Fundamentals of Cryptography — Handout 5.

Yu Yu

CPA Security and Theoretical Constructions of Pseudorandom Functions.

1 Security under Chosen-Plaintext Attacks (CPA)

All aforementioned security notations consider a relatively weak adversary who only passively eavesdrops on the communication (except for the part that the adversary gets to choose the plaintexts that are to be encrypted). In this lecture, we formally introduce a more powerful attack, namely a chosen-plaintext attack (CPA). The basic idea is that the adversary A is now allowed to ask for encryptions of multiple messages that he chooses on-the- fly in an adaptive manner. This is formalized by allowing A to interact with (more formally, we refer to as "oracle access") an encryption oracle $\mathsf{Enc}_k(\cdot)$, where A can choose inputs at his choices and observe the corresponding outputs under Enc_k accordingly in a black-box way (secret key k any intermediate results during the computation are hidden from A). Following standard notation in computer science, we denote by $\mathsf{A}^{\mathcal{O}(\cdot)}$ (which we call an oracle machine) an algorithm A given access to an oracle $\mathcal{O}$, and thus in this case we denote the computation of A with access to an encryption oracle by $\mathsf{A}^{\mathsf{Enc}_k(\cdot)}$. When A queries its oracle by providing it with a plaintext message m as input, the oracle returns a ciphertext $c \leftarrow \mathsf{Enc}_k(m)$ as the reply. When $\mathsf{Enc}_k(\cdot)$ is probabilistic, the oracle uses fresh random coins each time it answers a query. We first define an experiment for any private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, any adversary A, and any security parameter n as following.

The CPA indistinguishability experiment $\mathsf{PrivK}^{cpa}_{\mathsf{A},\Pi}(n)$.

1. A random key k is generated: $k \leftarrow \mathsf{Gen}(1^n)$.
2. The adversary A is given input 1^n and oracle access to $\mathsf{Enc}_k(\cdot)$, and outputs a pair of messages m_0, m_1 of the same length.
3. A random bit $b \leftarrow \{0,1\}$ is chosen, and then a challenge ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to A.
4. The adversary A continues to have oracle access to $\mathsf{Enc}_k(\cdot)$, and outputs a bit b'.
5. The output of the experiment is defined to be 1 iff $b' = b$ (indicating that A succeeded), and 0 otherwise.

Definition 1 (CPA security)**.** A private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions under a chosen-plaintext attack (or is CPA- secure) if for all probabilistic polynomial-time adversaries A there exists a negligible function negl such that

$$\Pr[\mathsf{PrivK}^{cpa}_{\mathsf{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

where the probability is taken over the random coins used by A, as well as the random coins used in the experiment.
Note that any scheme that is CPA secure is clearly secure in presence of eavesdropping adversaries (see Definition 2 in Handout 3) since $\mathsf{PrivK}^{eav}_{\mathsf{A},\Pi}(n)$ is a special case of $\mathsf{PrivK}^{cpa}_{\mathsf{A},\Pi}(n)$ where the adversary does not use the oracle access to $\mathsf{Enc}_k(\cdot)$ at all.

Impossibility of deterministic CPA encryption schemes. It is not hard to see that any deterministic encryption scheme (i.e., $\mathsf{Enc}_k(\cdot)$ is a deterministic function for fixed k) is not CPA secure. This is because on any pair of messages (m_0,m_1) an adversary can simply query encryption oracle to get $c_0 := \mathsf{Enc}_k(m_0)$ and $c_1 := \mathsf{Enc}_k(m_1)$, where one of the two must identical to the challenge ciphertext.

On the necessity of CPA security in practice. Refer to the KL book (page 83) for detailed discussions.

2 Pseudorandom Functions and Construction of CPA Secure Encryptions

Next we introduce pseudorandom functions (PRFs) and then show to construct CPA secure encryptions from PRFs.

Pseudorandom Functions In previous lectures we used pseudorandom generators to obtain security in the presence of eavesdroppers. Now, consider a two-input function $F : \{0,1\}^{\kappa(n)} \times \{0,1\}^n \to \{0,1\}^{l(n)}$, where the first input k is called the key, and the second input x is just called the input. In general, the secret key k will be randomly chosen and then fixed, and thus F can be viewed as a family of functions indexed by k $\{F_k : \{0,1\}^n \to \{0,1\}^{l(n)} \mid k \in \{0,1\}^{\kappa(n)}\}$, where $F_k(x) \stackrel{\text{def}}{=} F(k,x)$. For simplicity, in many occasions we will assume that F is length-preserving so that the key, input, and output lengths of F are all the same, i.e., $\kappa(n) = l(n) = n$, where every fixed key k corresponds to a function $F_k(\cdot)$ mapping n-bit strings to n-bit strings.

