From PRG to IND-CPA

Presenter: LIU Yi

Route

- $OWF \rightarrow OWP\ Axiom \rightarrow PRG\ Axiom \rightarrow PRG$
- PRG

$PRF \rightarrow IND - CPA$

PRP(block cipher)

$IND - CCA$

$CMA - Secure\ MAC$

Route

- $OWF \rightarrow OWP\ Axiom \rightarrow PRG\ Axiom \rightarrow PRG$

$PRF \rightarrow IND - CPA$

- PRG

PRP(block cipher)

$IND - CCA$

$CMA - Secure\ MAC$

Asymptotic Notation

- If f is a function mapping $\{0,1\}^*$ to $\{0,1\}$, then

  f has *linear time* ($O(n)$) algorithm if there is a constant c s.t. f's restriction to $\{0,1\}^n$ can be computed in at most cn steps for every $n \in \mathbb{N}$.

  f has *polynomial time* ($n^{O(1)}$) algorithm if there are constants c, d s.t. f's restriction to $\{0,1\}^n$ can be computed in at most cn^d steps for every $n \in \mathbb{N}$.

  f has *super-polynomial time* ($n^{\omega(1)}$) algorithm if for all constants c, d and sufficiently large n, f's restriction to $\{0,1\}^n$ can not be computed in cn^d steps.

Probabilities

- Compare $2^{-n}, 2^{-n/10}, 2^{-n^{1/3}}$ vs. $1/10, 1/n, 1/n^2$

  A function $\epsilon : \mathbb{N} \to [0,1]$ is *polynomially bounded* if $\epsilon(n) \geq 1/n^{O(1)}$.

  E.g., $\epsilon(n) = 1/10, 1/n^2, 1/n^5 \log n$

  A function $\epsilon : \mathbb{N} \to [0,1]$ is *negligible* if $\epsilon(n) < 1/n^{w(1)}$.

  E.g., $\epsilon(n) = 2^{-n}, 2^{-\sqrt{n}}, n^{-\log n}$

  $$negl(n) + negl(n) = negl(n)$$
  $$poly(n)negl(n) = negl(n)$$

  We use the convention that *efficient computation is equal to polynomial-time* (useful and reasonable).

Computational Security

- **Definition 2.6** Let (E, D) be an encryption scheme that uses n-bit keys to encrypt $\ell(n)$-length messages. (E, D) is *computationally secure* if for every polynomial-time algorithm $Eve : \{0,1\}^* \to \{0,1\}$, every polynomially bounded $\epsilon : \{0,1\}^* \to [0,1]$, n, and $x_0, x_1 \in \{0,1\}^{\ell(n)}$,

$$|\Pr[Eve(E_{U_n}(x_0)) = 1] - \Pr[Eve(E_{U_n}(x_1)) = 1]| < \epsilon(n).$$

Computational Security

- **Definition 2.6** Let (E, D) be an encryption scheme that uses n-bit keys to encrypt $\ell(n)$-length messages. (E, D) is *computationally secure* if for every polynomial-time algorithm $Eve : \{0,1\}^* \to \{0,1\}$, every polynomially bounded $\epsilon : \{0,1\}^* \to [0,1]$, n, and $x_0, x_1 \in \{0,1\}^{\ell(n)}$,

$$|\Pr[Eve(E_{U_n}(x_0)) = 1] - \Pr[Eve(E_{U_n}(x_1)) = 1]| < \epsilon(n).$$

  Theorem 3.1 (Main theorem on *computational security*) If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

Computational Indistinguishability

- **Definition 3.2** Let $\{X_n\}$, $\{Y_n\}$ be sequences of distributions with X_n, Y_n ranging over $\{0,1\}^{\ell(n)}$ for some $\ell(n) = n^{O(1)}$. $\{X_n\}$ and $\{Y_n\}$ are *computationally indistinguishable* ($X_n \approx Y_n$) if for every polnomial-time algorithm A and polynomially-bounded ϵ, and sufficiently large n,

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leq \epsilon(n).$$

Computational Indistinguishability

- **Definition 3.2** Let $\{X_n\}$, $\{Y_n\}$ be sequences of distributions with X_n, Y_n ranging over $\{0,1\}^{\ell(n)}$ for some $\ell(n) = n^{O(1)}$. $\{X_n\}$ and $\{Y_n\}$ are *computationally indistinguishable* ($X_n \approx Y_n$) if for every polnomial-time algorithm A and polynomially-bounded ϵ, and sufficiently large n,

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leq \epsilon(n).$$

Note: Sometimes we say two distributions X and Y are computationally indistinguishable, when we mean that they are part of two computationally indistinguishable sequences.

Properties of $X_n \approx Y_n$

- **Properties of $X_n \approx Y_n$**
  - $\diamond$ reflexive
  - $\diamond$ symmetric
  - $\diamond$ transitive
  - $\diamond$ If $X_n \approx Y_n$ and f is a polynomial time computable function then $f(X_n) \approx f(Y_n)$
  - $\diamond$ If $X_n \approx Y_n$, then for every $m < n$, the *truncation* of X_n to the first m bits is indistinguishable from the truncation of Y_n to the first m bits.

Properties of $X_n \approx Y_n$

- **Properties of $X_n \approx Y_n$**
  - ◇ reflexive
  - ◇ symmetric
  - ◇ transitive

$\approx$ is an equivalence relation!

◇ If $X_n \approx Y_n$ and f is a polynomial time computable function then $f(X_n) \approx f(Y_n)$

◇ If $X_n \approx Y_n$, then for every $m < n$, the *truncation* of X_n to the first m bits is indistinguishable from the truncation of Y_n to the first m bits.

Properties of $X_n \approx Y_n$

- **Properties of** $X_n \approx Y_n$

  ◇ reflexive

  ◇ symmetric $\approx$ is an equivalence relation!

  ◇ transitive

  ◇ If $X_n \approx Y_n$ and f is a polynomial time computable function then $f(X_n) \approx f(Y_n)$

  ◇ If $X_n \approx Y_n$, then for every $m < n$, the *truncation* of X_n to the first m bits is indistinguishable from the truncation of Y_n to the first m bits.

Proof of transitivity If $X_n \approx Y_n$ and $Y_n \approx Z_n$, then $X_n \approx Z_n$.

Properties of $X_n \approx Y_n$

- **Properties of $X_n \approx Y_n$**
  - ◇ reflexive
  - ◇ symmetric
  - ◇ transitive

  $\approx$ is an equivalence relation!

  - ◇ If $X_n \approx Y_n$ and f is a polynomial time computable function then $f(X_n) \approx f(Y_n)$
  - ◇ If $X_n \approx Y_n$, then for every $m < n$, the *truncation* of X_n to the first m bits is indistinguishable from the truncation of Y_n to the first m bits.

Proof of transitivity If $X_n \approx Y_n$ and $Y_n \approx Z_n$, then $X_n \approx Z_n$.

$$
\begin{aligned}
&\Pr[A(X)=1]-\Pr[A(Z)=1] \\
&\quad = \Pr[A(X)=1]-\Pr[A(Y)=1]+\Pr[A(Y)=1]-\Pr[A(Z)=1] \\
&\quad \leq |\Pr[A(X)=1]-\Pr[A(Y)=1]|+|\Pr[A(Y)=1]-\Pr[A(Z)=1]| \\
&\quad \leq 2\epsilon
\end{aligned}
$$

Polynomial Transitivity

- **Proof of transitivity** If $X_n \approx Y_n$ and $Y_n \approx Z_n$, then $X_n \approx Z_n$.

$$\begin{aligned}
&\Pr[A(X)=1]-\Pr[A(Z)=1] \\
&\quad = \Pr[A(X)=1]-\Pr[A(Y)=1]+\Pr[A(Y)=1]-\Pr[A(Z)=1] \\
&\quad \leq |\Pr[A(X)=1]-\Pr[A(Y)=1]|+|\Pr[A(Y)=1]-\Pr[A(Z)=1]| \\
&\quad \leq 2\epsilon
\end{aligned}$$

Proof of transitivity can be generalized to a polynomial number m of distributions $X^1, X^2, \ldots, X^m$, where $X^i \approx X^{i+1}$ for every i. Then we have $X^1 \approx X^m$.

