# MIT 6.875 - Foundations of Cryptography

Lectures by Vinod Vaikuntanathan, Notes by Gary Hu

Fall 2021

These are my notes for MIT 6.875, taught by Vinod Vaikuntanathan.

## Contents

# 1 Introduction

## 1.1 Themes

The following themes will be central to the course and will recur throughout the lectures:

1. Model the worst-case adversary: Understand what the adversary knows, what they can do, and what their objectives are.
2. Leverage computational hardness to "control" the adversary.
3. Provide security proofs via reductions.

## 1.2 The Setup

Cryptography aims to enable secure communication between Alice and Bob, such that Alice can send a message $M$ to Bob without Eve being able to intercept or decipher it.

Before Alice and Bob can exchange messages, they must first agree on a secret key $k$.

For simplicity, we assume Alice and Bob are using secret-key encryption. This involves an encryption scheme consisting of three (potentially probabilistic) polynomial-time algorithms:

- **Key Generation Algorithm (Gen)**: $k \leftarrow \text{Gen}(1^n)$.
- **Encryption Algorithm (Enc)**: $c \leftarrow \text{Enc}(k, m)$.
- **Decryption Algorithm (Dec)**: $m \leftarrow \text{Dec}(k, c)$.

Note that the key generation algorithm is inherently probabilistic.

We assume the worst-case adversary, Eve, who has the following abilities:

- Eve is an arbitrary computationally unbounded algorithm.
- **Kerckhoff's Principle**: Eve knows Alice and Bob's algorithms Gen, Enc, and Dec, but does not have knowledge of the key $k$ or any internal randomness.
- Eve can observe the ciphertexts transmitted through the channel, but, for now, is unable to modify them.

## 1.3 Shannon's Perfect Secrecy

The fundamental idea behind perfect secrecy is that the a-posteriori probability of a message should equal the a-priori probability. More formally:

**Definition 1.1** (Shannon's Perfect Secrecy)**.** *An encryption scheme is perfectly secure if*

$$\forall \mathcal{M}, \forall m \in \mathit{Supp}(\mathcal{M}), \forall c \in \mathit{Supp}(\mathcal{C}),$$
$$\mathit{Pr}[\mathcal{M} = m \mid \mathit{Enc}(\mathcal{K}, \mathcal{M}) = c] = \mathit{Pr}[\mathcal{M} = m]$$

Now, let's examine an equivalent definition based on perfect indistinguishability.

**Definition 1.2** (Perfect Indistinguishability)**.** *An encryption scheme exhibits perfect indistinguishability if*

$$\forall \mathcal{M}, \forall m, m' \in \mathit{Supp}(\mathcal{M}), \forall c \in \mathit{Supp}(\mathcal{C}),$$

$$\mathit{Pr}[\mathit{Enc}(\mathcal{K}, m) = c] = \mathit{Pr}[\mathit{Enc}(\mathcal{K}, m') = c]$$

We will now prove that these two definitions are equivalent.

**Theorem 1.3.** *An encryption scheme satisfies perfect secrecy if and only if it satisfies perfect indistinguishability.*

*Proof.* We begin with the "if" direction. The key observation is that for all messages $m$,

$$\Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c] = \Pr[\mathrm{Enc}(\mathcal{K}, m) = c]$$

This follows from the fact that:

$$\begin{aligned}
\Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c] &= \sum \Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c \mid \mathcal{M} = m] \Pr[\mathcal{M} = m] \\
&= \sum \Pr[\mathrm{Enc}(\mathcal{K}, m) = c] \Pr[\mathcal{M} = m] \\
&= \alpha \sum \Pr[\mathcal{M} = m] \\
&= \alpha
\end{aligned}$$

where $\alpha = \Pr[\mathrm{Enc}(\mathcal{K}, m) = c] = \Pr[\mathrm{Enc}(\mathcal{K}, m') = c]$.

From this, we can proceed as follows:

$$\begin{aligned}
\Pr[\mathcal{M} = m \mid \mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c] &= \frac{\Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c \mid \mathcal{M} = m] \Pr[\mathcal{M} = m]}{\Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c]} \\
&= \frac{\Pr[\mathrm{Enc}(\mathcal{K}, m) = c] \Pr[\mathcal{M} = m]}{\Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c]} \\
&= \frac{\alpha \Pr[\mathcal{M} = m]}{\alpha} \\
&= \Pr[\mathcal{M} = m]
\end{aligned}$$

This concludes the "if" direction.

Now, for the "only if" direction, we assume the perfect secrecy condition holds, and we show that it implies perfect indistinguishability.

$$\begin{aligned}
\Pr[\mathrm{Enc}(\mathcal{K}, m) = c] &= \Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c \mid \mathcal{M} = m] \\
&= \frac{\Pr[\mathcal{M} = m \mid \mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c] \Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c]}{\Pr[\mathcal{M} = m]} \\
&= \Pr[\mathrm{Enc}(\mathcal{K}, \mathcal{M}) = c] \\
&= \Pr[\mathrm{Enc}(\mathcal{K}, m') = c]
\end{aligned}$$

Thus, we conclude that perfect secrecy implies perfect indistinguishability. □

## 1.4 One-time Pad Construction

Let us now describe a simple construction of a perfectly secure encryption scheme: the one-time pad.