Definition 2 (Pseudorandom functions)**.** Let $F : \{0,1\}^{\kappa(n)} \times \{0,1\}^n \to \{0,1\}^{l(n)}$ be an efficient keyed function ($F_k(x) \stackrel{\text{def}}{=} F(k,x)$). We say F is a pseudorandom function if for all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that:

$$|\Pr[\mathsf{D}^{F_k(\cdot)}(1^n) = 1] \; - \; \Pr[\mathsf{D}^{R(\cdot)}(1^n) = 1]| \; \leq \; \mathsf{negl}(n)$$

where $k \leftarrow \{0,1\}^{\kappa(n)}$ and $R(\cdot)$ is a function chosen uniformly at random from the set of all functions mapping n-bit strings to $l(n)$-bit strings, and the probability is taken over the coins for selecting k, $R(\cdot)$ as well as the internal random coins of D.

Some intuitions. Informally, we call F pseudorandom if the function F_k (for randomly-chosen key k) is indistinguishable from a random function $R(\cdot)$ with the same domain and range as F_k, where PPT adversaries are given only oracle access to the two functions. From a mathematical point of view, we can consider $R(\cdot)$ as a random function from the set of all functions mapping n-bit strings to $l(n)$-bit strings $\mathcal{R} = \{\{0,1\}^n \to \{0,1\}^{l(n)}\}$. The cardinality of $\mathcal{R}$ is $2^{l(n)\cdot 2^n}$ (viewing the function as a large look-up table that stores $R(x)$ in the row of the able labelled by x) and we know many functions from $\mathcal{R}$ not efficiently computable in polynomial-time. However, the oracle access to R can be efficiently simulated if the number of queries is polynomially bounded: one just needs to record all the previous queries (x_1,$R(x_1)$), $\cdots$, (x_i,$R(x_i)$) which is empty at the beginning, and

on every query x_{i+1}, if x_{i+1} equals any of $x_1, \cdots, x_i$, returns the corresponding output. Otherwise, returns a random string $R(x_{i+1}) \xleftarrow{\$} \{0,1\}^{l(n)}$, and adds $(x_{i+1}, R(x_{i+1}))$ to the record list.

Construction 2.1 (CPA-secure encryption from PRFs)**.** *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function. Define a private-key encryption scheme with key length $\kappa(n) = n$ and message length $l(n) = n$ as follows:*

1. *(Key generation). On input 1^n, Gen outputs a string $k \in \{0,1\}^n$ uniformly at random, i.e., $k \xleftarrow{\$} \{0,1\}^n$*
2. *(Encryption). On input $k \in \{0,1\}^n$ and every plaintext $m \in \{0,1\}^n$, Enc_k chooses a random $r \xleftarrow{\$} \{0,1\}^n$ and outputs ciphertext $c := (r, F_k(r) \oplus m)$.*
3. *(Decryption). On input $k \in \{0,1\}^n$ and every ciphertext $c = (r, s) \in \{0,1\}^n \times \{0,1\}^n$, Dec_k outputs $m := F_k(r) \oplus s$.*

Theorem 1. *If F is a pseudorandom function, then Construction 2.1 (Construction 3.25 from the KL book) is a fixed-length private-key encryption scheme with length parameter $l(n) = n$ that has indistinguishable encryptions under a chosen-plaintext attack.*

Proof. See the proof of Theorem 3.26 from the KL book. □

3 Pseudorandom Functions from Pseudorandom Generators

Construction 3.1 (the Goldreich-Goldwasser-Micali Construction)**.** *Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a length-doubling pseudorandom generator. Define $G_0 : \{0,1\}^n \to \{0,1\}^n$ and $G_1 : \{0,1\}^n \to \{0,1\}^n$ such that $G_0(k)$ and $G_1(k)$ equal the first and second n bits of $G(k)$ respectively, i.e., $G(k) = (G_0(k), G_1(k))$. Then, function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ that on input $k \in \{0,1\}^n$ and $x = x_1 \dots x_n \in \{0,1\}^n$ defined as*

$$F_k(x) := G_{x_n}(G_{x_{n-1}}(\cdots G_{x_2}(G_{x_1}(k)) \cdots)) \qquad (1)$$

is a pseudorandom function.
The evaluation of the function F can be visualized using a binary tree of depth n (the GGM tree), with a copy of the generator G at each node. The root receives the input k and passes its outputs $G_0(k)$ and $G_1(k)$ to its two children, and it takes the path identified by the value of x_1, at node that corresponds to $G_{x_1}(k)$ it takes the path by the value of x_2 to reach node $G_{x_2}(G_{x_1}(k))$, repeat the above process until it gets to the leaf node, and produces the corresponding string as the output for $F_k(x)$.