Polynomial Transitivity

- **Proof of transitivity** If $X_n \approx Y_n$ and $Y_n \approx Z_n$, then $X_n \approx Z_n$.

$$\begin{aligned}
&\Pr[A(X)=1]-\Pr[A(Z)=1] \\
&= \Pr[A(X)=1]-\Pr[A(Y)=1]+\Pr[A(Y)=1]-\Pr[A(Z)=1] \\
&\le |\Pr[A(X)=1]-\Pr[A(Y)=1]|+|\Pr[A(Y)=1]-\Pr[A(Z)=1]| \\
&\le 2\epsilon
\end{aligned}$$

Proof of transitivity can be generalized to a polynomial number m of distributions $X^1, X^2, \ldots, X^m$, where $X^i \approx X^{i+1}$ for every i. Then we have $X^1 \approx X^m$.

This is also called *hybrid argument*, and will be used in proof later.

Computational Security Recall

- **Definition 2.6** Let (E, D) be an encryption scheme that uses n-bit keys to encrypt $\ell(n)$-length messages. (E, D) is *computationally secure* if for every polynomial-time algorithm $Eve : \{0,1\}^* \to \{0,1\}$, every polynomially bounded $\epsilon : \{0,1\}^* \to [0,1]$, n, and $x_0, x_1 \in \{0,1\}^{\ell(n)}$,

$$|\Pr[Eve(E_{U_n}(x_0)) = 1] - \Pr[Eve(E_{U_n}(x_1)) = 1]| < \epsilon(n).$$

Computational Security Recall

- **Definition 2.6** Let (E, D) be an encryption scheme that uses n-bit keys to encrypt $\ell(n)$-length messages. (E, D) is *computationally secure* if for every polynomial-time algorithm $Eve : \{0,1\}^* \to \{0,1\}$, every polynomially bounded $\epsilon : \{0,1\}^* \to [0,1]$, n, and $x_0, x_1 \in \{0,1\}^{\ell(n)}$,

$$|\Pr[Eve(E_{U_n}(x_0)) = 1] - \Pr[Eve(E_{U_n}(x_1)) = 1]| < \epsilon(n).$$

$E_{U_n}(x_0) \approx E_{U_n}(x_1)$ for every two messages x_0, x_1.

Computational Security Recall

- **Definition 2.6** Let (E, D) be an encryption scheme that uses n-bit keys to encrypt $\ell(n)$-length messages. (E, D) is *computationally secure* if for every polynomial-time algorithm $Eve : \{0,1\}^* \to \{0,1\}$, every polynomially bounded $\epsilon : \{0,1\}^* \to [0,1]$, n, and $x_0, x_1 \in \{0,1\}^{\ell(n)}$,

$$|\Pr[Eve(E_{U_n}(x_0)) = 1] - \Pr[Eve(E_{U_n}(x_1)) = 1]| < \epsilon(n).$$

$E_{U_n}(x_0) \approx E_{U_n}(x_1)$ for every two messages x_0, x_1.

More formally, we say that for every two sequences of messages $\{x_0^n\}$ and $\{x_1^n\}$, where $x_0^n, x_1^n \in \{0,1\}^{\ell(n)}$, the two sequences $\{E_{U_n}(x_0^n)\}$ and $\{E_{U_n}(x_1^n)\}$ are computationally indistinguishable.

Pseudorandomness

- **Definition 3.3** A distribution $\{X_n\}$ is *pseudorandom* if it is computationally inditinguishable from the *uniform distribution*.

Pseudorandomness

- **Definition 3.3** A distribution $\{X_n\}$ is *pseudorandom* if it is computationally inditinguishable from the *uniform distribution*.

  Definition 3.4 (*PRG*) A polynomial-time-computable deterministic function G mapping n bit strings into $\ell(n)$ bit strings for $\ell(n) \geq n$ is called a *pseudorandom generator* (PRG) if $G(U_n) \approx U_{\ell(n)}$. The function $\ell(n)$ is called the *stretch* of the PRG.

Pseudorandomness

- **Definition 3.3** A distribution $\{X_n\}$ is *pseudorandom* if it is computationally inditinguishable from the *uniform distribution*.

Definition 3.4 (*PRG*) A polynomial-time-computable deterministic function G mapping n bit strings into $\ell(n)$ bit strings for $\ell(n) \geq n$ is called a *pseudorandom generator* (PRG) if $G(U_n) \approx U_{\ell(n)}$. The function $\ell(n)$ is called the *stretch* of the PRG.

Note: It is trivial to construct a PRG with $\ell(n) = n$. Because of the truncation property, a PRG with $\ell(n)$ trivially give a PRG with $\ell'(n) < \ell(n)$.

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Theorem 3.5 $\rightarrow$ **Theorem 3.6** $\rightarrow$ **Theorem 3.7** $\rightarrow$ **Theorem 3.1**

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Proof idea: Given such a PRG, construct such a encryption scheme.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Proof idea: Given such a PRG, construct such a encryption scheme.

  Let G be the PRG mapping n bit strings to $\ell(n)$ bit strings.
  $E_k(x) = x \oplus G(k)$
  $D_k(y) = y \oplus G(k)$
  Prove that this encryption scheme is computationally secure.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Proof idea: Given such a PRG, construct such a encryption scheme.

  Let G be the PRG mapping n bit strings to $\ell(n)$ bit strings.
  $E_k(x) = x \oplus G(k)$
  $D_k(y) = y \oplus G(k)$
  Prove that this encryption scheme is computationally secure.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Proof idea: Given such a PRG, construct such a encryption scheme.

  Let G be the PRG mapping n bit strings to $\ell(n)$ bit strings.
  $E_k(x) = x \oplus G(k)$
  $D_k(y) = y \oplus G(k)$
  Prove that this encryption scheme is computationally secure.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  It follows from this claim that, forevery pair of messages x_0, x_1, we have $E_{U_n}(x_0) \approx U_{\ell(n)} \approx E_{U_n}(x_1)$.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  Proof. (By contradiction).

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  Proof. (By contradiction).

  Suppose that there exists a polynomial-time A such that
  $$|\Pr[A(G(U_n) \oplus x) = 1] - \Pr[A(U_{\ell(n)}) = 1]| \geq \epsilon$$

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  Proof. (By contradiction).

  Suppose that there exists a polynomial-time A such that
  $$|\Pr[A(G(U_n) \oplus x) = 1] - \Pr[A(U_{\ell(n)}) = 1]| \geq \epsilon$$
  $E_{U_n}(x)$ is not pseudorandom.

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  Proof. (By contradiction).

  Suppose that there exists a polynomial-time A such that
  $$|\Pr[A(G(U_n) \oplus x) = 1] - \Pr[A(U_{\ell(n)}) = 1]| \geq \epsilon$$
  $E_{U_n}(x)$ is not pseudorandom.