**Definition 1.4** (One-time Pad)**.** *The one-time pad encryption scheme is defined as follows:*

- ***Gen**: Choose an $n$-bit string $k$ at random.*
- ***Enc**$(k, m)$: Given a message $m$ (an $n$-bit string), the ciphertext $c$ is computed as $c = m \oplus k$, where $\oplus$ denotes bitwise addition modulo 2.*
- ***Dec**$(k, c)$: Given the ciphertext $c$, the message is recovered by computing $m = c \oplus k$.*

We now show that this construction is indeed perfectly secure.

**Theorem 1.5.** *The one-time pad is perfectly secure.*

*Proof.* To demonstrate that the one-time pad achieves perfect secrecy, consider any message $m$ and ciphertext $c \in \{0, 1\}^n$. The probability that the encryption of $m$ results in $c$ is:

$$\Pr[\mathrm{Enc}(k, m) = c] = \Pr[m \oplus k = c] = \Pr[k = c \oplus m] = \frac{1}{2^n}$$

Thus, the probability of obtaining any ciphertext $c$ is the same regardless of the message $m$, implying that:

$$\Pr[\mathrm{Enc}(\mathcal{K}, m) = c] = \Pr[\mathrm{Enc}(\mathcal{K}, m') = c]$$

which confirms that the one-time pad is perfectly secure. □

However, the one-time pad has a critical limitation: it cannot be reused. If the same key is used for encrypting multiple messages, Eve can XOR the ciphertexts and gain information about the messages.

## 1.5 Limitations of Perfect Secrecy

While perfect secrecy is often considered the ideal in cryptography, it has certain limitations. Specifically, the following theorem highlights one key constraint:

**Theorem 1.6.** *For any perfectly secure encryption scheme, the size of the key space must be at least as large as the size of the message space:*

$$|\mathcal{K}| \geq |\mathcal{M}|$$

*Proof.* Assume for contradiction that $|\mathcal{K}| < |\mathcal{M}|$. Let $c \in \mathcal{C}$ be any ciphertext. Consider the set of all possible messages that could be encrypted to $c$. The set of keys mapping these messages to $c$ must be distinct, leading to a contradiction. There must exist a message $\tilde{m} \in \mathcal{M}$ that is not encrypted to $c$, which violates the definition of perfect indistinguishability because:

$$\Pr[\text{Enc}(\mathcal{K}, m) = c] > 0 \quad \text{but} \quad \Pr[\text{Enc}(\mathcal{K}, \tilde{m}) = c] = 0$$

Thus, we conclude that $|\mathcal{K}| \geq |\mathcal{M}|$ must hold for perfect secrecy. □

In subsequent lectures, we will relax some of the definitions and explore more practical cryptographic schemes.

# 2 Circumventing Shannon's Lower Bound Using PRGs

## 2.1 Computationally Bounded Adversaries

In the previous post, we discussed the limitations of perfect security. Today, we explore how we can circumvent this limitation.

The key idea is to assume computationally bounded adversaries. We assume the Church-Turing Thesis holds, meaning feasible computation is defined as probabilistic polynomial time (p.p.t.). In this context, Alice and Bob are fixed p.p.t. algorithms, and Eve is any p.p.t. adversary.

Let's attempt to define *computational indistinguishability.* A natural approach would be to define computational indistinguishability as follows:

For all p.p.t. Eve and for all messages $m_0, m_1$, the following condition should hold:

$$\Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\}; c = \text{Enc}(k, m_b) : \text{Eve}(c) = b] = \frac{1}{2}$$

However, there is a critical flaw in this approach: it is impossible to have an encryption scheme that encrypts an $n+1$-bit message with an $n$-bit key.

Let $m_0$ be a message that can be encrypted to a ciphertext $c$, and let $m_1$ be a message that cannot be encrypted to $c$. Suppose Eve picks a random key $k$ and:

- Outputs 0 if $\text{Dec}(k, c) = m_0$
- Outputs 1 if $\text{Dec}(k, c) = m_1$
- Outputs a random bit if neither of the above conditions hold

Then the probability of the first case occurring is at least $\frac{1}{2^n}$, but the second case has a probability of 0, leading to a contradiction.

Thus, we must relax the definition of computational indistinguishability. Before proceeding, we first need to introduce the concept of negligible functions.

## 2.2 Negligible Functions

**Definition 2.1.** *A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if, for every polynomial function $p$, there exists an $n_0$ such that for all $n \geq n_0$, $\mu(n) < \frac{1}{p(n)}$.*

**Example 2.2.**

1. *Let $\mu(n) = \frac{1}{n^{\log n}}$. Is $\mu$ negligible?*
2. *Let $\mu(n) = \frac{1}{n^{100}}$ if $n$ is prime and $\mu(n) = \frac{1}{2^n}$ otherwise. Is $\mu$ negligible?*
3. *Let $\mu(n)$ be a negligible function and $q(n)$ be a polynomial function. Is $\mu(n) \cdot q(n)$ negligible?*

*Here are the answers:*

1. *No.*
2. *No, because there are infinitely many primes.*
3. *Yes.*

Now that we understand negligible functions, we can proceed to define computational indistinguishability.

