CSCI 5440: Cryptography **Lecture 3**
The Chinese University of Hong Kong 26 January 2011

1 Indistinguishability for multiple encryptions

We now have a reasonable encryption scheme, which we proved is message indistinguishable, and so semantically secure, assuming pseudorandom generators exist. Let's recall how it works. Given a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^m$, encryption and decryption work like this:

$$Enc(K, M) = G(K) + M \qquad Dec(K, C) = G(K) + C$$

One thing that is unrealistic about our scheme is that it allows Alice and Bob to communicate only one message. In reality, we would expect them to exchange multiple messages. Is the scheme secure in such a scenario?

How do we define security for multiple messages? The setting is the same as before, but now there are c messages that Alice and Bob want to communicate. Even defining the functionality of such a scheme raises several questions: Are the messages always sent from Alice to Bob, or can they also go the other way? Are they always received in the same order in which they are sent? Should Bob be required to decode the second message even if he hasn't received the first one? And so on.

Let us not worry too much about formalizing all these functionality requirements for now, since eventually we will be able to achieve even the strongest ones. It is a good exercise for you to do at home.

Instead let us focus on the security requirement. Just as before, our notion of security will require that Eve cannot distinguish between scenarios where different messages are sent:

Definition 1. A private-key encryption scheme (Enc, Dec) with key length k and message length m is (s, ε) message indistinguishable for c messages if for all messages $M_1, \dots, M_c, M'_1, \dots, M'_c \in \{0,1\}^m$:

$$|\Pr[A(Enc(K, M_1), \dots, Enc(K, M_c)) = 1] - \Pr[A(Enc(K, M'_1), \dots, Enc(K, M'_c)) = 1]| \leq \varepsilon$$

for every circuit A of size at most s.

You can also define semantic security for multiple messages and show it is an equivalent notion.

Does the encryption scheme that we saw achieve semantic security? It is not difficult to see that it doesn't – think of what happens when the same message is sent twice: Then the encodings of the two messages are the same, so by adding them up Eve can find out that the same message has been sent twice.

It seems that our source of trouble is that we are using the same portion of the pseudorandom generator output to encrypt multiple messages. One way around this is to use the first m bits to encrypt the first message, the second m for the second one, and so on. (This amounts to treating the c messages as one long message of length cm.) However, an annoying aspect of this solution is that the encoder and decoder need to keep state: In particular Bob must know when he is looking at the ith message so he can start his decoding at the correct position. If the messages are always sent sequentially from Alice to Bob and they are received in the same order this is not a problem, but that is a fairly unrealistic scenario.

Can we design a stateless encryption scheme for multiple messages?[1] As a first observation, notice that no *deterministic* scheme is up to the task, because it is vulnerable to the same-message attack that broke our original single message scheme. So we need to use randomness in the encryption. But before we go onto designing such a scheme, it will be convenient to introduce a new, stronger notion of security.

2 The chosen plaintext attack

For a moment let's go back to the setting of single message encryption. Recall the objective of message indistinguishability: Eve cannot tell apart the encryption of any two messages. An implicit assumption that we made when we defined message indistinguishability is that Eve is a *passive adversary*: She gets to observe all she wants (apart from the secret key), but does not get to interact with the scheme in any way.

Here is an imaginary scenario in which Eve can do more. Perhaps what Alice is sending to Bob is the encryption of messages Eve already knows (say the contents of an inflammatory email that Eve wrote about Bob). Can this information help Eve recover the content of other messages that Alice sends to Bob?

Here is how we formalize this scenario. We want to allow Eve the ability to encrypt some messages (without knowing the key), and ask if this ability helps her distinguish other encryptions. One possibility that comes to mind is to give Eve the private key so she can encrypt her own messages. But once Eve has the key, she can also decrypt anything she wants. How do we allow Eve to encrypt *without* access to the private key?

