CS6222 Cryptography

Topic: Introduction
Lecturer: Wei-Kai Lin (TA: Arup Sarker)

Date: Sep 12, 2024
Scriber: Shiyu Li, Han Yang

---

1 Pseudorandom Functions

We begin with the definition of random functions.

1.1 Random functions

Definition 1 (Random Functions)**.** A random function $f : \{0,1\}^n \to \{0,1\}^n$ is a random variable sampled uniformly from the set $\mathsf{RF}_n := \{f : \{0,1\}^n \to \{0,1\}^n\}$.

The terminology could be confusing. Like we just said, a mathematical function should be something deterministic. It is an object that always maps the same input to the same output. How can it be random in any sense? When we talk about random functions, it is a set of functions. Here, we just define the set RF to be the set of all functions that maps from n bits to n bits. Every function is an object, and we can have a set of objects and call it another set. We can sample an object from this set.

That is the definition of random function. It is mathematically a function, but we sampled the uniformly random from a set of functions. It is random in the sense that we don't know exactly what the sampling process will give us. The point is, if we have a random function sampled, it still maps the same input to the same output.

View random function in two views. Combinatorial view. RF is parameterize by some finite integer n, we can indeed count how many objects we have in this set. For every function f that the maps from n bits to n bits, we can view this function as a truth table (Table 1). For every f that the maps from n bits to n, we can write down with two tables. The first row of the input column will be a string with n zeros, and that of the output column is another n-bit string. We do not know exactly what the value is, but it is an n-bit string.

Input	Output
0000...00	f(0000...00)
0000...01	f(0000...01)
...	...
1111...11	f(1111...11)

Table 1: View f as a truth table.

n it is a finite integer, so we have 2^n distinct inputs. For every row here, there is a n-bit output, and each output has 2^n options. We can then count the number of functions in the set RF, which is $2^{n \cdot 2^n}$.

Computational view. A random function f is a data structure that on any input x, perform the following:

- initialize a map m to empty before any query
- if $x \notin m$, then sample $y \leftarrow \{0,1\}^n$ and set $m[x] \leftarrow y$
- output $m[x]$

We can use some shorter description to emulate a random function. First think a little bit back in the example of encryption. Suppose Alice and Bob share a random function. That means, Alice and Bob share a huge truth table. With this random function shared the in advance as their secret key, Alice and bob can indeed encrypt the many messages using this random function. For example, Alice can encrypt some message m by calling this random function f on some sequential number r_0 and sending $(r_0, f(r_0) \oplus m)$. Then Bob can call f on r_0 and further recover the message. They can use this random function to encrypt many messages. But it is super efficient to share this random function in advance. We want to replace this random function with something that makes encryption more efficient. That is the purpose of considering random functions as well as their alternatives.

1.2 Oracle indistinguishability

When I say replacing this random function with something else that makes encryption more efficient, I am talking about the computational indistinguishability. But now it is oracle indistinguishability. Oracle is a fancy name for functions, and random functions are also called random oracles. When we say oracle, we want to stress the property of functions that you can only query on some inputs and get the outputs, but you do not know how the function is computed.

Oracle in real world is typically a human that knows the future. The person can answer your questions, but the person will not show you anything else.

Definition 2 (Oracle Indistinguishability)**.** Let $\{\mathcal{O}_n^0\}_{n\in\mathbb{N}}$ and $\{\mathcal{O}_n^1\}_{n\in\mathbb{N}}$ be ensembles where $\mathcal{O}_n^0, \mathcal{O}_n^1$ are probability distributions over functions. We say that $\{\mathcal{O}_n^0\}_n$ and $\{\mathcal{O}_n^1\}_n$ are *computationally indistinguishable* if for all NuPPT machines D that is given oracle accesses to a function, there exists a negligible function $\epsilon(\cdot)$ such that for all $n \in \mathbb{N}$,

$$\Pr[F \leftarrow \mathcal{O}^0 : D^{F(\cdot)}(1^n) = 1] - \Pr[F \leftarrow \mathcal{O}^1 : D^{F(\cdot)}(1^n) = 1] \leq \epsilon(n).$$

Purpose of defining oracle queries. Each distribution is a distribution over functions, and the description of each function can be really long. When the adversary takes a very long input, it is hard to find the running time of the adversary. That is one reason we define oracle queries (e.g., $D^{F(\cdot)}$). Another reason is that when we put a function on the superscript of an algorithm, that means the algorithm can query this function using inputs and get the outputs, but it doesn't know anything beyond these.

Purpose of taking 1^n as an input. It is necessary to say D is a polynomial-time algorithm. Otherwise, it has no input, then how do we say polynomial time in precise? 1^n tells D the problem size is n.

Discuss. For any NuPPT M, is $M(f(\cdot) \leftarrow \mathcal{O}_n^0)$ indistinguishable from $M(f(\cdot) \leftarrow \mathcal{O}_n^1)$?

It is easy to verify that oracle indistinguishability satisfies "closure under efficient operations" using the Hybrid Lemma and the Prediction Lemma.