  Define $B : \{0,1\}^{\ell(n)} \to \{0,1\}$ as: $B(y) = A(y \oplus x)$, which means $A(z) = B(z \oplus x)$. The running tims of B is the same as that of A, but we have
  $$|\Pr[B(G(U_n)) = 1] - \Pr[B(U_{\ell(n)} \oplus x) = 1]| \geq \epsilon$$

PRG implies computational security

- **Theorem 3.7**
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Claim 3.7.1
  For every message x, the distribution $E_{U_n}(x)$ is *pseudorandom*.

  Proof. (By contradiction).

  Suppose that there exists a polynomial-time A such that

  $$|\Pr[A(G(U_n) \oplus x) = 1] - \Pr[A(U_{\ell(n)}) = 1]| \geq \epsilon$$

  $E_{U_n}(x)$ is not pseudorandom.

  Define $B : \{0,1\}^{\ell(n)} \to \{0,1\}$ as: $B(y) = A(y \oplus x)$, which means $A(z) = B(z \oplus x)$. The running tims of B is the same as that of A, but we have

  $$|\Pr[B(G(U_n)) = 1] - \Pr[B(U_{\ell(n)} \oplus x) = 1]| \geq \epsilon$$

  Since $U_{\ell(n)} \oplus x \equiv U_{\ell(n)}$, this contradicts to the fact that G is a PRG.

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Theorem 3.5 $\rightarrow$ **Theorem 3.6** $\rightarrow$ **Theorem 3.7** $\rightarrow$ **Theorem 3.1**

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Theorem 3.5 $\rightarrow$ **Theorem 3.6** $\rightarrow$ **Theorem 3.7** $\rightarrow$ **Theorem 3.1**

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Theorem 3.5 $\rightarrow$ **Theorem 3.6** $\rightarrow$ **Theorem 3.7** $\rightarrow$ **Theorem 3.1**

Stretch $n + 1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof idea: Given such a PRG G' mapping n bits to $n+1$ bits, construct such a PRG G mapping n bits to $\ell(n)$ bits. Then use the *hybrid* technique.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof idea: Given such a PRG G' mapping n bits to $n+1$ bits, construct such a PRG G mapping n bits to $\ell(n)$ bits. Then use the *hybrid* technique.
  For a string $x \in \{0,1\}^k$, and $i < j \leq k$, $x_{[i \ldots j]}$ is $x_i x_{i+1} \ldots x_j$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof idea: Given such a PRG G' mapping n bits to $n+1$ bits, construct such a PRG G mapping n bits to $\ell(n)$ bits. Then use the *hybrid* technique.
  For a string $x \in \{0,1\}^k$, and $i < j \leq k$, $x_{[i\ldots j]}$ is $x_i x_{i+1} \ldots x_j$.

  G:
  Input: $x \in \{0,1\}^n$
  $j \leftarrow 0$
  $x^{(0)} \leftarrow x$
  while $j < \ell(n)$:
  $\quad j \leftarrow j + 1$
  $\quad x^{(j)} \leftarrow G'_n(x^{(j-1)}_{[1\ldots n]})$
  $\quad$ output $x^{(j)}_{n+1}$
  end while

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

G:

Input: $x \in \{0,1\}^n$
$j \leftarrow 0$
$x^{(0)} \leftarrow x$
while $j < \ell(n)$:
    $j \leftarrow j+1$
    $x^{(j)} \leftarrow G'_n(x^{(j-1)}_{[1\ldots n]})$
    output $x^{(j)}_{n+1}$
end while

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

G:

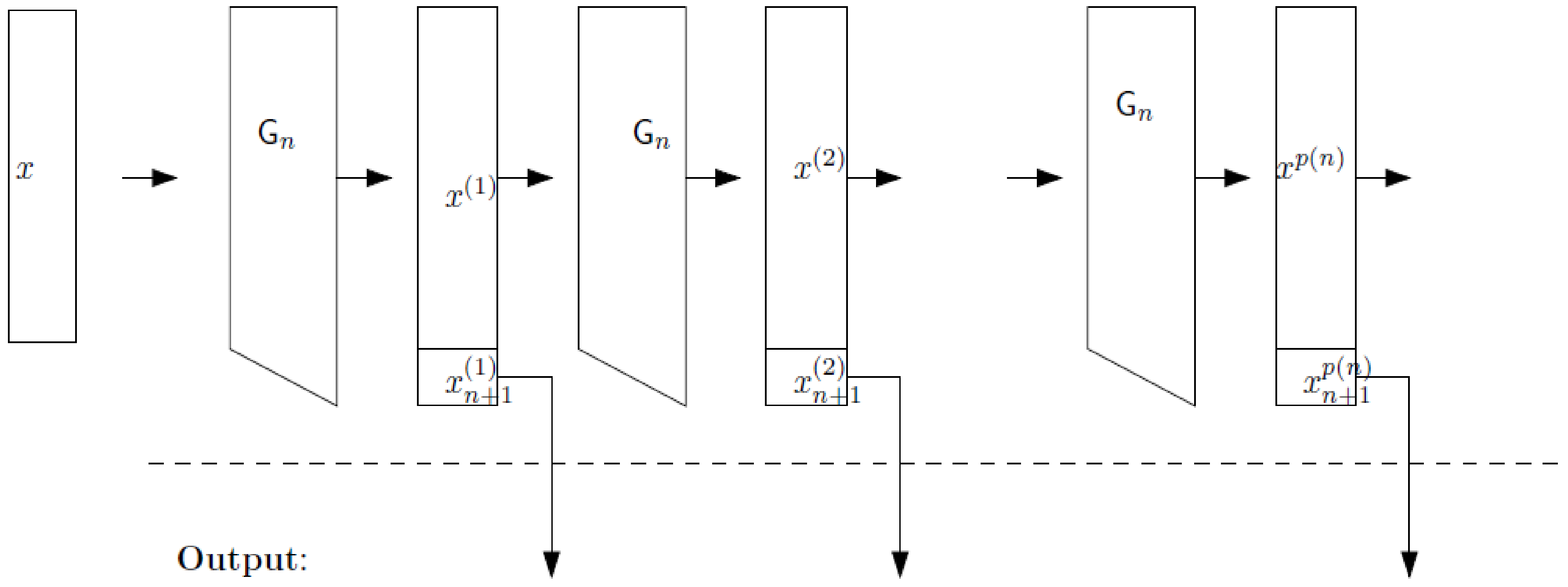


Figure 1: Extending output of pseudorandom genertator

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof. It remains to prove that $G(U_n) \approx U_{\ell(n)}$

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof. It remains to prove that $G(U_n) \approx U_{\ell(n)}$

  We define random variables $Y^{(0)}, Y^{(1)}, \ldots, Y^{(\ell(n))}$ over $\{0,1\}^{\ell(n)}$. $Y^{(i)}$ corresponds to running the pseudorandom generator from the i-th iteration onwards, starting from the uniform distribution U_{n+i}.

Stretch $n + 1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof. It remains to prove that $G(U_n) \approx U_{\ell(n)}$

  We define random variables $Y^{(0)}, Y^{(1)}, \ldots, Y^{(\ell(n))}$ over $\{0,1\}^{\ell(n)}$. $Y^{(i)}$ corresponds to running the pseudorandom generator from the i-th iteration onwards, starting from the uniform distribution U_{n+i}.