**Definition 2.3.** ***Computational Indistinguishability:*** *For all p.p.t. Eve, there exists a negligible function $\mu$ such that for all messages $m_0, m_1$, we have:*

$$Pr[k \leftarrow \mathcal{K}; b \leftarrow \{0,1\}; c = Enc(k, m_b) : Eve(c) = b] \leq \frac{1}{2} + \mu(n)$$

## 2.3 Pseudorandom Generators

Informally, pseudorandom generators (PRGs) are deterministic algorithms that can stretch a "truly random" seed into a much longer sequence of "seemingly random" bits.

There are several equivalent ways to think about pseudorandom generators:

- **Indistinguishability:** No polynomial-time algorithm can distinguish between the output of a PRG on a random seed and a truly random string.
- **Next-bit unpredictability:** No polynomial-time algorithm can predict the $(i+1)$-th bit of the output of a PRG given the first $i$ bits better than random chance.
- **Incompressibility:** No polynomial-time algorithm can compress the output of a PRG into a shorter string.

We can now formalize the definition of indistinguishability for a PRG:

**Definition 2.4.** ***Indistinguishability:*** *A deterministic polynomial-time computable function $G : \{0,1\}^n \rightarrow \{0,1\}^m$ is a PRG if:*

1. *It is expanding: $m > n$.*
2. *For every p.p.t. algorithm $D$, there exists a negligible function $\mu$ such that*

   $$|Pr[D(G(U_n)) = 1] - Pr[D(U_m) = 1]| = \mu(n),$$

   *where $U_n$ denotes the uniform distribution over $n$-bit strings.*

The purpose of PRGs is to allow us to encrypt $n+1$ bits using an $n$-bit key. We achieve this using the following construction:

- **Gen($1^n$):** Generate a random $n$-bit key $k$.
- **Enc($k, m$), where $m$ is an $m(n)$-bit message:** Expand $k$ into an $n+1$-bit pseudorandom string $k' = G(k)$. Perform a one-time pad with $k'$, and the ciphertext is $k' \oplus m$.

- **Dec($k, c$):** Output $G(k) \oplus c$.

**Theorem 2.5.** *Suppose $G$ is a PRG. Then the above encryption scheme is computationally secure.*

*Proof.* Suppose for contradiction that there exists a p.p.t. Eve, a polynomial function $p$, and messages $m_0, m_1$ such that

$$\rho = \Pr[k \leftarrow \{0,1\}^n; b \leftarrow \{0,1\}; c = G(k) \oplus m_b : \text{Eve}(c) = b] \geq \frac{1}{2} + \frac{1}{p(n)}.$$

Let

$$\rho' = \Pr[k \leftarrow \{0,1\}^{n+1}; b \leftarrow \{0,1\}; c = k' \oplus m_b : \text{Eve}(c) = b] = \frac{1}{2}.$$

Now, we construct a distinguisher Eve' for $G$, which leads to a contradiction.

Given an input string $y$, run Eve$(y \oplus m_b)$ for a random $b$, and let Eve$'$'s output be $b'$. Output "PRG" if $b = b'$, and "random" otherwise. Then:

$$\Pr[\text{Eve' outputs "PRG"} | y \text{ is pseudorandom}] = \rho \geq \frac{1}{2} + \frac{1}{p(n)},$$

and

$$\Pr[\text{Eve' outputs "PRG"} | y \text{ is random}] = \rho' = \frac{1}{2}.$$

Therefore,

$$\Pr[\text{Eve' outputs "PRG"} | y \text{ is pseudorandom}] - \Pr[\text{Eve' outputs "PRG"} | y \text{ is random}] \geq \frac{1}{p(n)},$$

which is a contradiction. Hence, the encryption scheme is secure. □

In the next post, we will explore PRGs in more detail.

# 3 Stateless Secret-key Encryption Leads to PRFs

## 3.1 Next-bit Unpredictability (NBU)

The question for today is: How do you encrypt a polynomial number of messages with a fixed key?

In the previous lecture, we introduced next-bit unpredictability informally. Now, we will define it rigorously.

**Definition 3.1** (Next-bit Unpredictability)**.** *A deterministic polynomial-time computable function $G : \{0,1\}^n \to \{0,1\}^m$ is **next-bit unpredictable** if for every p.p.t. algorithm $P$ and every $i \in \{1, \dots, m\}$, there exists a negligible function $\mu(n)$ such that*

$$Pr[y \leftarrow G(U_n) : P(y_1 y_2 \dots y_{i-1}) = y_i] = \frac{1}{2} + \mu(n)$$

This definition is useful due to the following theorem:

**Theorem 3.2.** *A PRG G passes all polynomial-time statistical tests if and only if it passes all polynomial-time next-bit tests.*

## 3.2 Proof of NBU $\Leftrightarrow$ Indistinguishability

The proof is lengthy, so I will break it into sections.

First, let's prove that indistinguishability implies next-bit unpredictability.

Suppose, for the sake of contradiction, that there exists a p.p.t. predictor $P$, a polynomial function $p(n)$, and an index $i \in \{1, \dots, m\}$ such that

$$\Pr[y \leftarrow G(U_n) : P(y_1 y_2 \dots y_{i-1}) = y_i] \geq \frac{1}{2} + \mu(n)$$

Now, let's construct the distinguisher $D$. Consider $D$, which receives an $m$-bit string $y$ and proceeds as follows:

1. Run $P$ on the $(i-1)$-bit prefix $y_1 y_2 \dots y_{i-1}$.
2. If $P$ returns the $i$-th bit $y_i$, output 1 ("PRG"); otherwise, output 0 ("Random").

