C}(1^n);\ \alpha \leftarrow E(m)) < p_E + \frac{1}{n^c}. \tag{2}$$

Notice that p_E implicitly depends on n, because E depends on n. Notice also that we quantify over message spaces in order to take into account all possible probability distributions on the messages.

3.5 Y-security (3-pass)

Yao's definition [8] is inspired by information theory, but its context differs from classical information theory in that the communicating agents, A(lice) and B(ob), are limited to probabilistic polynomial-time computations.

An intuitive explanation of Yao's definition is the following: A has a series of n^k messages, selected from a probability space known to both A and B, and an encryption of each message. She wishes to transmit enough bits to B so that he can (in polynomial time with very high probability) compute all the plaintexts. A cryptosystem is Y-secure if the average number of bits A must send B is the same regardless of whether B possesses a copy of the ciphertexts.

We now make this notion precise, first by defining "Alice and Bob," and then eventually defining Y-security itself.

Let $M = \{M_n\}$ be a sequence of message spaces. Each M_n is $\{0,1\}^n$ with a fixed probability distribution. (Note that an information theorist would consider M to be a sequence of *sources*.)

Let $\varepsilon(n)$ be any function that vanishes faster than n^{-c} for all positive c.

For the sake of compactness of notation, the expression $\vec{m}$ will denote a particular series of n^k messages. That is, $\vec{m}$ stands for $m_1, m_2, \ldots, m_{n^k}$.

Let f be any positive function such that $f(n) \leq n$. Intuitively, $f(n)$ is the number of bits per message that A must transmit to B in order for B to recover the plaintexts. Recall that all the messages in M_n have length n.

Definition An $f(n)$ *compressor/decompressor pair* (hereinafter c/d pair) for M is a pair of families of probabilistic polynomial-size circuits, $\{A_n\}$ and $\{B_n\}$, satisfying the following three properties for some constant k and all sufficiently large n:

1. "B_n understands A_n."

$$\Pr(\vec{m} = y \mid m_1 \leftarrow M_n; \ldots; m_{n^k} \leftarrow M_n; \beta \leftarrow A_n(\vec{m}); \quad y \leftarrow B_n(\beta)) = 1 - O(\varepsilon(n)). \tag{3}$$

2. "A_n transmits only $f(n)$ bits per message."

$$\mathrm{Ex}\left[\frac{|\beta|}{n^k} \mid m_1 \leftarrow M_n; \ldots; m_{n^k} \leftarrow M_n; \beta \leftarrow A_n(\vec{m})\right] \leq f(n). \tag{4}$$

3. "The output of A_n can be parsed."
   For all polynomials Q there exists a probabilistic polynomial-time Turing machine S^Q such that S^Q takes as input n and a concatenated string of $Q(n)$ βs, each of which is a good output from A_n, and separates them. That is, its input is $\beta_1\beta_2\ldots\beta_{Q(n)}$ and its output is $\beta_1\#\beta_2\#\ldots\#\beta_{Q(n)}$. We require that

$$\Pr(S^Q \text{ correctly splits } \beta_1\beta_2\ldots\beta_{Q(n)}) = 1 - O(\varepsilon(n)). \tag{5}$$

Remark: The requirement that S^Q exist is a technical requirement. It creates a finite analogue of classical information theory's requirement that messages be transmitted one bit at a time, in an infinite sequence of bits.

We say that the *cost of communicating M is less than or equal to $f(n)$*, in symbols $C(M) \leq f(n)$, if there exists an $f(n)$ c/d pair for M.

We define $C(M) > f(n)$ to be the negation of $C(M) \leq f(n)$—that is, *any* circuits "communicating M" must use at least $f(n)$ bits. The definition of $C(M) = f(n)$ is analogous.

Let $\mathcal{C}$ be a cryptosystem. We define $C(M \mid E_{\mathcal{C}}(M))$, *the cost of communicating M given encryptions from $\mathcal{C}$* in a manner analogous to $C(M)$. The only difference is that now *both A_n and B_n* also get E and the n^k values of some encryption function $E \in [\mathcal{C}(1^n)]$ as inputs. That is, for this definition we must rewrite Equation 3 above to read:

$$\Pr(\vec{m} = y \mid m_1 \leftarrow M_n; \ldots; m_{n^k} \leftarrow M_n; E \leftarrow \mathcal{C}(1^n); \quad \alpha_1 \leftarrow E(m_1); \ldots; \alpha_{n^k} \leftarrow E(m_{n^k}); \quad \beta \leftarrow A_n(E, \vec{m}, \vec{\alpha}); y \leftarrow B_n(E, \beta, \vec{\alpha})) = 1 - O(\varepsilon(n)). \tag{6}$$

An analogous change must also be made to Equation 4.

Notice that for this definition, the probabilities involved must be taken over the different choices of E from $\mathcal{C}$ as well as everything else.