  More precisely, $Y^{(i)}$ is obtained by concatenating random i bits to the output of the following algorithm $G^{\ell(n)-i}$ on input $x \leftarrow_R \{0,1\}^n$:

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Proof. It remains to prove that $G(U_n) \approx U_{\ell(n)}$

G^{j_0}: **Input:** $x \in \{0,1\}^n$
$j \leftarrow j_0$
$x^{(j)} \leftarrow x$
while $j < \ell(n)$:
  $y \leftarrow G'(x^{(j-1)})$
  $x^{(j+1)} \leftarrow y_{[1\ldots n]}$
  output y_{n+1}
  $j \leftarrow j+1$
end while

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  More precisely, $Y^{(i)}$ is obtained by concatenating random i bits to the output of the following algorithm G^{m-i} on input $x \leftarrow_R \{0,1\}^n$:
  Note that $Y^{(0)} \approx G(U_n)$, and $Y^{(\ell(n))} \approx U_{\ell(n)}$. Thus, we need to show that $Y^{(0)} \approx Y^{(\ell(n))}$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  Note that

$$\begin{aligned} Y^{(i)} &= U_i || G^{\ell(n)-i}(U_n) \\ Y^{(i+1)} &= U_{i+1} || G^{\ell(n)-i-1}(U_n) \end{aligned}$$

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  Note that

$$\begin{aligned} Y^{(i)} &= U_i || G^{\ell(n)-i}(U_n) \\ Y^{(i+1)} &= U_{i+1} || G^{\ell(n)-i-1}(U_n) \end{aligned}$$

  It suffices to prove $X = G^{(\ell(n)-i)}(U_n) \approx Y = U_1 || G^{(\ell(n)-i-1)}(U_n)$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  Note that

$$
\begin{aligned}
Y^{(i)} &= U_i || G^{\ell(n)-i}(U_n) \\
Y^{(i+1)} &= U_{i+1} || G^{\ell(n)-i-1}(U_n)
\end{aligned}
$$

  It suffices to prove $X = G^{(\ell(n)-i)}(U_n) \approx Y = U_1 || G^{(\ell(n)-i-1)}(U_n)$.
  Define $f : \{0,1\}^{n+1} \to \{0,1\}^{\ell(n)-i}$ as:
  $f(y) = y_{n+1} || G^{(\ell(n)-i-1)}(y_{[1\ldots n]})$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  Note that

$$
\begin{aligned}
Y^{(i)} &= U_i || G^{\ell(n)-i}(U_n) \\
Y^{(i+1)} &= U_{i+1} || G^{\ell(n)-i-1}(U_n)
\end{aligned}
$$

  It suffices to prove $X = G^{(\ell(n)-i)}(U_n) \approx Y = U_1 || G^{(\ell(n)-i-1)}(U_n)$.
  Define $f : \{0,1\}^{n+1} \to \{0,1\}^{\ell(n)-i}$ as:
  $f(y) = y_{n+1} || G^{(\ell(n)-i-1)}(y_{[1...n]})$.
  Then $X = f(G'(U_n))$, and $Y = f(U_{n+1})$.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n+1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  Note that

$$\begin{aligned} Y^{(i)} &= U_i || G^{\ell(n)-i}(U_n) \\ Y^{(i+1)} &= U_{i+1} || G^{\ell(n)-i-1}(U_n) \end{aligned}$$

  It suffices to prove $X = G^{(\ell(n)-i)}(U_n) \approx Y = U_1 || G^{(\ell(n)-i-1)}(U_n)$.
  Define $f : \{0,1\}^{n+1} \to \{0,1\}^{\ell(n)-i}$ as:
  $f(y) = y_{n+1} || G^{(\ell(n)-i-1)}(y_{[1\ldots n]})$.
  Then $X = f(G'(U_n))$, and $Y = f(U_{n+1})$.
  Since $G'(U_n) \approx U_{n+1}$, we have $f(G'(U_n)) \approx f(U_{n+1})$ for every polynomial-time computable function f.

Stretch $n+1$ implies Stretch n^c

- **Theorem 3.6**
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Claim 3.6.1
  For every $i \in [m]$, $Y^{(i)} \approx Y^{(i+1)}$.

  The running time of G is roughly $\ell(n)$ times the running time of G'.

  G:
  Input: $x \in \{0,1\}^n$
  $j \leftarrow 0$
  $x^{(0)} \leftarrow x$
  while $j < \ell(n)$:
  $\quad j \leftarrow j + 1$
  $\quad x^{(j)} \leftarrow G'_n(x^{(j-1)}_{[1\dots n]})$
  $\quad$ output $x^{(j)}_{n+1}$
  end while

Main Theorem and the PRG Axiom

- **Theorem 3.5** (*The PRG Axiom*)
  There exists a PRG with stretch $\ell(n) = n + 1$.

  Theorem 3.1 (Main theorem on *computational security*)
  If the *PRG Axiom* is true, then for every constant c, there exists a computationally secure encryption scheme with message length $\ell(n) = n^c$.

  Theorem 3.6
  If there exists a PRG with stretch $\ell(n) = n + 1$, then for every constant c, there exists a PRG with stretch $\ell(n) = n^c$.

  Theorem 3.7
  If there exists a PRG with stretch $\ell(n)$, then there exists a *computationally secure* encryption scheme with message length $\ell(n)$.

  Theorem 3.5 $\rightarrow$ **Theorem 3.6** $\rightarrow$ **Theorem 3.7** $\rightarrow$ **Theorem 3.1**
  ✓ ✓

Pseudorandom Function (PRF)

- What can a *random* function $F(\cdot)$ from n bits to n bits be?

Pseudorandom Function (PRF)

- What can a *random* function $F(\cdot)$ from n bits to n bits be?

  For each of its possible 2^n inputs x, choose a random n-bit string as the output $F(x)$.

Pseudorandom Function (PRF)

- What can a *random* function $F(\cdot)$ from n bits to n bits be?

  For each of its possible 2^n inputs x, choose a random n-bit string as the output $F(x)$.

  We need $2^n \cdot n$ bits to choose a random function. A function that can be described in n bits is very far from being a random function.

Pseudorandom Function (PRF)

- What can a *random* function $F(\cdot)$ from n bits to n bits be?

  For each of its possible 2^n inputs x, choose a random n-bit string as the output $F(x)$.

  We need $2^n \cdot n$ bits to choose a random function. A function that can be described in n bits is very far from being a random function.

  We will show that, if the PRG Axiom is true, there exists a *pseudorandom function* (PRF) collection that can be described and computed with $poly(n)$ bits but is *indistinguishable* from a random function.

Pseudorandom Function (PRF)

- Let $\mathcal{F} = \{f_s\}_{s \in \{0,1\}^*}$ be a *collection* of functions, and suppose that $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$. Wa say that the collection is *efficiently computable* if the mapping $s, x \mapsto f_s(x)$ is computable in polynomial time. Fix an efficiently computable collection and consider the following two games:

Pseudorandom Function (PRF)

- Let $\mathcal{F} = \{f_s\}_{s \in \{0,1\}^*}$ be a *collection* of functions, and suppose that $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$. Wa say that the collection is *efficiently computable* if the mapping $s, x \mapsto f_s(x)$ is computable in polynomial time. Fix an efficiently computable collection and consider the following two games:

Game 1

• $s \leftarrow_R \{0,1\}^n$

• Eve gets black-box access to the function $f_s(\cdot)$ for as long as it wishes (but within $poly(n)$ running time)

• Eve outputs a bit $v \in \{0,1\}$.

Game 2

• Random $F : \{0,1\}^n \to \{0,1\}^n$

• Eve gets black-box access to the function $F(\cdot)$ for as long as it wishes (but within $poly(n)$ running time)

• Eve outputs a bit $v \in \{0,1\}$.