Let's show that $D$ works. We have the following:

$$\begin{aligned}\Pr[y \leftarrow G(U_n) : D(y) = 1] &= \Pr[y \leftarrow G(U_n) : P(y_1 y_2 \dots y_{i-1}) = y_i] \\ &\geq \frac{1}{2} + \frac{1}{p(n)}\end{aligned}$$

and

$$\begin{aligned}\Pr[y \leftarrow U_m : D(y) = 1] &= \Pr[y \leftarrow U_m : P(y_1 y_2 \ldots y_{i-1}) = y_i] \\ &= \frac{1}{2}\end{aligned}$$

which implies

$$|\Pr[y \leftarrow G(U_n) : D(y) = 1] - \Pr[y \leftarrow U_m : D(y) = 1]| \geq \frac{1}{p(n)}$$

This is a contradiction, so we conclude that indistinguishability implies next-bit unpredictability.

Next, we prove the reverse direction: next-bit unpredictability implies indistinguishability.

Suppose, for contradiction, that there exists a distinguisher $D$ and a polynomial function $p(n)$ such that

$$|\Pr[y \leftarrow G(U_n) : D(y) = 1] - \Pr[y \leftarrow U_m : D(y) = 1]| \geq \frac{1}{p'(n)}$$

Let $\epsilon := \frac{1}{p'(n)}$ for simplicity. Before proceeding, we need the following lemma:

**Lemma 3.3.** *Let $p_0, p_1, p_2, \ldots, p_m$ be real numbers such that*

$$p_m - p_0 \geq \epsilon$$

*Then, there exists an index $i$ such that*

$$p_i - p_{i-1} \geq \frac{\epsilon}{m}$$

*Proof.*

$$p_m - p_0 = (p_m - p_{m-1}) + (p_{m-1} - p_{m-2}) + \cdots + (p_1 - p_0) \geq \epsilon$$

Averaging the terms completes the proof. □

Using the lemma, we have that for each $i$ such that $D$ distinguishes between $H_{i-1}$ and $H_i$ with advantage greater than or equal to $\frac{\epsilon}{m}$,

$$\Pr[D(H_i) = 1] - \Pr[D(H_{i-1}) = 1] \geq \frac{\epsilon}{m}$$

Define $p_i := \Pr[D(H_i) = 1]$, $p_0 := \Pr[D(U_m) = 1]$, and $p_m := \Pr[D(G(U_n))]$. It's clear that $p_i - p_{i-1} \geq \frac{\epsilon}{m}$. The key intuition is that the only difference between the hybrid distributions $H_{i-1}$ and $H_i$ is the $i$-th bit. So, $D$ can tell whether the given bit is the correct $i$-th bit or not.

Now, let's define a new variable $\overline{H_i} := 1 - y_1$ and $\overline{p_i} := \Pr[D(\overline{H_i}) = 1]$. To continue, we need another lemma:

**Lemma 3.4.**

$$p_{i-1} = \frac{p_i + \overline{p_i}}{2}$$

*Proof.* This follows from the fact that $p_i$ is the probability that $D$ outputs 1 when given 0 in the $i$-th bit, and $\overline{p_i}$ is the probability that $D$ outputs 1 when given 1 in the $i$-th bit. □

This leads to the following corollary:

**Corollary 3.5.**

$$p_i - \overline{p_i} \geq 2\frac{\epsilon}{m}$$

*Proof.* This is immediate. □

This shows that $D$ outputs 1 more frequently when given the correct bit than when given the wrong bit. Finally, we can construct our predictor $P$:

1. Pick a random bit $b$.
2. Feed $D$ with the input $y_1 y_2 \dots y_{i-1}|b|u_{i+1} \dots u_m$, where $u$'s are random.
3. If $D$ outputs 1, output $b$ as the prediction for $y_i$; otherwise, output $\overline{b}$.

To finish, we need to show that our predictor works:

$$\begin{aligned}
&\Pr[x \leftarrow \{0,1\}^n; y = G(x) : P(y_1 y_2 \dots y_{i-1}) = y_i] \\
=&\Pr[D(y_1 y_2 \dots y_{i-1} b \dots) = 1 \mid b = y_i] \cdot \Pr[b = y_i] + \Pr[D(y_1 y_2 \dots y_{i-1} b \dots) = 0 \mid b \neq y_i] \cdot \Pr[b \neq y_i] \\
=&\frac{1}{2} \left( \Pr[D(y_1 y_2 \dots y_{i-1} b \dots) = 1 \mid b = y_i] + \Pr[D(y_1 y_2 \dots y_{i-1} b \dots) = 0 \mid b \neq y_i] \right) \\
=&\frac{1}{2} \left( \Pr[D(y_1 y_2 \dots y_{i-1} y_i \dots) = 1] + \Pr[D(y_1 y_2 \dots y_{i-1} \overline{y_i} \dots) = 0] \right) \\
=&\frac{1}{2} \left( 1 + 2\frac{\epsilon}{m} \right) \\
\geq&\frac{1}{2} + \frac{1}{p(n)}
\end{aligned}$$

Thus, we have completed the proof of the theorem.