We will do so with the help of a theoretical device called an *oracle*. Informally, an oracle is a black box that allows us to perform a given functionality without getting any information about how that functionality is implemented. In programming we do this all the time: Suppose you are writing a computer program that does data analysis and you need to calculate the eigenvalues of some matrix. Instead of writing your own function f for this, it makes more sense to use a library: In such a case you only care about the interface of f (i.e. how many arguments it takes and what those parameters mean), but not about its implementation. This is the kind of function that the oracle represents; but in cryptography (and complexity theory) we make the stronger requirement that not only do you not care about the implementation of f, but you cannot figure out what it is even if you wanted to.

Formally, an *oracle circuit* C with oracle access to $f\colon \{0,1\}^k \to \{0,1\}^m$, denoted by C^f, is an augmented kind of circuit which, in addition to its AND, OR, and NOT gates, also has special f-gates, each of which has k inputs and m outputs. The size of C^f is the number of AND, OR, and f-gates. A computation of an f-gate is simply an evaluation of f on its inputs. We want to allow f to be randomized; in that case, we will assume that different calls to f make use of independent randomness in their evaluations.

[1] In our formalism, the encryption algorithm *Enc* is stateless by definition, since it only gets a key and a message as inputs.

To describe the chosen plaintext attack, we will give Eve access to an *encryption oracle*. This is simply an oracle that computes the function $f(M) = Enc(K, M)$. This oracle gives Eve the ability to encrypt arbitrary messages but without seeing the key.

Definition 2. A private-key encryption scheme (Enc, Dec) (with key length k and message length m) is (s, ε) *message indistinguishable for a chosen plaintext attack* (in short *CPA message indistinguishable*) if for every oracle circuit $A^?$ of size at most s and every pair of messages M, M':

$$|\Pr[A^{Enc(K,\cdot)}(Enc(K, M)) = 1] - \Pr[A^{Enc(K,\cdot)}(Enc(K, M')) = 1]| \leq \varepsilon.$$

In this definition, the probability is taken over the choice of key $K \sim \{0,1\}^k$, the randomness of Enc, and the randomness of A. We use $Enc(K, \cdot)$ to denote the function that on input M, outputs $Enc(K, M)$.

A chosen plaintext attack allows Eve to encrypt arbitrary messages of her choice. After seeing all these encryptions, can Eve tell the encryptions of M and M' apart? If the encryption scheme is deterministic, she certainly can: She merely needs to ask the oracle for the encryptions of M and M' and see which one she holds in her hand. But if the scheme is randomized, it could be that although she asks the oracle to give her encryptions of M, the oracle returns a different one every time, and this may not give her much information about the encryptions she is trying to distinguish.

It turns out that CPA message indistinguishability is a fairly strong requirement: In particular it implies indistinguishability for multiple messages.

Claim 3. *If (Enc, Dec) is (s, ε) CPA message indistinguishable, then it is $(s - cm, c\varepsilon)$ message indistinguishable for c messages.*

Proof. We prove the contrapositive. Suppose (Enc, Dec) is not $(s, c\varepsilon)$ message indistinguishable for c messages. Then there exist two sequences of c messages $(M_1, \dots, M_c)$ and $(M'_1, \dots, M'_c)$ and a circuit A of size s such that

$$|\Pr[A(Enc(K, M_1), \dots, Enc(K, M_c)) = 1] - \Pr[A(Enc(K, M'_1), \dots, Enc(K, M'_c)) = 1]| > c\varepsilon \quad (1)$$

We will apply the hybrid argument. Consider the following sequence of distributions $H_0, H_1, \dots, H_c$:

$$\begin{array}{lllll} H_0: & Enc(K, M_1), & Enc(K, M_2) & \cdots, & Enc(K, M_c) \\ H_1: & Enc(K, M'_1), & Enc(K, M_2) & \cdots, & Enc(K, M_c) \\ H_2: & Enc(K, M'_1), & Enc(K, M'_2) & \cdots, & Enc(K, M_c) \\ \vdots & & \vdots & & \\ H_c: & Enc(K, M'_1), & Enc(K, M'_2) & \cdots, & Enc(K, M'_c) \end{array}$$

Equation (1) says that $|\Pr[A(H_0) = 1] - \Pr[A(H_c) = 1]| > c\varepsilon$. By the triangle inequality, there must exist some i, where $1 \leq i \leq c$, such that

$$|\Pr[A(H_{i-1}) = 1] - \Pr[A(H_i) = 1]| > \varepsilon$$

Notice that the distributions H_{i-1} and H_i only differ in position i: They are both of the form

$$(Enc(K, M'_1), \dots, Enc(K, M'_{i-1}), Enc(K, M), Enc(K, M_{i+1}), \dots, Enc(K, M_c))$$

where $M = M_i$ in H_{i-1} and $M = M'_i$ in H_i.