Definition Let $\mathcal{C}$ be a public-key cryptosystem. Fix a sequence of message spaces $M = \{M_n\}$ (and thus the probability distribution on each M_n). We say that $\mathcal{C}$ is *Y-secure with respect to M* if

$$C(M) = C(M \mid E_{\mathcal{C}}(M)) + O(\varepsilon(n)). \tag{7}$$

We say that $\mathcal{C}$ is *Y-secure* if for all M, $\mathcal{C}$ is Y-secure with respect to M.

3.6 The original definitions vs. ours

As we pointed out in the introduction, we made minor changes in cryptographic scenario from [5] and [8]. Here we will spell out what those changes are and why they were made.

Changes to Goldwasser and Micali's Definition

There are two ways a cryptosystem (the server that generates encryption/decryption algorithm pairs) can achieve security:

1. The cryptosystem gets a description of a message space M (and thus its probability distribution) as one of its inputs and will output an encryption/decryption algorithm pair to securely encrypt M.

2. The cryptosystem is told nothing about the message space. The encryption algorithms it outputs are supposed to be secure for every possible message space.

We will call the former cryptosystems *adaptive* and the latter *oblivious*.

Goldwasser and Micali consider adaptive cryptosystems for both of their definitions of security [5]; Yao doesn't make it clear which type of cryptosystem he is assuming for his definition of security [8]. We believe it makes more sense to consider oblivious cryptosystems, for both theoretical and applied reasons.

The theoretical reason for preferring oblivious cryptosystems is that all three definitions of security are equivalent. (See Section 4.) This is a desirable property that fails to hold for adaptive cryptosystems, as we will show in the next section.

The practical reason for preferring oblivious cryptosystems is that, although it is certainly conceivable that having knowledge of the message space would allow one to design a better encryption algorithm, cryptographers have in fact normally tried to design cryptosystems that are secure for all message spaces. For example, consider the cryptosystem based on arbitrary trapdoor predicates proposed by Goldwasser and Micali [5]. Although they only considered security in the adaptive cryptosystem sense, their cryptosystem is in fact secure in the stronger, oblivious sense.

Changes to Yao's Definition

In [8], Yao assumes deterministic private key cryptography, but the definition is immediately extended to probabilistic public-key cryptography.

Yao defines the compressor A and decompressor B to be Turing machines, not circuits. We have switched to circuits because it is not clear that there are *any* secure cryptosystems with respect to probabilistic Turing machines. It might be that one can always achieve greater polynomial-time compression given the ciphertext simply because having a *shared* random (enough) string (in this case the ciphertext!) helps. If it does help, however, having made the compressor and decompressor nonuniform circuits, we can always hardwire in a shared random string of bits.

3.7 Inequivalence of the original definitions

In this section, we point out that, for adaptive cryptosystems, GM-security is a notion stronger than either semantic security or Y-security. We do this in the following two claims, each supported by an informal argument. These claims can be easily transformed to theorems after formalizing the discussed security notions in terms of adaptive cryptosystems, a tedious effort once we have realized that the adaptive setting is not the "right" one.

Claim 1 *If any GM-secure adaptive public-key cryptosystem exists, then there exist adaptive public-key cryptosystems that are semantically secure but not GM-secure.*

Let $\mathcal{C}(\cdot,\cdot)$ be any GM-secure (and thus semantically secure) adaptive cryptosystem. We'll construct a $\mathcal{C}'(\cdot,\cdot)$ that is still semantically secure, but is not GM-secure.

$\mathcal{C}'$ behaves identically to $\mathcal{C}$ for all message spaces, except for the message space $\{0,1\}^n$ with uniform probability distribution. In this case, $\mathcal{C}'$ runs $\mathcal{C}$ to compute an encryption algorithm E, and then outputs the algorithm E' defined by:

$$E'(x) = \begin{cases} 0^n & \text{if } x = 0^n \\ E(x) \neq 0^n & \text{otherwise} \end{cases} \tag{8}$$

$\mathcal{C}'$ is clearly not GM-secure, because, for the special message space described above, there is a message, 0^n, which is easily distinguished from other messages by its encryption. However, $\mathcal{C}'$ is still semantically secure. The message 0^n has such a low probability weight that it won't give an adversary any significant advantage—on average—in computing a function of the plaintext on input the ciphertext. □

Claim 2 *If any GM-secure adaptive public-key cryptosystem exists, then there exist adaptive public-key cryptosystems that are Y-secure but not GM-secure.*

We construct exactly the same $\mathcal{C}'$ as we did for the previous claim. $\mathcal{C}'$ is of course not GM-secure. However, the "weak message" has such low probability that it basically doesn't affect the average number of bits necessary to communicate messages from the message space. Thus $\mathcal{C}'$ is Y-secure. □

4 Main Results

GM-security, semantic security, and Y-security are all equivalent. We have chosen to prove this equivalence by showing that GM-security is equivalent to Y-security and that GM-security is equivalent to semantic-security. In this extended abstract, we simply state our theorems. The proofs may be found in the full paper. In fact, we prove only three of the four necessary implications. The proof that GM-security implies semantic security may be found in [5].