We first prove a useful technical lemma below:

Lemma 2 (Parallel Repetition of PRGs on Independent Seeds)**.** *Let $G : \{0,1\}^n \to \{0,1\}^{m(n)}$ be a (t, ε)-secure pseudorandom generator computable in time t_g. For any parameter $k = \mathsf{poly}(n)$, and define $G^k : \{0,1\}^{kn} \to \{0,1\}^{km}$ as*

$$G^k(x_1, \dots, x_k) := (\ G(x_1), G(x_2), \dots, G(x_k)\ ) \qquad (1)$$

Then, G^k is a $(t - O(k \cdot t_g), k\varepsilon)$-secure pseudorandom generator.

Proof. We will show that if G^k is not a $(t - O(k \cdot t_g), k\varepsilon)$-secure psedorandom generator, then G cannot be a (t, ε) pseudorandom generator. We will give the proof by a hybrid argument. If G^k is not a $(t - O(k \cdot t_g), k\varepsilon)$-secure pseudorandom generator, then there exists an algorithm D of running time at most $(t - O(k \cdot t_g)$, which distinguishes the output of G^k on a random seed, from a truly random km-bit string drawn from U_{km}, i.e.

$$\left|\Pr\left[\mathsf{D}(G(U_n^1), \ldots, G(U_n^k)) = 1\right] - \Pr\left[\mathsf{D}(U_m^1, \ldots, U_m^k) = 1\right]\right| \;>\; k\varepsilon$$

where U_n^1, ..., U_n^k are i.i.d. to U_n. Define the hybrid distributions $H_0, \ldots, H_k$, where in every H_i we relplace the first i outputs of the pseudorandom generator by truly random strings, i.e.,

$$H_i = (U_m^1, \ldots, U_m^i, G(U_n^{i+1}), \ldots, G(U_n^n))$$

We thus have by triangle inequality

$$\sum_{i=0}^{k-1} |\; \Pr[\mathsf{D}(H_i) = 1] - \Pr[\mathsf{D}(H_{i+1}) = 1]| \;\geq\; |\; \Pr[\mathsf{D}(H_0) = 1] - \Pr[\mathsf{D}(H_k) = 1]| \;>\; k\varepsilon$$

By an averaging argument there exists $i \in \{0, 1, \ldots, k-1\}$ such that

$$|\; \Pr[\mathsf{D}(H_i) = 1] - \Pr[\mathsf{D}(H_{i+1}) = 1]| \;>\; \varepsilon$$

Define $\mathsf{D}'(y) \stackrel{\text{def}}{=} \mathsf{D}(U_m^1, \ldots, U_m^i, y, G(U_n^{i+2}), \ldots, G(U_n^n))$, we have

$$\left|\; \Pr[\mathsf{D}'(G(U_n)) = 1] - \Pr[\mathsf{D}'(U_m) = 1] \;\right| = \;|\; \Pr[\mathsf{D}(H_i) = 1] - \Pr[\mathsf{D}(H_{i+1}) = 1]| \;>\; \varepsilon$$

where the running time of D' is the sum of that of D (i.e., $t - O(k \cdot t_g)$) and at most k invocations of G (i.e., $k \cdot t_g$) which is at most t. We complete the proof by reaching a contradiction. □

We will first consider a slightly simpler case which illustrates the main idea. We prove that if G is (t, ε) pseudorandom and runs in time t_g, then the concatenated output of all the leaves of the GGM tree of depth l, is $(t - O(2^l \cdot t_g), l2^l \cdot \varepsilon)$ pseudorandom. Note that the security bound deteriorates exponentially with respect to l, so the the result is only meaningful for small l.