Using A, we now design a circuit $B^{Enc(K,\cdot)}$ that CPA distinguishes messages M_i and M'_i. On input a ciphertext C, B calls its oracle on inputs $M'_1, \ldots, M'_{i-1}, M_{i+1}, \ldots, M_c$ to obtain ciphertexts $C'_1, \ldots, C'_{i-1}, C_{i+1}, \ldots, C_c$, respectively and outputs $A(C'_1, \ldots, C'_{i-1}, C, C_{i+1}, \ldots, C_c)$. By construction, B is an oracle circuit of size at most s such that

$$\begin{aligned} &\Pr[B^{Enc(K,\cdot)}(Enc(K, M_i)) = 1] = \Pr[A(H_{i-1}) = 1] \\ \text{and} \quad &\Pr[B^{Enc(K,\cdot)}(Enc(K, M'_i)) = 1] = \Pr[A(H_i) = 1] \end{aligned}$$

and so

$$|\Pr[A^{Enc(K,\cdot)}(Enc(K, M_i)) = 1] - \Pr[A^{Enc(K,\cdot)}(Enc(K, M'_i)) = 1]| > \varepsilon. \qquad \square$$

3 Towards CPA security

We have now reduced the task of constructing an encryption scheme that is indistinguishable for multiple messages to the task of constructing CPA message indistinguishable encryption. The construction will make use of a powerful cryptographic primitive called a *pseudorandom function.*

To gain some intuition about how to get CPA security, let's start with the ingredient that we have, namely a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^t$. We observed that the naive use of the output $G(K)$ as a pad to mask the message is inadequate because the scheme is deterministic, and so the encryption of the same message always comes out identical. Here is an idea: Think of the output of G as a stream of pseudorandom bits. Our current scheme takes the message M and adds it to the first m bits of $G(K)$. How about, instead of using the first m bits, we take a random offset r and output

$$(M_1 + G(K)_r, M_2 + G(K)_2, \ldots, M_m + G(K)_{r+m-1})$$

where $G(K)_i$ is the ith bit of the output of G? (if $r + m - 1 > t$, you can wrap around to the initial part of $G(K)$.)

Now the previous attack doesn't work, because different encryptions will use different values of r, which won't always hit the same part of the pseudorandom stream $G(K)$. However, we have a problem: How can Bob decrypt without knowing r? So it makes sense to include r as part of the encryption too:

$$Enc(K, M) = (r, M_1 + G(K)_r, M_2 + G(K)_2, \ldots, M_m + G(K)_{r+m-1})$$

where r is a random number between 1 and t. Now Bob can decrypt using the algorithm

$$Dec(K, (r, C_1, \ldots, C_m)) = (C_1 + G(K)_r, \ldots, C_m + G(K)_{r+m+1}).$$

How secure is this scheme? It depends on the value of t. One attack that Eve can always run when she wants to CPA distinguish between $C = Enc(K, M)$ and $C' = Enc(K, M')$ is to ask the oracle for $1/t$ encryptions of M. With constant probability, one of these encryptions will be identical to C, while we expect the chances of Eve hitting C' to be much smaller, on the order of 2^{-m}. So if t is reasonable – say polynomial in k – then Eve would be able to implement the attack in polynomial time and get constant distinguishing advantage.

To make the scheme more secure, it seems that what we want is a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^t$ where t is very large, say super-polynomial in k. Can we obtain such a pseudorandom generator? Last lecture we saw a technique that converts a pseudorandom generator $G_0\colon \{0,1\}^k \to \{0,1\}^{k+1}$ into one that produces many bits of output. However, we had to pay a price: Each time we wanted an extra bit we had to do another evaluation of G_0, so if we want t bits we must evaluate G_0 t times. It means that Alice and Bob must do an amount of work that is larger than the one we allotted Eve to do the inversion, and so this scheme does not meet our standards of being simultaneously efficient and secure.