## 3.3 Length Extension

Here's how you can use a PRG to turn an $n$-bit seed into a longer message of arbitrary length.

1. Take the $n$-bit seed and input it into a PRG $G$ to get an $n+1$-bit message.
2. Remove the $(n+1)$-th bit and store it separately.
3. Repeat steps 1 and 2 $L$ times to obtain the original message and $L$ pseudorandom bits.
4. Attach the pseudorandom bits to the end of the seed.

The important question now becomes: How can we achieve this without maintaining state?

Here are some ideas:

1. Alice picks a random index from $b_1, b_2, \ldots, b_{n^{100}}$ and sends Bob $(b_n, m \oplus b_n)$. However, this doesn't work due to collisions: Pr[Alice's first two indices collide] $\geq \frac{1}{n^{100}}$, which leads to Alice using the same one-time pad bit twice.
2. Alice picks a random index from $b_1, \ldots, b_{2^n}$. Now we have $\Pr[\exists \text{collision in } t = \text{poly}(n) \text{ indices}] \leq \frac{t^2}{2^n}$, which is negligible, but Alice and Bob are no longer polynomial-time computable.

Instead, we want a function $f_k(x) = b_x$, where $x$ is the index in an implicitly defined string, computable in polynomial time. Then, $f_k(x_1), f_k(x_2), \ldots$ must be computationally indistinguishable from random bits for random $x_1, x_2, \ldots$. This motivates the idea behind pseudorandom functions.

## 3.4 Pseudorandom Functions (PRFs)

**Definition 3.6** (Pseudorandom Function). *A* ***pseudorandom function*** *is a collection of functions* $\mathcal{F}_\ell = \{f_k : \{0,1\}^\ell \to \{0,1\}^m\}_{k \in \{0,1\}^n}$

- *indexed by a key $k$*
- *$n$: key length, $\ell$: input length, $m$: output length*
- *independent parameters, all poly(sec-param) = poly($n$)*
- *the number of functions in $\mathcal{F}_\ell \leq 2^n$*

*with the following algorithms:*

- ***Gen***$(1^n)$*: Generate a random $n$-bit key $k$.*
- ***Eval***$(k, x)$*: A polynomial-time algorithm that outputs $f_k(x)$.*

We want this to be indistinguishable from the collection of all functions $\text{ALL}_\ell = \{f : \{0,1\}^\ell \to \{0,1\}^m\}$, where the number of functions in $\text{ALL}_\ell \leq 2^{m2^\ell}$.

In the pseudorandom world, the distinguisher $D$ is given $f \leftarrow \mathcal{F}_\ell$, $x$, and receives $f(x)$. In the random world, $D$ is given $f \leftarrow \mathrm{ALL}_\ell$, $x$, and receives $f(x)$.

Thus, for all p.p.t. $D$, there is a negligible function $\mu(n)$ such that

$$\left|\Pr[f \leftarrow \mathcal{F}_\ell : D^f(1^n) = 1] - \Pr[f \leftarrow \mathrm{ALL}_\ell : D^f(1^n) = 1]\right| \leq \mu(n)$$

## 3.5 PRFs Imply Stateless Secret-key Encryption

**Theorem 3.7** (PRFs Imply Stateless Secret-key Encryption)**.** *PRFs imply stateless secret-key encryption.*

*Proof.* The following construction works:

1. **Gen**$(1^n)$: Generate a random $n$-bit key $k$ that defines $f_k : \{0,1\}^\ell \rightarrow \{0,1\}^m$.
2. **Enc**$(k, m)$: Pick a random $x$ and let the ciphertext $c$ be the pair $(x, y = f_k(x) \oplus m)$.
3. **Dec**$(k, c = (x, y))$: Output $f_k(x) \oplus y$.

It remains to show correctness, which is trivial. □

Next time: More on PRFs and their connections to PRGs.

# 4 More on Pseudorandom Functions

## 4.1 Definitions

With a Pseudorandom Generator (PRG), accessing the $2^\ell$-th bit requires time $2^\ell$, while for a Pseudorandom Function (PRF), accessing the $i$-th bit only requires time $\ell$. Consequently, we can think of a PRF as a locally accessible PRG.

Let's define secret-key encryption for a single message rigorously:

**Definition 4.1** (Secret-key Encryption for One Message)**.** *For all $m_0, m_1$ and all p.p.t. distinguishers $D$, there exists a negligible function $\mu(n)$ such that*

$$|Pr[k \leftarrow \mathcal{K} : D(Enc(k, m_0)) = 1] - Pr[k \leftarrow \mathcal{K} : D(Enc(k, m_1)) = 1]| \leq \mu(n)$$

But what if we want to define secret-key encryption for multiple messages? Let's introduce the following oracles, both of which only take strings of the same length:

- **Left Oracle:** Left$(\cdot, \cdot)$ has $k \leftarrow \mathcal{K}$ and $c \leftarrow$ Enc$(k, m_L)$, gives the distinguisher $D$ the ciphertext $c$, and receives $m_L, m_R$ back.
- **Right Oracle:** Right$(\cdot, \cdot)$ has $k \leftarrow \mathcal{K}$ and $c \leftarrow$ Enc$(k, m_R)$, gives the distinguisher $D$ the ciphertext $c$, and receives $m_L, m_R$ back.

