Lecture 16: Pseudorandom Functions
Random Functions

- Let $\mathcal{F}_{m,n}$ be the set of all function from the domain $\{0, 1\}^m$ to the range $\{0, 1\}^n$

- Each function $f \in \mathcal{F}_{m,n}$ can be uniquely represented by a list of length $\{0, 1\}^m$ where the $i$-th entry in the list is the entry $f(i)$, for $i \in \{0, 1\}^m$

- So, each entry in the list has $2^n$ options. And, there are a total of $2^m$ such entries. So, the total number of distinct functions from the set $\{0, 1\}^m \to \{0, 1\}^n$ is

$$\underbrace{(2^n) \times \cdots \times (2^n)}_{2^m\text{-times}} = 2^{n2^m}$$

- So, we can conclude that each function $f \in \mathcal{F}_{m,n}$ can be described using $n2^m$ bits
Crucial Property of Random Functions

Intuition.

- Suppose we pick a random $f \leftarrow \mathcal{F}_{m,n}$.
- Then the evaluation of $f$ at any input $x_1$ is uniformly random over $\{0,1\}^n$.
- Further, the evaluation of $f$ at any other input $x_2$ given $f(x_1)$ is again uniformly random over $\{0,1\}^n$.
- In particular, the evaluation of $f$ at an input $x_t$ given $f(x_1), \ldots, f(x_{t-1})$ is uniformly random.
- Intuitively, the evaluation of a random $f$ is completely unpredictable at any new input.

Formally. For any distinct inputs $x_1, \ldots, x_t \in \{0,1\}^m$ and any outputs $y_1, \ldots, y_t \in \{0,1\}^n$, the following holds

$$
\mathbb{P}_{f \leftarrow \mathcal{F}_{m,n}} \left[ f(x_t) = y_t \mid f(x_1) = y_1, \ldots, f(x_{t-1}) = y_{t-1} \right] = \frac{1}{2^n}
$$
Secret-key Encryption using Random Functions

Consider the following private-key encryption scheme

1. **Gen()**: Return \( sk = f \leftarrow \mathcal{F}_{m,n} \)
2. **Enc\(_f\)(m)**: Pick a random \( r \leftarrow \{0, 1\}^m \). Return \((m \oplus f(r), r)\), where \( m \in \{0, 1\}^n \).
3. **Dec\(_f\)(\(\tilde{c}, \tilde{r}\))**: Return \( \tilde{c} \oplus f(\tilde{r}) \).

**Features.** Suppose the messages \( m_1, \ldots, m_u \) are encrypted as the cipher-texts \((c_1, r_1), \ldots, (c_u, r_u)\).

- As long as the \( r_1, \ldots, r_u \) are all distinct, each one-time pad \( f(r_1), \ldots, f(r_u) \) are uniform and independent of others. So, this encryption scheme is perfectly secure!

- The probability that any two of the randomness in \( r_1, \ldots, r_u \) are not distinct is very small (We shall prove this later as “Birthday Paradox”)

- This scheme is a “state-less” encryption scheme. Alice and Bob do not need to remember any private state (except the secret-key \( sk \))!
Bottleneck of using Random Functions

- The secret-key sk needs $n2^m$ bits to represent it, which is exponentially large.
- We shall replace “random functions” using “pseudorandom functions” to construct an encryption scheme that has short keys and remains secure against computationally bounded adversaries!
Pseudo-random Functions (PRF)

- Let $G_{m,n,k} = \{g_1, g_2, \ldots, g_{2^k}\}$ be a set of functions such that each $g_i: \{0, 1\}^m \rightarrow \{0, 1\}^n$.

- This set of functions $G_{m,n,k}$ is called a pseudo-random function if the following holds. Suppose we pick $g \leftarrow G_{m,n,k}$. Let $x_1, \ldots, x_t \in \{0, 1\}^m$ be distinct inputs. Given $(x_1, g(x_1)), \ldots, (x_{t-1}, g(x_{t-1}))$ for any computationally bounded party the value $g(x_t)$ appears to be uniformly random over $\{0, 1\}^n$. 

PRF
Before we construct a PRF, let us consider the following secret-key encryption scheme.