The reason that a G with many output bits is too inefficient to evaluate is because the length extension of pseudorandom generators from last lecture was sequential: To obtain the rth bit of $G(K)$, we have to first evaluate all the $r-1$ bits before it. Can we come up with a parallel construction instead? The answer is yes, but to understand why it will help to shift perspective a bit. Instead of thinking of the output $G(K)$ as a sequence of t bits, it will be more helpful to think of it as a *function* that given an index r outputs the value $G(K)_r$.

4 Pseudorandom functions and CPA security

Definition 4. A collection of functions $\{F_K\colon \{0,1\}^k \to \{0,1\}^m \mid K \in \{0,1\}^k\}$ is called an (s,ε) *pseudorandom function family* if for every oracle circuit $A^?$ of size at most s,

$$|\Pr_{K\sim\{0,1\}^k}[A^{F_K}=1] - \Pr_R[A^R=1]| \le \varepsilon,$$

where R is a random function from $\{0,1\}^k \to \{0,1\}^m$.

It is common to abuse notation and to say "pseudorandom function" for a random member of a pseudorandom function family. Let's explain this definition. Going back to the analogy with pseudorandom generators, we can think of F_K as a list of all its 2^k values $F_K(0^k), F_K(0^{k-1}1), \dots, F_K(1^k)$, called the *truth table* of F_K. If we put them together, we obtain a list of $k2^k$ bits. The definition essentially postulates that this list of $m2^k$ bits when K is random should be computationally indistinguishable from a truly random list of as many bits, i.e. from the truth table of a truly random function.

Notice that this is a very strong requirement: There are 2^k functions F_K in the pseudorandom family $\{F_K\}$, while there are 2^{m2^m} possible functions R. Although 2^k is a tiny fraction of 2^{m2^k}, the definition asks that the two collections of functions be indistinguishable!

Notice also that the distinguisher A here is given oracle access to the function it is supposed to distinguish. In contrast, when we defined pseudorandom generators, we gave the distinguisher access to the output of the generator. The reason for this is technical: It is infeasible to feed the whole description of the function (which is $m2^m$ bits long) as the input to an "efficient" distinguisher A (whose size would be polynomial in m). So we provide the function as an oracle instead. The distinguisher is allowed to evaluate the oracle in arbitrary places and use the results of previous evaluations in arbitrary ways, but it cannot evaluate the mystery function more than s times, as the size of the distinguisher is at most s.

Let us now construct CPA secure encryption from an efficiently computable pseudorandom function family $\{F_K\}$. The idea is to use the output of F_K evaluated at a random place as a pad to mask the message. Since the domain of F_K is so large, even when F_K is evaluated many times, it is unlikely to hit the same seed twice, thwarting the attack we had when a pseudorandom generator was used in lieu of $\{F_K\}$.

Formally, consider the following encryption scheme (Enc, Dec):

$$Enc(K, M) = (S, F_K(S) + M) \qquad Dec(K, (S, C)) = (F_K(S) + C).$$

Here S denotes a random string in $\{0,1\}^k$ (chosen by Alice). We now prove the security of this scheme.

Claim 5. *If $\{F_K\}$ is an (s, ε) pseudorandom function family, then (Enc, Dec) is $(s, 2\varepsilon + 2s/2^m)$ CPA message indistinguishable.*

The idea of the proof is that as long as the adversary A who is trying to CPA distinguish between the encryptions of M and M' invokes the encryption oracle on distinct values of S (that A has no control over), what he observes are messages masked with distinct pseudorandom strings. If F_K is pseudorandom, this should look like completely random information to him and so he gets essentially no help from the oracle in distinguishing the two encryptions.