Pseudorandom Function (PRF)

- Let $\mathcal{F} = \{f_s\}_{s \in \{0,1\}^*}$ be a *collection* of functions, and suppose that $f_s : \{0,1\}^{|s|} \to \{0,1\}^{|s|}$. Wa say that the collection is *efficiently computable* if the mapping $s, x \mapsto f_s(x)$ is computable in polynomial time. Fix an efficiently computable collection and consider the following two games:

  Definition 4.1 $\mathcal{F}$ is a *pseudorandom function* (PRF) ensemble, if for every polynomial-time *Eve* and polynoimially-bounded $\epsilon : \mathbb{N} \to [0,1]$, and large enough n,
  $|\Pr[\textit{Eve outputs } 1 \textit{ in Game } 1] - \Pr[\textit{Eve outputs } 1 \textit{ in Game } 2]| < \epsilon(n)$

Pseudorandom Function (PRF)

- It is not clear whether PRFs exist.

Pseudorandom Function (PRF)

- It is not clear whether PRFs exist.

  Theorem 4.2 (Goldreich, Goldwasser, Micali 1984)

  If the PRG Axiom is true, then there exist PRFs.

Construction of Pseudorandom Function (PRF)

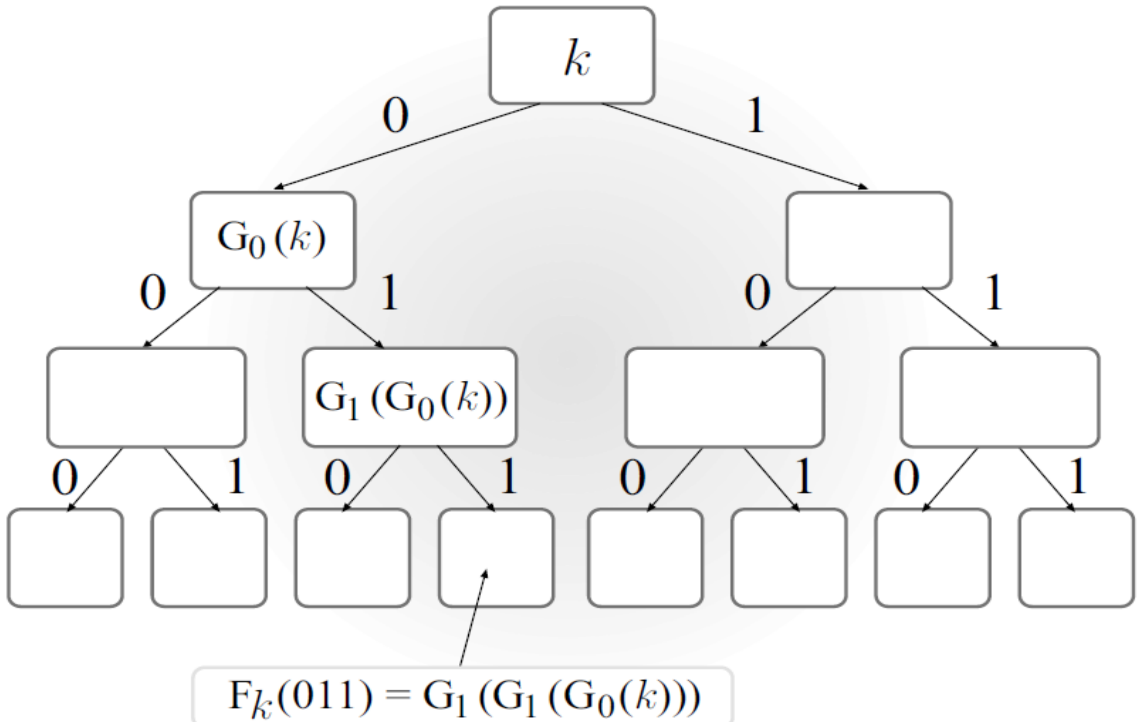
- Suppose that we have a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, we can construct a n-depth full binary tree as follows:

Construction of Pseudorandom Function (PRF)

- Suppose that we have a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, we can construct a n-depth full binary tree as follows:
  - ◇ the root is labeled with a string s (the seed of the function).
  - ◇ for each non-leaf node labeled v, the two children are labeled with $G_0(v) = G(v)_{[1\ldots n]}$ and $G_1(v) = G(v)_{[n+1\ldots 2n]}$.
  - ◇ $f_s(x)$ is the label of the leaf corresponding to x.

Construction of Pseudorandom Function (PRF)

- Suppose that we have a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, we can construct a *n*-depth full binary tree as follows:
  - ◇ the root is labeled with a string s (the seed of the function).
  - ◇ for each non-leaf node labeled v, the two children are labeled with $G_0(v) = G(v)_{[1\dots n]}$ and $G_1(v) = G(v)_{[n+1\dots 2n]}$.
  - ◇ $f_s(x)$ is the label of the leaf corresponding to x.



Construction of Pseudorandom Function (PRF)

- **Theorem 4.2** (Goldreich, Goldwasser, Micali 1984)
  If the PRG Axiom is true, then there exist PRFs.

  Lemma 4.3 If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

Construction of Pseudorandom Function (PRF)

- **Theorem 4.2** (Goldreich, Goldwasser, Micali 1984)
  If the PRG Axiom is true, then there exist PRFs.

  Lemma 4.3 If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Suppose that we have a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, we can construct a n-depth full binary tree as follows:

  ◇ the root is labeled with a string s (the seed of the function).

  ◇ for each non-leaf node labeled v, the two children are labeled with $G_0(v) = G(v)_{[1\ldots n]}$ and $G_1(v) = G(v)_{[n+1\ldots 2n]}$.

  ◇ $f_s(x)$ is the label of the leaf corresponding to x.

Construction of Pseudorandom Function (PRF)

- **Theorem 4.2** (Goldreich, Goldwasser, Micali 1984)
  If the PRG Axiom is true, then there exist PRFs.

  Lemma 4.3 If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Suppose that we have a PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$, we can construct a n-depth full binary tree as follows:

  ◇ the root is labeled with a string s (the seed of the function).

  ◇ for each non-leaf node labeled v, the two children are labeled with $G_0(v) = G(v)_{[1\ldots n]}$ and $G_1(v) = G(v)_{[n+1\ldots 2n]}$.

  ◇ $f_s(x)$ is the label of the leaf corresponding to x.

  For $i \in \{0,1\}^n$, define $f_s(i)$ as

  $$G_{i_n}(G_{i_{n-1}}(\cdots G_{i_1}(s)))$$

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

For $i \in \{0,1\}^n$, define $f_s(i)$ as

$$G_{i_n}(G_{i_{n-1}}(\cdots G_{i_1}(s)))$$

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  For $i \in \{0,1\}^n$, define $f_s(i)$ as

  $$G_{i_n}(G_{i_{n-1}}(\cdots G_{i_1}(s)))$$

  To evaluate $f_s(i)$, we need to evaluate the PRG n times on inputs of length n. If the PRG is efficiently computable, so is the PRF.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  For $i \in \{0,1\}^n$, define $f_s(i)$ as
  $$G_{i_n}(G_{i_{n-1}}(\cdots G_{i_1}(s)))$$

  To evaluate $f_s(i)$, we need to evaluate the PRG n times on inputs of length n. If the PRG is efficiently computable, so is the PRF.

  Proof idea: By contradiction. Suppose that there is an T-time *Eve* that can distinguish between access to $f_s(\cdot)$ and access to a random function with probability at least ϵ. We then convert it to a $poly(T)$-time Eve' that can distinguish between $G(U_n)$ and U_{2n} with probability at least $\epsilon/poly(T)$.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Proof idea: By contradiction. Suppose that there is an T-time *Eve* that can distinguish between access to $f_s(\cdot)$ and access to a random function with probability at least ϵ. We then convert it to an T'-time *Eve'* that can distinguish between $G(U_n)$ and U_{2n}(by reduction). Also use the *hybrid* technique.