1. **Gen()**: Return \( sk = \text{id} \leftarrow \{1, \ldots, 2^k\} \)

2. **Enc\(_{\text{id}}\)(m)**: Pick a random \( r \leftarrow \{0, 1\}^m \). Return \((m \oplus g_{\text{id}}(r), r)\), where \( m \in \{0, 1\}^n \).

3. **Dec\(_{\text{id}}\)(c, r)**: Return \( \tilde{c} \oplus g_{\text{id}}(\tilde{r}) \).

**Features.** Suppose the messages \( m_1, \ldots, m_u \) are encrypted as the cipher-texts \((c_1, r_1), \ldots, (c_u, r_u)\).  

- As long as the \( r_1, \ldots, r_u \) are all distinct, each one-time pad \( g_{\text{id}}(r_1), \ldots, g_{\text{id}}(r_u) \) appear uniform and independent of others to computationally bounded adversaries. So, this encryption scheme is secure against computationally bounded adversaries!

- The probability that any two of the randomness in \( r_1, \ldots, r_u \) are not distinct is very small (We shall prove this later as “Birthday Paradox”)

- This scheme is a “state-less” encryption scheme. Alice and Bob do not need to remember any private state (except the secret-key \( sk \)).
We shall consider the construction of Goldreich-Goldwasser-Micali (GGM) construction.

Let $G : \{0, 1\}^k \rightarrow \{0, 1\}^{2^k}$ be a PRG. We define $G(x) = (G_0(x), G_1(x))$, where $G_0, G_1 : \{0, 1\}^k \rightarrow \{0, 1\}^k$

Let $G' : \{0, 1\}^k \rightarrow \{0, 1\}^n$ be a PRG

We define $g_{id}(x_1 x_2 \ldots x_m)$ as follows

\[ G'(G_{x_m}(\cdots G_{x_2}(G_{x_1}(id))\cdots)) \]
Consider the execution for $x = x_1x_2x_3 = 010$. Output $z$ is computed as follows.

Go Left because $x_1 = 0$

Go Right because $x_2 = 1$

Go Left because $x_3 = 0$
We give the pseudocode of algorithms to construct PRG and PRF using a OWP \( f : \{0, 1\}^{k/2} \rightarrow \{0, 1\}^{k/2} \)

- Suppose \( f : \{0, 1\}^{k/2} \rightarrow \{0, 1\}^{k/2} \) is a OWP
- We provide the pseudocode of a PRG \( G : \{0, 1\}^k \rightarrow \{0, 1\}^t \), for any integer \( t \), using the one-bit extension PRG construction of Goldreich-Levin hardcore predicate construction. Given input \( s \in \{0, 1\}^k \), it outputs \( G(s) \).

\[
\begin{align*}
G(k, t, s) : & \\
1 & \text{Interpret } s = (r, x), \text{ where } r, x \in \{0, 1\}^{k/2} \\
2 & \text{Initialize } \text{bits} = [ ] \text{ (i.e., an empty list)} \\
3 & \text{Initialize } z = x \\
4 & \text{For } i = 1 \text{ to } t: \\
 & \quad 1 \text{ bits.append}(\langle r, z \rangle), \text{ here } \langle \cdot, \cdot \rangle \text{ is the inner-product} \\
 & \quad 2 \ z = f(z) \\
5 & \text{Return } \text{bits}
\end{align*}
\]
We provide the pseudocode of the PRF
\[ g_{\text{id}} : \{0, 1\}^m \rightarrow \{0, 1\}^n, \text{ where } \text{id} \in \{0, 1\}^k, \text{ using the GGM construction. Given input } x \in \{0, 1\}^m, \text{ it outputs } g_{\text{id}}(x). \]

\[ g(m, n, k, \text{id}, x) : \]

1. Interpret \( x = x_1x_2 \ldots x_m \), where \( x_1, \ldots, x_m \in \{0, 1\} \)
2. Initialize \( \text{inp} = \text{id} \)
3. For \( i = 1 \) to \( m \):
   1. Let \( y = G(k, 2k, \text{inp}) \)
   2. If \( x_i = 0 \), then \( \text{inp} \) is the first \( k \) bits of \( y \). Otherwise (if \( x_i = 1 \)), \( \text{inp} \) is the last \( k \) bits of \( y \).
4. Return \( G(k, n, \text{inp}) \)