Now we can proceed with the definition.

**Definition 4.2** (Secret-key Encryption for Many Messages)**.** *For all p.p.t. distinguishers $D$, there exists a negligible function $\mu(n)$ such that*

$$|Pr[k \leftarrow \mathcal{K} : D^{Left(\cdot,\cdot)}(1^n) = 1] - Pr[k \leftarrow \mathcal{K} : D^{Right(\cdot,\cdot)}(1^n) = 1]| \leq \mu(n)$$

## 4.2 Example

**Theorem 4.3.** *Our PRF encryption scheme from the previous lecture satisfies this definition.*

Consider the following sequence of hybrids:

- **Hybrid 0:** $D$ gets access to the left oracle, with ciphertext $c = (x, y = f_k(x) \oplus m_L)$.
- **Hybrid 1:** Replace $f_k$ with a random function, so $c = (x, y = r_x \oplus m_L)$.
- **Hybrid 2:** Replace $f_k$ with a random function, so $c = (x, y = r_x)$.
- **Hybrid 3:** Replace $f_k$ with a random function, so $c = (x, y = r_x \oplus m_R)$.

- **Hybrid 4:** $D$ gets access to the right oracle, with ciphertext $c = (x, y = f_k(x) \oplus m_R)$.

The rest is straightforward.

## 4.3 Goldreich-Goldwasser-Micali (GGM) PRF Construction

Let $G(s) = G_0(s) \parallel G_1(s)$, where $G_0(s)$ is 1 bit and $G_1(s)$ is $n$ bits. However, accessing the $i$-th output bit takes time $i$. We need a new theorem, which we will leave unproved for now:

**Theorem 4.4.** *Let $G$ be a PRG. Then, for every polynomial $\ell = \ell(n)$ and $m = m(n)$, there exists a PRG family $\mathcal{F}_\ell = \{f_s : \{0,1\}^\ell \to \{0,1\}^m\}_{s\in\{0,1\}^n}$.*

Now, let's look at the GGM PRF construction.

**Definition 4.5** (GGM PRF Construction)**.** *Let $G(s) = G_0(s) \parallel G_1(s)$, where both $G_0(s)$ and $G_1(s)$ are $n$-bit strings.*

*The pseudorandom function family $\mathcal{F}_\ell$ is defined by a collection of functions $f_s$, where:*

$$f_s(x_1 x_2 \dots x_\ell) = G_{x_\ell}(G_{x_{\ell-1}}(\dots G_{x_1}(s)))$$

Note that:

- $f_s$ defines $2^\ell$ pseudorandom bits.
- The $x$-th bit can be computed using $\ell$ evaluations of the PRG $G$.

## 4.4 Security Analysis

Let's now analyze the security of the GGM PRF. We begin with a useful lemma:

**Lemma 4.6** (PRG Repetition Lemma)**.** *Let $G$ be a PRG. Then, for every polynomial $L = L(n)$, the following two distributions are computationally indistinguishable:*

$$(G(s_1), G(s_2), \dots, G(s_L)) \approx (u_1, u_2, \dots, u_L)$$

*Proof.* Here is an outline of the proof, with the rest left as an exercise to the reader: If there exists a p.p.t. distinguisher between the two distributions with distinguishing advantage $\epsilon$, then there exists a p.p.t. distinguisher for $G$ with advantage $\geq \frac{\epsilon}{L}$. ☐

Now, we are ready to prove that the GGM PRF is secure.

**Theorem 4.7.** *The GGM PRF construction is secure over many messages.*

As with the previous lecture, this proof is lengthy, so we will split it into its own section.

## Proof

Assume for contradiction that there is a p.p.t. distinguisher $D$ and a polynomial function $p(n)$ such that

$$|\Pr[k \leftarrow \mathcal{K} : D^{\text{Left}(\cdot,\cdot)}(1^n) = 1] - \Pr[k \leftarrow \mathcal{K} : D^{\text{Right}(\cdot,\cdot)}(1^n) = 1]| \geq \frac{1}{p(n)}$$

We now proceed with the hybrid argument, applying it layer by layer:

- **Hybrid 0:** The standard GGM PRF tree.
- **Hybrid 1:** The GGM PRF tree, but the top layer is cut off, and $G_0(s)$ and $G_1(s)$ are replaced with random strings $s_0$ and $s_1$, respectively.
- **Hybrid 2:** Cut off $s_0, s_1$ and replace the next layer with random strings $s_{00}, s_{01}, s_{10}, s_{11}$.
- Continue this process until **Hybrid** $\ell$, where each leaf contains a truly random string. These hybrids are efficiently computable through lazy evaluation.

Define $p_i := \Pr[f \leftarrow H_i : D^f(1^n) = 1]$. We know that $p_0 - p_\ell \geq \epsilon$. By the hybrid argument, for some $i$, we have $p_i - p_{i+1} \geq \frac{\epsilon}{\ell}$, where $\ell$ is the depth of the tree.

Next, we will prove the following lemma:

**Lemma 4.8.** *A distinguisher with advantage $\frac{\epsilon}{\ell}$ between the hybrids $i-1$ and $i$ implies a distinguisher with advantage $\geq \frac{\epsilon}{q\ell}$ for the PRG, where $q$ is the number of queries that $D$ makes.*

*Proof.* By the PRG repetition lemma, it suffices to design a repeated PRG breaker $D'(y_1 \dots y_q)$ where $y_i$ is either a random $2n$-bit string $y_i^L$ or a pseudo-random bit string $y_i^R$.