Theorem 1 *Let $\mathcal{C}$ be a public-key cryptosystem. If $\mathcal{C}$ is semantically secure, then $\mathcal{C}$ is GM-secure.*

The proof of this theorem is quite simple. The key idea is that if a cryptosystem is not GM-secure, then there exist two messages, m_1 and m_2, which we can easily distinguish. If we make a new message space in which these are the only messages, then given only a ciphertext, one has a better than random chance of figuring out which of the two plaintext messages this ciphertext represents.

Theorem 2 *Let $\mathcal{C}$ be a cryptosystem. If $\mathcal{C}$ is Y-secure, then $\mathcal{C}$ is GM-secure.*

Theorem 3 *Let $\mathcal{C}$ be a public-key cryptosystem. If $\mathcal{C}$ is GM-secure, then $\mathcal{C}$ is Y-secure.*

5 One-Pass Scenarios

In this section we present the proper definitions for one-pass cryptography, and then go on to show that these definitions are all equivalent to one another. (They are *not* equivalent to the three-pass definitions.) These definitions are all considerably more complicated than the analogous definitions for the three-pass scenario.

5.1 GM-security (1-pass)

As discussed above in Section 2.2, for a one-pass cryptosystem, we must change from requiring security "for all messages m," to requiring security for every message m that is efficiently computable on input the encryption algorithm alone. In order to do this, we introduce an adversary called a message finder.

A *message finder* is a family of polynomial-size probabilistic circuits $F = \{F_n(\cdot)\}$ each of which takes the description of an encryption algorithm as its input and has two messages of length n as its output. Intuitively, on input E, F_n tries to find m_0 and m_1 such that it's easy for a fellow adversary (a line tapper) to distinguish encryptions of m_0 from encryptions of m_1.

Definition Let $\mathcal{C}$ be a public-key cryptosystem. $\mathcal{C}$ is *GM-secure (one-pass)* if for all message finders F, line tappers T, and $c > 0$, for all sufficiently large n,

$$\Pr(T_n(E, m_0, m_1, \alpha) = m \mid E \leftarrow \mathcal{C}(1^n); m_0, m_1 \leftarrow F_n(E); \\ m \leftarrow \{m_0, m_1\}; \alpha \leftarrow E(m)) \leq \frac{1}{2} + n^{-c}. \tag{9}$$

5.2 Semantic Security (1-pass)

To change the definition of semantic security to fit the one-pass scenario, we need to introduce something like the message finders of the previous section. For semantic security, however, we're concerned not with finding two "weak" messages, but rather with the probability distribution of the entire message space. Thus our second adversary will not pick out particular messages, but instead set the probability distribution of the message space. Furthermore, we now explicitly give the other adversary a description of that probability distribution.

A *message space enemy* is a family of polynomial-size probabilistic circuits $B = \{B_n(\cdot)\}$. Each B_n takes the description of a encryption algorithm as its input, and outputs the description of a probabilistic Turing machine $N()$. N outputs elements of $\{0,1\}^n$ with some probability distribution.

As in the three-pass definition, we let V be any set and let $\mathcal{F} = \{f_n^E : M_n \to V \mid E \in [\mathcal{C}(n)]\}$ be any set of functions. Again set p_N^E to be the probability of the most probable value for $f(m)$; set $p_n^E = \max\{\sum_{m \in f_n^{E-1}(v)} \Pr_n(m) \mid v \in V\}$.

Definition Let $\mathcal{C}$ be a public-key cryptosystem. $\mathcal{C}$ is *semantically secure* if for every message space enemy B, family of polynomial-size probabilistic circuits $A = \{A_n(\cdot,\cdot,\cdot)\}$, and $c > 0$, for all sufficiently large n

$$\Pr(A_n(E,N,\alpha) = f_n^E(m) \mid E \leftarrow \mathcal{C}(1^n);\ N \leftarrow B_n(E); \\ m \leftarrow N();\ \alpha \leftarrow E(m)) \ < \ p_n^E + \frac{1}{n^c}. \tag{10}$$

5.3 Y-security (1-pass)

The changes that must be made to the definition of Y-security are completely analogous to the changes we made to the definition of semantic security.

5.4 Equivalence

The proofs that the three definitions of security are all equivalent are quite similar to the proofs for the three-pass case.

Acknowledgment

We wish to thank Shafi Goldwasser and Shimon Even for insight and helpful discussions about the notions of security, and general help in preparing this paper.

References

[1] R. Boppana, private communication.

[2] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics* 23 (1952).

[3] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, 1985, pp. 383–95.

[4] W. Diffie and M. E. Hellman, "New direction in cryptography," *IEEE Trans. Inform. Theory,"* Vol. IT-22, No. 6, 1976, pp. 644–54.

[5] S. Goldwasser and S. Micali, "Probabilistic Encryption," *JCSS*, vol. 28, No. 2, April 1984, pp. 270–99.

[6] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed., McGraw-Hill, New York, 1976.

[7] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Tech. J.*, 28 (1949), pp. 656–749.

[8] A. C. Yao, "Theory and Applications of Trapdoor Functions (extended abstract)," *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, 1982, pp. 80–91.