Theorem 3 (The GGM construction with small depth)**.** *Suppose that $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a (t, ε) pseudorandom generator and G is computable in time t_g. For parameter $l = O(\log n)$ and define $F_k : \{0,1\}^l \to \{0,1\}^n$ as $F_k(y) := G_{y_l}(G_{y_{l-1}}(\cdots G_{y_2}(G_{y_1}(k)) \cdots))$ Then $\overline{G} : \{0,1\}^n \to \{0,1\}^{2^l \cdot n}$ defined as*

$$\overline{G}(k) := (F_k(0^l), F_k(0^{l-1}1), \ldots, F_k(1^l))$$

is a $(t - O(2^l \cdot t_g), l \cdot 2^l \cdot \varepsilon)$ pseudorandom generator.
Note that the above essentially says that the concatenated outputs of F_k on all possible inputs are jointly distinguishable from uniform, and thus it implies that F_k is a pseudorandom function.

Proof. Let $G^k(\cdot)$ be defined as in Equation (1), and we know by Lemma 2 that $G^k(\cdot)$ is (t', ε')-secure pseudorandom generator for $t' = t - O(k \cdot t_g)$ and $\varepsilon' = k\varepsilon$. Assume towards contradiction that there exists D of running time $(t - O(2^l \cdot t_g)$ such that

$$\left|\Pr\left[\mathsf{D}(\;\overline{G}(U_n)\;) = 1\right] - \Pr\left[\mathsf{D}(U_{2^l n}) = 1\right]\right| \;>\; l \cdot 2^l \cdot \varepsilon \;\;.$$

Define hybrid distributions as the following:

$$\begin{aligned} H_0 &= G^{2^{l-1}}(G^{2^{l-2}}(\ldots G^2(G(U_n))\ldots)) = \overline{G}(U_n) \\ H_1 &= G^{2^{l-1}}(G^{2^{l-2}}(\ldots G^2(U_{2n})\ldots)) \\ &\vdots \\ H_{l-1} &= G^{2^{l-1}}(U_{2^{l-1}n}) \\ H_l &= U_{2^l n} \end{aligned}$$

and we know by averaging there exists some $i \in \{0, 1, 2, \ldots, l-1\}$ such that

$$|\Pr[\mathsf{D}(\ H_i\ ) = 1] - \Pr[\mathsf{D}(H_{i+1}) = 1]| \ > \ 2^l \cdot \varepsilon\ .$$

by defining $\mathsf{D}'(y) \stackrel{\text{def}}{=} G^{2^{l-1}}(\ G^{2^{l-2}}(\ \ldots\ G^{2^{i+1}}(y)\ldots\ )\ )$ the above yields

$$\left|\Pr\left[\mathsf{D}'(\ G^{2^i}(U_{2^i n})\ ) = 1\right] - \Pr\left[\mathsf{D}'(\ U_{2^{i+1}n}\ ) = 1\right]\right| \ > \ 2^l\ \cdot\varepsilon \ > \ 2^i\ \cdot\varepsilon\ .$$

where the running time of D' is no greater than $t - O(2^i \cdot t_g)$, namely, the sum of that of D (i.e., $t - O(2^l \cdot t_g)$) and that for evaluating function $G^{2^{l-1}}(\ G^{2^{l-2}}(\ \ldots\ G^{2^{i+1}}(\cdot)\ldots\ )\ )$ which is $O((2^l - 2^i) \cdot t_g)$. This contradicts to the $(t - O(2^i \cdot t_g), 2^i\varepsilon)$-security of G^{2^i} by Lemma 2, which completes the proof. □

Theorem 4 (The general case)**.** *If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a (t, ε) pseudorandom generator and G is computable in time t_g, then F is a $(t' = t/O(nt_g), \varepsilon \cdot n \cdot t')$ secure pseudorandom function.*

Proof. We assume for contradiction that F is not a $(t' = t/O(nt_g), \varepsilon{\cdot}n{\cdot}t')$ secure pseudorandom function, and will show that this implies $G^{t'}$ is not a $(t - O(t'{\cdot}t_g), \varepsilon{\cdot}t')$ secure pseudorandom generator (see Lemma 2). The assumption that F is not (t', ε') secure, refers to that there exists algorithm A of running time at most t' which distinguishes F_k on a random seed $k \stackrel{\$}{\leftarrow} \{0,1\}^n$ from a random function R, i.e.