Take the $i$-th level of the tree and label the node that the query passes through as $y_i^L$ (left) and $y_i^R$ (right).

Here is the construction:

1. Parse $y_i \to (y_i^L, y_i^R)$.
2. Construct an oracle: Place these $y_i$ into the $i$-th level of the tree, creating a truncated GGM tree. Now, check if $(y_1, y_2, \dots)$ is pseudorandom or random. The former implies Hybrid $i$, while the latter implies Hybrid $i+1$.

3. Let the distinguisher $D$ interact with this oracle. If $D$ queries Hybrid $i$, $D'$ outputs 'pseudorandom'. If $D$ queries Hybrid $i+1$, $D'$ outputs 'random'.

$\square$

Next time, we will look at Message Authentication Codes (MACs).

# 5 Message Authentication Codes

## 5.1 PRF Applications

Now that we have a solid understanding of Pseudorandom Functions (PRFs), let's explore their applications. Suppose that Eve can eavesdrop on and modify communications and wishes to impersonate Alice. We'll show how PRFs can mitigate this problem.

## 5.2 Unpredictability of PRFs

Let $f_s : \{0,1\}^\ell \to \{0,1\}^m$ be a PRF, and consider an adversary Eve who requests and receives $f_s(x_1), f_s(x_2), \ldots, f_s(x_q)$ for a polynomial number of queries $q = q(n)$. There are two key questions:

- Can Eve predict $f_s(x^*)$ for some $x^*$ of her choosing, where $x^* \notin \{x_1, x_2, \ldots, x_q\}$?
- How well can she do this?

Here's a useful lemma to address this issue:

**Lemma 5.1.** *If Eve succeeds with probability $\frac{1}{2^m} + \frac{1}{p(n)}$, then she has broken the PRF security.*

*Proof.* This probability is negligible in $n$ if $m$ is sufficiently large, i.e., $\omega(\log n)$. □

In Lecture 3, we established that unpredictability is equivalent to indistinguishability for individual bits. For PRFs, we will show that indistinguishability implies unpredictability, but the converse does not necessarily hold.

## 5.3 Challenge-Response Protocol

Consider an interaction between an ID card, which has a PRF key $s$, and a device that knows everyone's ID number $ID$ and the PRF key $s$. The protocol for interactions is as follows:

1. The device sends a random challenge $r$ to the ID card.
2. The ID card computes $f_s(r)$ and sends $(ID, f_s(r))$ back to the device.

**Theorem 5.2.** *This protocol is secure.*

*Proof.* Proof sketch: The adversary collects $(r_i, f_s(r_i))$ for a polynomial number of challenges $r_i$, and must produce $f_s(r^*)$ for a fresh random $r^*$ in order to impersonate Alice. This is computationally difficult as long as the input and output lengths of the PRF are sufficiently large, i.e., $\omega(\log n)$. □

## 5.4 PRFs for Message Authentication Codes (MACs)

One issue with the one-time pad is that it is malleable—an adversary can modify the message $m$ to $m'$. Similarly, in stateless secret-key encryption, an adversary can modify a ciphertext $(r, f_k(r) \oplus m)$ into a different ciphertext $(r, f_k(r) \oplus m')$. This is where Message Authentication Codes (MACs) come into play.

A MAC produces a tag for a given message, and the resulting message and tag can be verified. However, someone who possesses several messages and their corresponding tags cannot generate a new valid message-tag pair.

For instance, if Alice and Bob share a key $k$, Alice can send $MAC_k(m) = f_s(m)$ to Bob. If Alice and Bob use two keys $k$ and $k'$, Alice sends Bob the ciphertext $(c = (x, f_k(x) \oplus m), \text{tag} = f_{k'}(c))$. An adversary has two possible options:

- Change the tag. The ciphertext will fail verification because each message has a unique tag.
- Modify $c$. This requires computing the PRF on the new ciphertext, which is computationally infeasible.

MACs provide integrity but not privacy. However, combining encryption with a MAC solves this problem.

## 5.5 Applications to Learning Theory

Here is an example of how PRFs are applied outside of cryptography, in the field of learning theory:

**Theorem 5.3** (Kearns and Valiant, 1994)**.** *Assuming the existence of PRFs, there are hypothesis classes that cannot be learned by polynomial-time algorithms.*

## 6 Number Theory

The remainder of Lecture 5, all of Lecture 6, and the first part of Lecture 7 cover a comprehensive set of resources on number theory, specifically created for this class. I will refer you to the following handouts:

- Number Theory Handout for MIT 6.875
- Lecture Notes on the Complexity of Some Problems in Number Theory

These resources are exhaustive, so I will not repeat the material here.

# 7 The Goldreich-Levin Theorem

## 7.1 One-Way Functions

One-way functions (OWFs) were introduced intuitively in the handout. Now, let's define them more rigorously.