1.3 Pseudo-random Functions(PRFs)

The purpose of pseudo-random functions is an efficient computable set of functions that we can emulate the functions.

Definition 3 (Pseudo-random Functions)**.** A family of functions $\{f_s : \{0,1\}^n \to \{0,1\}^n, n = |s|\}, s \in \{0,1\}^*$ is pseudo-random if:

- (Easy to compute): $f_s(x)$ can be computed by a PPT algo that is given input s,x.
- (Pseudo-random): $\{s \leftarrow \{0,1\}^n : f_s\}_n \approx \{F \leftarrow RF_n : F\}_n$

For the second condition, on the right hand side, it is an ensemble of distributions on functions. For every n, I have one distribution RF and I am sampling from this distribution or set RF to the function F. On the left hand side, I am also talking about functions. I first pick a uniform seed of n-bit string, and then using this n-bit string to choose a function of f_s. For every n, we are comparing the distribution of functions to distribution of functions. We require that there is no efficient algorithm to distinguish these two distribution of functions.

Discuss.

- Does PRF exist?

  The existence of PRF implies the existence of PRG, and the existence of PRG implies $P \neq NP$, so we don't know.

- Suppose g is a PRG, is g a PRF?

  No, the syntax does not match.

- Can we construct a PRF $f_s(x)$ as $f_s(x) := g(s)[x+1, ..., x+n]$, where g is a PRG?

  No. First, it may take a long time to compute. For example, when $|x| = n$, the maximum value of x is $2^n - 1$, we have to compute $g(s)[2^n, ..., 2^n + n]$. Moreover, we do not know if this is strictly pseudo-random any more.

- Why a PRF must be a keyed function?

  Because we want to pick a function randomly, we cannot use a fixed function, otherwise the adversary will know the response from the function.

- The AES encryption is a deterministic algorithm $\mathsf{Enc}(k, m)$ that takes a 256-bit key k and an arbitrary-length message m. (We omit the initial vector and the block modes and just use the default.) Is Enc a PRF?

  It is not a well defined PRF. Reasons: 1) Key length is fixed in AES. 2) AES's key length is shorter.

1.4 Chosen-Plaintext-Attack secure (CPA)

Previously, when we discussed about the security encryption schemes, we mostly only consider the encryption of one single message. Now we are moving on to define a secure encryption for many messages.

Assuming there is an adversary who can send a plaintext to me and get the ciphertext. The adversary can repeat the process many times. But there is one message the adversary does not

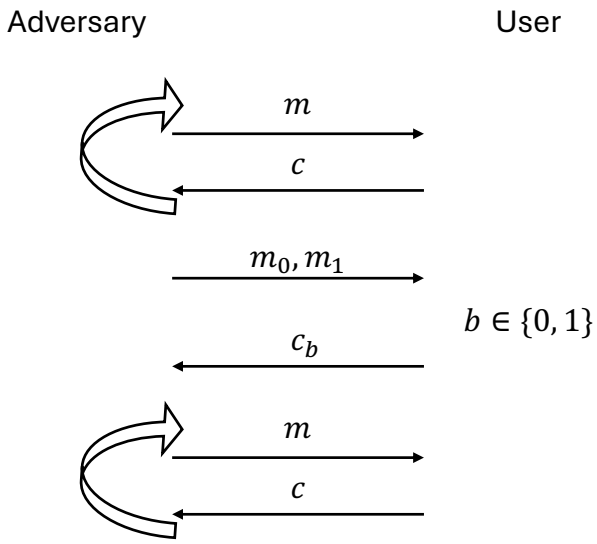

Figure 1: CPA Experiment.

know. In particular, the adversary chooses two messages m_0 and m_1. I am encrypting one of the two messages, so I choose a random bit b uniformly. My only secret is the b, and all the adversary wants to know is the secret b. I encrypt m_b using my key which is sampled in advance and send this ciphertext to the adversary. The adversary needs to guess the bit b, and the winning condition to the adversary is guessing b correctly.

Definition 4 (Chose-Plaintext-Attack Encryption (CPA))**.** Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. For any NuPPT adversary A, for any $n \in \mathbb{N}, b \in \{0,1\}$, define the experiment $\mathsf{Expr}_b^{\Pi,A}(1^n)$ to be:

Experiment $\mathsf{Expr}_b^{\Pi,A}(1^n)$:

1. $k \leftarrow \mathsf{Gen}(1^n)$
2. $(m_0, m_1, \mathsf{state}) \leftarrow A^{\mathsf{Enc}_k(\cdot)}(1^n)$
3. $c \leftarrow \mathsf{Enc}_k(m_b)$
4. Output $A^{\mathsf{Enc}_k(\cdot)}(c, \mathsf{state})$

Then we say Π is CPA secure if for all NuPPT A,

$$\left\{\mathsf{Expr}_0^{\Pi,A}(1^n)\right\}_n \approx \left\{\mathsf{Expr}_1^{\Pi,A}(1^n)\right\}_n.$$