Proof. For contradiction, suppose (Enc, Dec) is not $(s, 2\varepsilon + s/2^m)$ CPA message indistinguishable, so there exists an oracle circuit $A^?$ of size at most s and a pair of messages M, M' so that

$$|\Pr[A^{Enc(K,\cdot)}(Enc(K, M)) = 1] - \Pr[A^{Enc(K,\cdot)}(Enc(K, M')) = 1]| > 2\varepsilon + 2s/2^m. \tag{2}$$

The first step is to replace the pseudorandom function F_K in Enc with a truly random function R and argue that this does not affect the distinguishing probability by much. Let $REnc$ be an imaginary encryption scheme whose key is a truly random function R and $REnc(R, M) = (S, R(S)+M)$.

We claim that

$$|\Pr[A^{Enc(K,\cdot)}(Enc(K, M)) = 1] - \Pr[A^{REnc(R,\cdot)}(REnc(R, M)) = 1]| \le \varepsilon \tag{3}$$

If this was not the case, then the $\{F_K\}$ could not be (s, ε)-pseudorandom, for the following oracle circuit B^F would distinguish it:

> B^F: First, choose a random $S \sim \{0,1\}^k$ and run $A^?(S, F(S)+M)$. Whenever $A^?$ makes an oracle query Q, respond by $(T, F(T) + Q)$ where T is a random string in $\{0,1\}^k$.

By construction, $B^?$ has size s, B^{F_K} has the same distribution as $A^{Enc(K,\cdot)}(Enc(K, M))$, and B^R has the same distribution as $A^{REnc(R,\cdot)}(REnc(R, M))$, so $|\Pr_{K\sim\{0,1\}^k}[A^{F_K} = 1] - \Pr_R[A^R = 1]| > \varepsilon$ and F_K is not (s, ε)-pseudorandom. So equation (3) must hold, and by analogy we also get that

$$|\Pr[A^{Enc(K,\cdot)}(Enc(K, M')) = 1] - \Pr[A^{REnc(R,\cdot)}(REnc(R, M')) = 1]| \le \varepsilon.$$

Using (2) and the triangle inequality, we have that

$$|\Pr[A^{REnc(R,\cdot)}(REnc(R, M)) = 1] - \Pr[A^{REnc(R,\cdot)}(REnc(R, M')) = 1]| > 2s/2^m.$$

Expanding the expression for $REnc(R, M)$ and $REnc(R, M')$,

$$|\Pr[A^{REnc(R,\cdot)}(S, R(S) + M) = 1] - \Pr[A^{REnc(R,\cdot)}(S, R(S) + M') = 1]| > 2s/2^m. \tag{4}$$

Next we argue that

$$|\Pr[A^{REnc(R,\cdot)}(S, R(S) + M) = 1] - \Pr[A^{REnc(R,\cdot)}(S, T + M) = 1]| \le 2s/2^m.$$

where $T \in \{0,1\}^m$ is a random string. Informally, replacing the random function used to encrypt M by a one-time pad does not make much of a difference. This makes sense because as long as the oracle does not return an answer of the form $(S, \cdot)$, $R(S)$ is independent from all the oracle answers and so it can be replaced by a random string T. Since $A^?$ makes at most s queries and in each query, the chances of an answer of the form $(S, \cdot)$ is at most 2^{-m}, by a union bound the probability of distinguishing the two should be at most $2s/2^m$.

Here is a more formal version of the same argument. Let E denote the event that in at least one of the queries made by $A^?(S, R(S)+M)$ and $A^?(S, R(S)+M')$, the oracle $REnc(R, \cdot)$ returns an answer of the form $(S, \cdot)$. Notice that this event does not depend on the input of $A^?$, and by a union bound $\Pr[E] \leq 2s/2^m$. Conditioned on E, the distributions $A^{REnc(R,\cdot)}(S, R(S)+M)$ and $A^{REnc(R,\cdot)}(S, T+M)$ are identical, and

$$|\Pr[A^{REnc(R,\cdot)}(S, R(S)+M) = 1] - \Pr[A^{REnc(R,\cdot)}(S, T+M) = 1]| \leq \Pr[E] \leq 2s/2^m.$$

By the same argument we get an analogous inequality for M' and using (4) we get that

$$|\Pr[A^{REnc(R,\cdot)}(S, T+M) = 1] - \Pr[A^{REnc(R,\cdot)}(S, T+M') = 1]| > 0$$

which is impossible, because the strings $T+M$ and $T+M'$ are identically distributed. $\square$