  Assumptions of Eve:
  ◇ It makes exactly T queries.
  ◇ It never ask the same questions twice.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Proof idea: By contradiction. Suppose that there is an T-time *Eve* that can distinguish between access to $f_s(\cdot)$ and access to a random function with probability at least ϵ. We then convert it to an T'-time *Eve'* that can distinguish between $G(U_n)$ and U_{2n}(by reduction). Also use the *hybrid* technique.

  Assumptions of Eve:
  ◇ It makes exactly T queries.
  ◇ It never ask the same questions twice.

  Description of the $f_s(\cdot)$ oracle:
  ◇ Initially the tree contains the root labeled with s only.
  ◇ Whenever *Eve* makes a query for $f_s(x)$, the oracle will look at the path from the leaf x to the root.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Description of the $f_s(\cdot)$ oracle:
  ◇ Initially the tree contains the root labeled with s only.
  ◇ Whenever *Eve* makes a query for $f_s(x)$, the oracle will look at the path from the leaf x to the root.

  Whenever the oracle invokes G on a label x of an internal node v, it will label the children of v with $x_0 = G_0(x)$ and $x_1 = G_1(x)$ and erase the label of v (This is OK since the oracle will never use these values again). In all, the oracle needs to make at most $M = T \cdot n$ invocations of G during this process.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Using the *hybrid* technique.
  Define H_i as follows:
  This is Eve's view interacting with the oracle except that for the first i times, when the oracle is supposed to invoke G to label two children of some node v labeled x, the oracle does not do this but does a "fake invocaton": it chooses x_0, x_1 at random from $\{0,1\}^n$, instead of labeling the two childeren with $(x_0, x_1) = G(x)$.

  H_0 – Eve's view then interacting with $f_s(\cdot)$
  H_M – Eve's view then interacting with a random function

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  Using the *hybrid* technique.
  Define H_i as follows:
  This is Eve's view interacting with the oracle except that for the first i times, when the oracle is supposed to invoke G to label two children of some node v labeled x, the oracle does not do this but does a "fake invocaton": it chooses x_0, x_1 at random from $\{0,1\}^n$, instead of labeling the two childeren with $(x_0, x_1) = G(x)$.

  H_0 – Eve's view then interacting with $f_s(\cdot)$
  H_M – Eve's view then interacting with a random function

  It remains to prove that H^i is indistinguishable from H^{i-1}.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  We now prove that H^i is indistinguishable from H^{i-1}.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  We now prove that H^i is indistinguishable from H^{i-1}.

  Suppose that we have a distinguisher D between H^i and H^{i-1}. We will build a distinguisher D' for the PRG G.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  We now prove that H^i is indistinguishable from H^{i-1}.

  Suppose that we have a distinguisher D between H^i and H^{i-1}. We will build a distinguisher D' for the PRG G.

  Input: $y \in \{0,1\}^{2n}$ (y either from U_{2n} or from $G(U_n)$)

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  We now prove that H^i is indistinguishable from H^{i-1}.

  Suppose that we have a distinguisher D between H^i and H^{i-1}. We will build a distinguisher D' for the PRG G.

  Input: $y \in \{0,1\}^{2n}$ (y either from U_{2n} or from $G(U_n)$)

  In the hybrid H^{i-1} the oracle chooses $(x_0, x_1) = G(x)$ and uses that to label v's children, then erases x.

  In the hybrid H^i the oracle chooses x_0 and x_1 at random.

Construction of Pseudorandom Function (PRF)

- **Lemma 4.3** If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a *pseudorandom generator*, then the construction above is a *PRF collection*.

  We now prove that H^i is indistinguishable from H^{i-1}.

  Suppose that we have a distinguisher D between H^i and H^{i-1}. We will build a distinguisher D' for the PRG G.

  Input: $y \in \{0,1\}^{2n}$ (y either from U_{2n} or from $G(U_n)$)

  In the hybrid H^{i-1} the oracle chooses $(x_0, x_1) = G(x)$ and uses that to label v's children, then erases x.

  In the hybrid H^i the oracle chooses x_0 and x_1 at random.

  Simply let $(x_0, x_1) = y$. If $y \sim G(U_n)$ then we get H^{i-1} and if $y \sim U_{2n}$ then we get H^i. Thus, the success of D' in distinguishing $G(U_n)$ and U_{2n} equals the success of D in distinguishing H^{i-1} and H^i.

Chosen Plaintext Attack (CPA) Security

- **Definition 5.1** (*Chosen Plaintext Attack (CPA) secure encryption*)
An encryption scheme (E, D) is *secure* againt *chosen plaintext attack* (CPA secure) if for every polynomial time *Eve*, Eve wins with probability at most $1/2 + negl(n)$ in the following game:

1. The key k is chosen at random in $\{0,1\}^n$ and fixed.
2. Eve gets the length of the key 1^n as input.
3. Eve interacts with E for $t = poly(n)$ rounds as follows: in the i-th round, Eve chooses a message m_i and obtains $c_i = E_k(m_i)$.
4. Then Eve chooses two messages m_0, m_1, and gets $c^* = E_k(m_b)$ for $b \leftarrow_R \{0,1\}$.
5. Eve *wins* if she outputs b.

Chosen Plaintext Attack (CPA) Security

- **Definition 5.1** (*Chosen Plaintext Attack (CPA) secure encryption*)
  An encryption scheme (E, D) is *secure* againt *chosen plaintext attack* (CPA secure) if for every polynomial time *Eve*, Eve wins with probability at most $1/2 + negl(n)$ in the following game:

  1. The key k is chosen at random in $\{0,1\}^n$ and fixed.
  2. Eve gets the length of the key 1^n as input.
  3. Eve interacts with E for $t = poly(n)$ rounds as follows: in the i-th round, Eve chooses a message m_i and obtains $c_i = E_k(m_i)$.
  4. Then Eve chooses two messages m_0, m_1, and gets $c^* = E_k(m_b)$ for $b \leftarrow_R \{0,1\}$.
  5. Eve *wins* if she outputs b.

  Note: CPA security is stronger than computational secrecy, since Step 3. only gives the adversary more power.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.2** (CPA security requires randomization)
  There is no CPA secure (E, D) where E is deterministic.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.2** (CPA security requires randomization)
  There is no CPA secure (E, D) where E is deterministic.

  Proof. Eve will only use a single round of interacting with E where she will ask for the encryption $c_1 = E_k(0^\ell)$. In the second round, Eve will choose $m_0 = 0^\ell$ and $m_1 = 1^\ell$, and get $c^* = E_k(m_b)$. Eve will output 0 if and only if $c^* = c_1$.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.2** (CPA security requires randomization)
  There is no CPA secure (E, D) where E is deterministic.

  Proof. Eve will only use a single round of interacting with E where she will ask for the encryption $c_1 = E_k(0^\ell)$. In the second round, Eve will choose $m_0 = 0^\ell$ and $m_1 = 1^\ell$, and get $c^* = E_k(m_b)$. Eve will output 0 if and only if $c^* = c_1$.

  Note: We need to use a *randomized* (or *probabilistic*) encryption, such that if we encrypt the same message twice we *won't* see two copies of the same ciphertext.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.2** (CPA security requires randomization) There is no CPA secure (E, D) where E is deterministic.

  Proof. Eve will only use a single round of interacting with E where she will ask for the encryption $c_1 = E_k(0^\ell)$. In the second round, Eve will choose $m_0 = 0^\ell$ and $m_1 = 1^\ell$, and get $c^* = E_k(m_b)$. Eve will output 0 if and only if $c^* = c_1$.