**Definition 7.1.** *A function (family) $\{F_n\}_{n \in \mathbb{N}}$ where $F_n : \{0,1\}^n \to \{0,1\}^{m(n)}$ is* ***one-way*** *if, for every probabilistic polynomial-time (ppt) adversary $A$, there is a negligible function $\mu$ such that:*

$$Pr[x \leftarrow \{0,1\}^n; y = F_n(x); A(1^n, y) = x' \mid y = F_n(x')] \leq \mu(n)$$

Note that while we can always find an inverse with unbounded time, finding an inverse should be hard for ppt algorithms.

**Definition 7.2.** *A* ***one-way permutation*** *is a one-to-one one-way function with $m(n) = n$.*

## 7.2 Hardcore Bits

If $F$ is a one-way function (OWF), it is hard to compute a preimage of $F(x)$ for a randomly chosen $x$. But is there a bit of $x$ that is hard to guess with probability significantly better than $\frac{1}{2}$?

More rigorously:

**Definition 7.3.** *For any function (family) $F : \{0,1\}^n \to \{0,1\}^m$, a bit $i = i(n)$ is* ***hardcore*** *if, for every ppt adversary $A$, there is a negligible function $\mu$ such that:*

$$Pr[x \leftarrow \{0,1\}^n; y = F_n(x); A(y) = x_i] \leq \frac{1}{2} + \mu(n)$$

Here's an exercise: Prove that there exist functions that are one-way, yet every bit is relatively easy to predict.

Now, we generalize.

**Definition 7.4.** *For any function (family) $F : \{0,1\}^n \to \{0,1\}^m$, a function $B : \{0,1\}^n \to \{0,1\}$ is a* ***hardcore predicate*** *if, for every ppt adversary $A$, there is a negligible function $\mu$ such that:*

$$Pr[x \leftarrow \{0,1\}^n; y = F(x); A(y) = B(x)] \leq \frac{1}{2} + \mu(n)$$

For the purposes of this class, a hardcore bit is equivalent to a hardcore predicate.

## 7.3 OWFs Imply PRGs

Let's now construct a Pseudorandom Generator (PRG) from a one-way function.

Let $F$ be a one-way permutation, and $B$ an associated hardcore predicate for $F$. Then, define the PRG $G(x) = F(x)\|B(x)$.

**Theorem 7.5.** *$G$ is a PRG, assuming $F$ is a one-way permutation.*

*Proof.* Assume, for contradiction, that $G$ is not a PRG. Therefore, there exists a next-bit predictor $D$, an index $i$, and a polynomial function $p$ such that:

$$\Pr[x \leftarrow \{0,1\}^n; y = G(x) : D(y_{1\ldots i-1}) = y_i] \geq \frac{1}{2} + \frac{1}{p(n)}$$

The index $i$ must be $n+1$, because the first $n$ bits are random. Thus:

$$\Pr[x \leftarrow \{0,1\}^n; D(F(x)) = B(x)] \geq \frac{1}{2} + \frac{1}{p(n)}$$

This implies that $D$ is a hardcore predictor, as desired. □

## 7.4 The Goldreich-Levin Theorem

Let's now aim for a universal hardcore predicate: a single predicate $B$ such that it is hard to guess $B(x)$ given $F(x)$.

Unfortunately, this is not possible. If $f(x)$ is a one-way function, then $f'(x) = f(x)\|B(x)$ is also one-way. However, $B(x)$ is not a hardcore predicate for $f'(x)$, leading to a contradiction.

Fortunately, we can fix this by tweaking the statement.

**Theorem 7.6.** ***(Goldreich-Levin Theorem)***

*Let $\{B_r : \{0,1\}^n \rightarrow \{0,1\}\}$ be a collection of predicates (one for each $r$) where:*

$$B_r(x) = \langle r, x\rangle = \sum_{i=1}^{n} r_i x_i \mod 2$$

*Then, a random $B_r$ is hardcore for every one-way function $F$. Specifically, for every OWF $F$, and every ppt $A$, there exists a negligible function $\mu$ such that:*

$$Pr[x \leftarrow \{0,1\}^n; r \leftarrow \{0,1\}^n; A(F(x), r) = B_r(x)] \leq \frac{1}{2} + \mu(n)$$

A proof of this result was presented in class, assuming both a perfect and a nearly-perfect predictor (simplified versions of the theorem). However, some details were omitted, and it felt somewhat handwavy, so I won't include it here.

Two interesting interpretations:

- For every one-way function $F$, there is a related one-way function $F'(x,r) = (F(x), r)$, which has a deterministic hardcore predicate.
- For every one-way function $F$, there exists a (non-uniform) hardcore predicate $\langle r_F, x\rangle$. **(Open problem: remove the non-uniformity of this statement.)**

## 7.5 The Coding-Theoretic View of Goldreich-Levin

The Goldreich-Levin theorem has significantly influenced a branch of coding theory. Here are some related results:

- $x \to (\langle x, r\rangle)_{r\in\{0,1\}^n}$ can be viewed as a highly redundant, exponentially long encoding of $x$, which is the Hadamard code.
- $P(F(x), r)$ can be thought of as providing access to a noisy codeword.
- What we proved is a unique decoding algorithm for the Hadamard code with an error rate of $\frac{1}{4} - \frac{1}{p(n)}$.
- The real proof involves a list-decoding algorithm for the Hadamard code with an error rate of $\frac{1}{2} - \frac{1}{p(n)}$.