Homework 3 for 6.875

Teacher: Silvio Micali, TA: Yevgeniy Dodis

Due Date: **Wednesday, October 21**

Main Problem

Prove that the following 3 notions of security for probabilistic encryption schemes are equivalent:

1. **GM-security** (aka *polynomial indistinguishability*, proposed by Goldwasser and Micali).
2. **Semantic Security** (proposed by Goldwasser and Micali).
3. **Y-security** (proposed by Yao).

That is, any cryptosystem secure under any one of these notions is secure under all of them. For simplicity, prove the equivalence of these notions in the *three-pass* scenario.

Starting from the next page, we provide you with the necessary background on all three notions of security in the three-pass scenario. Make sure your solutions are precise, but try to *briefly* explain first the intuition of what you are going to do, for each of the implications you are able to prove. Also, split your solutions into clearly marked theorems (each proven on a separate sheet of paper) of the form:

Theorem: $A \Longrightarrow B$.[1]

Extra Credit

You can get extra credit (or a substitute for one of the equivalence directions that you are not able to prove) by solving the following 2 problems.

Extra Problem 1

Try to formalize the notion of a *trapdoor permutation*, whose informal description is the following:

> *bijective function that is easy to compute, hard to invert but there exists a secret (necessarily hard to find!) such that the function is easy to invert if the secret is known.*

(It might help to think whether RSA or exponentiation modulo a prime are good candidates.)

Extra Problem 2

Assume that there exists s GM-secure system $\mathcal{C}$ (under three-pass, for example). Show that there exists a GM-secure system $\mathcal{C}'$ that can be completely broken once the encryption of the secret key is known.

[1] meaning that a cryptosystem secure under type A is also secure under type B, where $A, B \in$ {GM-security, Semantic Security, Y-security}.

1 Various Definitions of Security

1.1 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*. For this assignment, we concentrate on the case of three passes.[2]

The three-pass case 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.

1.2 Circuits

For technical reasons, we will view an efficient algorithm as being a family of *probabilistic polynomial-size circuits* rather than a probabilistic polynomial-time Turing machine.

A family of *probabilistic polynomial-size circuits* is a collection $\mathcal{C}$ of probabilistic Turing machines $\{M_n, n \geq 1\}$ such that there exists a polynomial p such that for all $n \geq 1$,

1. The description of M_n, $|M_n|$, is at most $p(n)$.

2. The running time of M_n on inputs of length n is at most $p(n)$.

Given an input x of length n, the output of $\mathcal{C}$ on x is $M_n(x)$. The difference between a family of circuits and a Turing machine is that the algorithms for various n in a family of circuits need not be "uniform", i.e. one might not find a uniform description of the algorithm for all the input lengths, as is the case for a Turing machine.

1.3 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 $\sum_{m \in M_n} \Pr_n(m) = 1$.

[2]Note, if trapdoor permutations exist, a well designed probabilistic encryption scheme can achieve as much security as is possible using only three passes.

- 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.

1.4 GM-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 $m_0, m_1 \in \{0,1\}^n$

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

Remark: In reading the above definition, one should pay close attention to the 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.

1.5 Semantic Security

This notion can be viewed as a polynomial-time bounded version of Shannon's "perfect secrecy". 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. Let $\tilde{p} = \mathbf{E}[p_E]$ be the expected value of p_E over the random choice of $E \leftarrow \mathcal{C}(1^n)$.

$\mathcal{C}$ 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

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

Notice that p_E (and thus $\tilde{p}$) 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.

1.6 Y-security

This definition 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. 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."

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

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 $C(M) \le f(n)$, if there exists an $f(n)$ c/d pair for M.

We define $C(M) > f(n)$ to be the negation of $C(M) \le f(n)$—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:

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

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.