  Note: We need to use a *randomized* (or *probabilistic*) encryption, such that if we encrypt the same message twice we *won't* see two copies of the same ciphertext.

  $\mathcal{Q}$: How do we do that?

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.2** (CPA security requires randomization)
  There is no CPA secure (E, D) where E is deterministic.

  Proof. Eve will only use a single round of interacting with E where she will ask for the encryption $c_1 = E_k(0^\ell)$. In the second round, Eve will choose $m_0 = 0^\ell$ and $m_1 = 1^\ell$, and get $c^* = E_k(m_b)$. Eve will output 0 if and only if $c^* = c_1$.

  Note: We need to use a *randomized* (or *probabilistic*) encryption, such that if we encrypt the same message twice we *won't* see two copies of the same ciphertext.

  $\mathcal{Q}$: How do we do that?
  $\mathcal{A}$: Using PRFs.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

$$E_s(m) = (r, f_s(r) \oplus m)$$
$$D_s(r, z) = f_s(r) \oplus z$$

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:
  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$

  Proof. It is straightforward to verify that $D_s(E_s(m)) = m$. We need to show the CPA security property.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$

  Proof. It is straightforward to verify that $D_s(E_s(m)) = m$. We need to show the CPA security property.

  We first show that this scheme will be secure if f_s was a random function, and then use that to derive security.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

$$E_s(m) = (r, f_s(r) \oplus m)$$
$$D_s(r, z) = f_s(r) \oplus z$$

  Let r_i be the random string chosen by E in the i-th round and r^* the string chosen in the last round.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

$$E_s(m) = (r, f_s(r) \oplus m)$$
$$D_s(r, z) = f_s(r) \oplus z$$

  Let r_i be the random string chosen by E in the i-th round and r^* the string chosen in the last round.

  Lemma 5.3.1 The probability that $r^* = r_i$ for some i is at most $poly(n)/2^n$.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:
  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$

  Let r_i be the random string chosen by E in the i-th round and r^* the string chosen in the last round.

  Lemma 5.3.1 The probability that $r^* = r_i$ for some i is at most $poly(n)/2^n$.

  Proof. For a particular i, since r^* is chosen independently of r_i, the probability that $r^* = r_i$ is 2^{-n}. Hence the claim follows from the union bound.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

$$E_s(m) = (r, f_s(r) \oplus m)$$
$$D_s(r, z) = f_s(r) \oplus z$$

  Lemma 5.3.1 The probability that $r^* = r_i$ for some i is at most $poly(n)/2^n$.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:
  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$
  Lemma 5.3.1 The probability that $r^* = r_i$ for some i is at most $poly(n)/2^n$.

  This means that with probability $1 - poly(n)/2^n$ $(1 - negl(n))$, the string r^* is distinct from any string that was chosen before.
  The value $f_s(r^*)$ can be considered as being chosen at random in the final round independent of anything that happened before.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$

  By one-time pad, the distributions $f_s(r^*) \oplus m_0$ and $f_s(r^*) \oplus m_1$ are both equal to U_n. Hence, Eve gets no info about b.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

  $$E_s(m) = (r, f_s(r) \oplus m)$$
  $$D_s(r, z) = f_s(r) \oplus z$$

  By one-time pad, the distributions $f_s(r^*) \oplus m_0$ and $f_s(r^*) \oplus m_1$ are both equal to U_n. Hence, Eve gets no info about b.
  This shows that, if $f_s(\cdot)$ was a random function, Eve would win with probability at most $1/2$.

Chosen Plaintext Attack (CPA) Security

- **Theorem 5.3** (CPA security from PRFs)
  Suppose that $\{f_s\}$ is a PRF collection where $f_s : \{0,1\}^n \to \{0,1\}^\ell$, then the following is a *CPA secure encryption scheme*:

$$E_s(m) = (r, f_s(r) \oplus m)$$
$$D_s(r, z) = f_s(r) \oplus z$$

  By one-time pad, the distributions $f_s(r^*) \oplus m_0$ and $f_s(r^*) \oplus m_1$ are both equal to U_n. Hence, Eve gets no info about b.
  This shows that, if $f_s(\cdot)$ was a random function, Eve would win with probability at most $1/2$.
  If we have some efficient Eve that wins with probability at least $1/2 + \epsilon$ then we can build an adversary Eve' for the PRF as: run the entire game with black box access to $f_s(\cdot)$ and will output 1 iff Eve wins.

Chosen Plaintext Attack (CPA) Security

- By one-time pad, the distributions $f_s(r^*) \oplus m_0$ and $f_s(r^*) \oplus m_1$ are both equal to U_n. Hence, Eve gets no info about b.
  This shows that, if $f_s(\cdot)$ was a random function, Eve would win with probability at most $1/2$.
  If we have some efficient Eve that wins with probability at least $1/2 + \epsilon$ then we can build an adversary Eve' for the PRF as: run the entire game with black box access to $f_s(\cdot)$ and will output 1 iff Eve wins.

  There would be a difference of at least ϵ in the probability it outputs 1 when $f_s(\cdot)$ is random vs. when it is pseudorandom, contradicting the secuirty property of the PRF.

Pesudorandom Permutations (PRPs)

- **Definition 5.4** (Pesudorandom Permutations)
  Let $\ell : \mathbb{N} \to \mathbb{N}$ be some function that is *polynomially bounded* (i.e., there are some $0 < c < C$ such that $n^c < \ell(n) < n^C$ for every n). A collection of functions $\{f_s\}$ where $f_s : \{0,1\}^\ell \to \{0,1\}^\ell$ for $\ell = \ell(|s|)$ is called a *pseudorandom permutation* (*PRP*) collection if:
  1. It is a pseudorandom function collection (i.e., the map, $s, x \mapsto f_s(x)$ is efficiently computatble and there is no efficient distinguisher between $f_s(\cdot)$ with a random s and a random function).
  2. Every function f_s is a permutation of $\{0,1\}^\ell$ (i.e., a one to one and onto map)
  3. There is an efficient algorithm that on input s, y returns $f_s^{-1}(y)$.

Pesudorandom Permutations (PRPs)

- **Definition 5.4** (Pesudorandom Permutations)
  Let $\ell : \mathbb{N} \to \mathbb{N}$ be some function that is *polynomially bounded* (i.e., there are some $0 < c < C$ such that $n^c < \ell(n) < n^C$ for every n). A collection of functions $\{f_s\}$ where $f_s : \{0,1\}^\ell \to \{0,1\}^\ell$ for $\ell = \ell(|s|)$ is called a *pseudorandom permutation* (*PRP*) collection if:
  1. It is a pseudorandom function collection (i.e., the map, $s, x \mapsto f_s(x)$ is efficiently computatble and there is no efficient distinguisher between $f_s(\cdot)$ with a random s and a random function).
  2. Every function f_s is a permutation of $\{0,1\}^\ell$ (i.e., a one to one and onto map)
  3. There is an efficient algorithm that on input s, y returns $f_s^{-1}(y)$.