$$\left|\Pr_{k \leftarrow U_n}\left[A^{F_k(\cdot)}(1^n) = 1\right] - \Pr_{R}\left[A^{R(\cdot)}(1^n) = 1\right]\right| > \varepsilon{\cdot}n{\cdot}t'$$

We consider hybrids $H_0, \ldots, H_n$ as in the proof of Theorem 3 (let $l = n$). H_0 is the distribution of F_K for $K \sim U_n$ and H_n is the uniform distribution over all functions from $\{0,1\}^n \to \{0,1\}^n$. For every $i \in \{0, 1, \ldots, n\}$, parse the $2^n n$-bit H_i into 2^n n-bit strings in the left-to-right order of leaf nodes such that $H_0(\cdot)$ corresponds to function $F_K(\cdot)$ and $H_n(\cdot)$ is the same as $R(\cdot)$.

$$H_i = (H_i(0^l), H_i(0^{l-1}1), \ldots, H_i(1^l))$$

As before, there exists i such that

$$\left|\Pr_{h \leftarrow H_i}\left[A^{h(\cdot)}(1^n) = 1\right] - \Pr_{h \leftarrow H_{i+1}}\left[A^{h(\cdot)}(1^n) = 1\right]\right| > \varepsilon \cdot t'$$

However, now we can no longer use A to construct a distinguisher for G^{2^i} as in Theorem 3 since i may now be as large as n. An important observation is that since A has running time t', it can

make at most t' queries to the function oracle. Since the (at most) t' queries made by A will be paths in the tree from the root to the leaves, they can contain at most t' nodes at depth $i+1$. Hence, to use A, we only need to simulate a function distributed according to H_i or H_{i+1} on t' inputs.

We will use A to construct an algorithm D which distinguishes the output of $G^{t'}(U_{nt'})$ from $U_{2nt'}$. D takes as input a string of length $2t'n$, which we parse as t' pairs of t strings

$$(z_{1,0}, z_{1,1}), \ldots, (z_{t',0}, z_{t',1})$$

where every $z_{j,k} \in \{0,1\}^n$. When queried on an input $x \in \{0,1\}^n$, D will pick a pair $(z_{j,0}, z_{j,1})$ according to the first i bits of x (i.e. choose the randomness for the node at depth i which lies on the path), and then choose $z_{k,x_{i+1}}$. In particular, D works as below:

1. Start with counter $ctr = 0$, and define integer function $P(x_1, ..., x_i)$ initialized to -1 (meaning that no x with prefix $x_1, ..., x_i$ has been queried before).
2. Simulate the oracle of A as follows: for every query $x = x_1 \ldots x_n$
   - Check if $P(x_1, \ldots, x_i) = -1$ or not.
   - If $P(x_1, \ldots, x_i) = -1$, set $P(x_1, \ldots, x_{i+1}) = ctr + 1$ and set $ctr := ctr + 1$.
   - Answer the query made by A as $G_{x_n}(\cdots G_{i+2}(z_{P(x_1,\ldots,x_{i+1}),x_{i+1}}) \cdots)$.
3. Return the final output given by A.

Then, if input is a random string from $U_{2nt'}$, the answers received by A are as identical to that given by a oracle function distributed according to $H_{i+1}(\cdot)$. Hence,

$$\Pr[\mathsf{D}(U_{2nt'}) = 1] = \Pr_{h \leftarrow H_{i+1}}\left[\mathsf{A}^{h(\cdot)}(1^n) = 1\right]$$

Similarly, if the input is from $G^{t'}(U_{nt'})$, then the view of A is the same as in the case with an oracle function distributed according to H_i. This gives

$$\Pr[\mathsf{D}(G^{t'}(U_{nt'})) = 1] = \Pr_{h \leftarrow H_i}\left[A^{h(\cdot)}(1^n) = 1\right]$$

Hence, D distinguishes the output of $G^{t'}$ from a random string with probability $\varepsilon \cdot t'$. The running time of D is no greater than

$$time(\mathsf{A}) \ + \ time(\text{oracle simulation}) \ \approx \ t' + t'(n \cdot t_g) \ = \ O(t' \cdot n \cdot t_g)$$

where by setting $t' = t/O(n \cdot t_g)$ for some constant $O(1)$ the above will be less than $t - O(t' \cdot t_g)$, and thus contradicts the $(t - O(t' \cdot t_g), \varepsilon \cdot t')$-security of $G^{t'}$. □

HOMEWORK 4. Exercises 3.8, 3.10, 3.11, 6.14, 6.16 from the KL book.