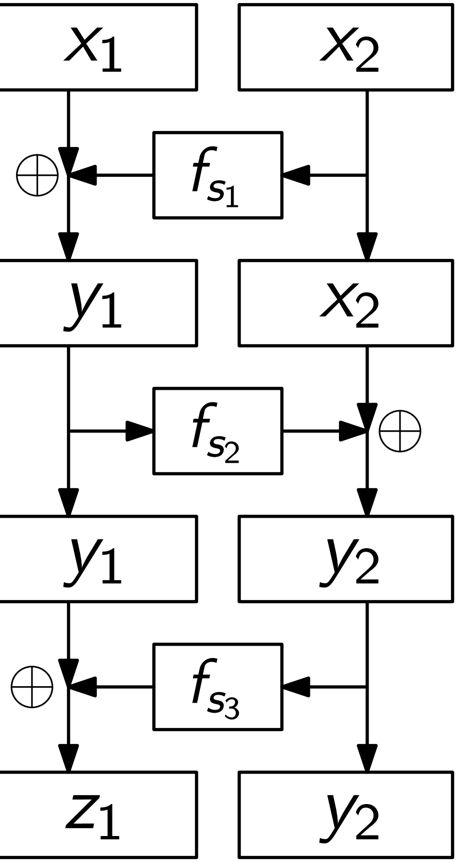
The parameter n is known as the *key length* of the PRP collection and the parameter $\ell = \ell(n)$ is known as the *input length* or *block length*. Often, $\ell = n$, and mostly we can safely ignore this distinction.

Construction of Pesudorandom Permutations (PRPs)

- **Theorem 5.5** (PRPs from PRFs)
  If the PRG Axiom is true, then there exists a PRP collection.

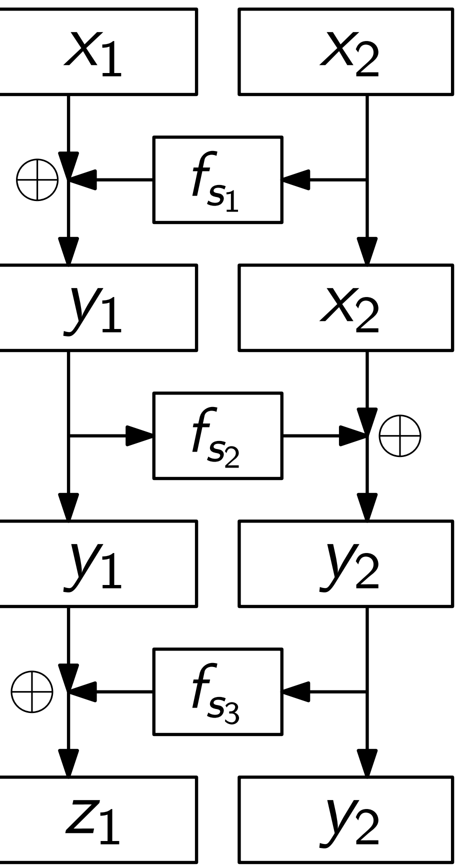
Construction of Pesudorandom Permutations (PRPs)

- **Theorem 5.5** (PRPs from PRFs)
  If the PRG Axiom is true, then there exists a PRP collection.



Construction of Pesudorandom Permutations (PRPs)

- **Theorem 5.5** (PRPs from PRFs)
  If the PRG Axiom is true, then there exists a PRP collection.



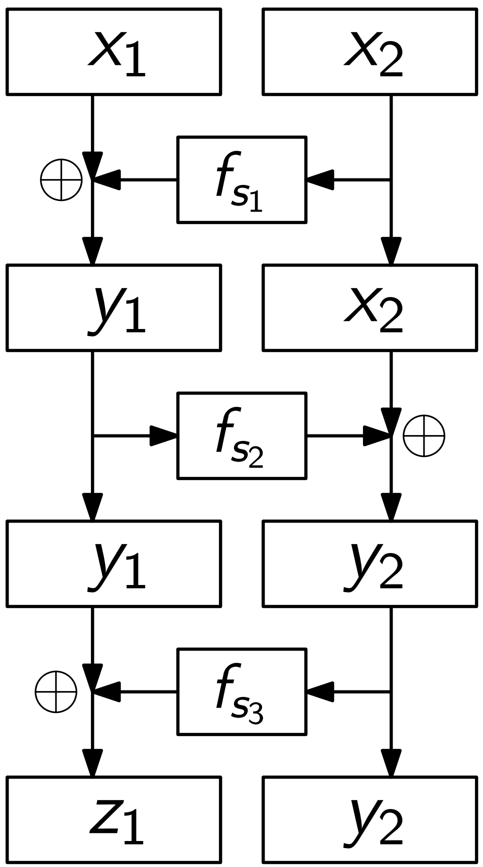
We build a PRP p on $2n$ bits from three PRFs $f_{s_1}, f_{s_2}, f_{s_3}$ on n bits by letting

$$p_{s_1,s_2,s_3}(x_1, x_2) = (z_1, y_2)$$

where $y_1 = x_1 \oplus f_{s_1}(x_2)$,
$y_2 = x_2 \oplus f_{s_2}(y_1)$, and
$z_1 = f_{s_3}(y_2) \oplus y_1$.

Construction of Pesudorandom Permutations (PRPs)

- **Theorem 5.5** (PRPs from PRFs)
  If the PRG Axiom is true, then there exists a PRP collection.



We build a PRP p on $2n$ bits from three PRFs $f_{s_1}, f_{s_2}, f_{s_3}$ on n bits by letting

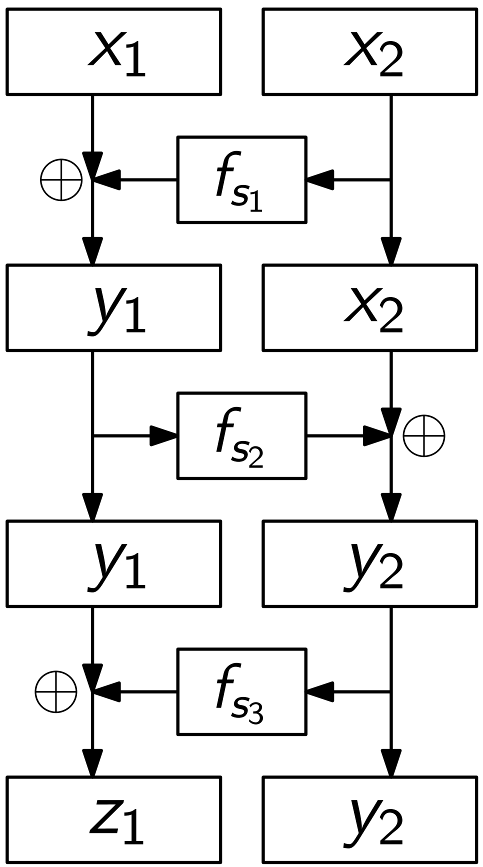
$$p_{s_1,s_2,s_3}(x_1, x_2) = (z_1, y_2)$$

where $y_1 = x_1 \oplus f_{s_1}(x_2)$,
$y_2 = x_2 \oplus f_{s_2}(y_1)$, and
$z_1 = f_{s_3}(y_2) \oplus y_1$.

This is so-called *Luby-Rackoff construction*, uses several rounds of *Feistel Transformation*.

Construction of Pesudorandom Permutations (PRPs)

- **Theorem 5.5** (PRPs from PRFs)
  If the PRG Axiom is true, then there exists a PRP collection.



We build a PRP p on $2n$ bits from three PRFs $f_{s_1}, f_{s_2}, f_{s_3}$ on n bits by letting

$$p_{s_1,s_2,s_3}(x_1, x_2) = (z_1, y_2)$$

where $y_1 = x_1 \oplus f_{s_1}(x_2)$, $y_2 = x_2 \oplus f_{s_2}(y_1)$, and $z_1 = f_{s_3}(y_2) \oplus y_1$.

For an overview of the proof, see Section 7.6 in Katz-Lindell.

Acknowledgement

- Some materials are extracted from the slides created by Prof. Qi Wang.