Cryptography
Lecture 6

Arkady Yerukhimovich

September 16, 2024

Outline

Lecture 5 Review

- Security of PRG+OTP
- Quiz on reductions

Outline

1. Lecture 5 Review
2. Quiz on Reductions
3. Review of PRG+OTP Proof
4. Chosen-Plaintext Attack (CPA) Security (Chapter 3.4.2)
5. Pseudorandom Function (PRF) (Chapter 3.5.1)

Problem 1

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG. Prove that

$$G'(s) = \overline{G(s)}$$

is a secure PRG

Problem 2

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme secure vs. eavesdropper. Prove that

$$\mathsf{Enc}'_k(m) = \overline{\mathsf{Enc}_k(m)}$$

is also secure

Problem 3

What would change if we defined $\mathsf{Enc}'_k(m) = \mathsf{Enc}_k(\overline{m})$

Some Notes on Pseudorandomness

What is a PRG (Informal)

PRG says the following two distributions are indistinguishable

- $s \leftarrow \{0,1\}^n$, output $G(s)$
- $r \leftarrow \{0,1\}^{l(n)}$

Some Notes on Pseudorandomness

What is a PRG (Informal)

PRG says the following two distributions are indistinguishable

- $s \leftarrow \{0,1\}^n$, output $G(s)$
- $r \leftarrow \{0,1\}^{l(n)}$

Observations:

- This does not mean that $G(s) = r$. Equality of distributions, not strings

Some Notes on Pseudorandomness

What is a PRG (Informal)

PRG says the following two distributions are indistinguishable

- $s \leftarrow \{0,1\}^n$, output $G(s)$
- $r \leftarrow \{0,1\}^{l(n)}$

Observations:

- This does not mean that $G(s) = r$. Equality of distributions, not strings
- In particular, incorrect to say string w is pseudorandom

Some Notes on Pseudorandomness

What is a PRG (Informal)

PRG says the following two distributions are indistinguishable

- $s \leftarrow \{0,1\}^n$, output $G(s)$
- $r \leftarrow \{0,1\}^{l(n)}$

Observations:

- This does not mean that $G(s) = r$. Equality of distributions, not strings
- In particular, incorrect to say string w is pseudorandom
- Indistinguishability only holds for PPT adversaries

Some Notes on Pseudorandomness

What is a PRG (Informal)

PRG says the following two distributions are indistinguishable

- $s \leftarrow \{0,1\}^n$, output $G(s)$
- $r \leftarrow \{0,1\}^{l(n)}$

Observations:

- This does not mean that $G(s) = r$. Equality of distributions, not strings
- In particular, incorrect to say string w is pseudorandom
- Indistinguishability only holds for PPT adversaries
- Most easily captured in a game

Outline

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k, c)$: $m = G(k) \oplus c$

Security of PRG+OTP: Intuition

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Security of PRG+OTP: Intuition

Assumption: $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ is PRG
Goal: Prove that $\Pi =$PRG+OTP is secure
Proof:

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG
Goal: Prove that $\Pi =$PRG+OTP is secure
Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] > 1/2 + 1/\text{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

Goal: Prove that Π =PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] > 1/2 + 1/\text{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:

Intuition

- $\mathcal{A}_r$ receives either $r \leftarrow \{0,1\}^{l(n)}$ or $r = G(s)$, uses r as mask

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG
Goal: Prove that $\Pi =$PRG+OTP is secure
Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π $(\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] > 1/2 + 1/\text{poly}(n))$
- Construct $\mathcal{A}_r$ that breaks G:

Intuition

- $\mathcal{A}_r$ receives either $r \leftarrow \{0,1\}^{l(n)}$ or $r = G(s)$, uses r as mask
- If $r \leftarrow \{0,1\}^{l(n)}$, Π is just OTP ($\Pr[\mathcal{A}_c$ WINS$] = 1/2$)

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Assumption: $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ is PRG
Goal: Prove that Π =PRG+OTP is secure
Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] > 1/2 + 1/\text{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:

Intuition

- $\mathcal{A}_r$ receives either $r \leftarrow \{0,1\}^{l(n)}$ or $r = G(s)$, uses r as mask
- If $r \leftarrow \{0,1\}^{l(n)}$, Π is just OTP ($\Pr[\mathcal{A}_c$ WINS$] = 1/2$)
- If $r = G(s)$, Π is PRG+OTP (by assumption, $\Pr[\mathcal{A}_c$ WINS$] > 1/2 + 1/\text{poly}(n)$)

Security of PRG+OTP: Intuition

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG
Goal: Prove that Π =PRG+OTP is secure
Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] > 1/2 + 1/\text{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:

Intuition

- $\mathcal{A}_r$ receives either $r \leftarrow \{0,1\}^{l(n)}$ or $r = G(s)$, uses r as mask
- If $r \leftarrow \{0,1\}^{l(n)}$, Π is just OTP ($\Pr[\mathcal{A}_c$ WINS$] = 1/2$)
- If $r = G(s)$, Π is PRG+OTP (by assumption, $\Pr[\mathcal{A}_c$ WINS$] > 1/2 + 1/\text{poly}(n)$)
- $\mathcal{A}_r$ runs $\mathcal{A}_c$ generating challenge c using r, observes if $\mathcal{A}_c$ wins, and if so outputs "PRG".

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b = 1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},n}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},n} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

Goal: Prove that $\Pi =$PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\mathrm{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\mathrm{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathrm{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathrm{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathrm{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathrm{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathrm{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

Goal: Prove that $\Pi =$PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π $(\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\mathrm{poly}(n))$
- Construct $\mathcal{A}_r$ that breaks G:
  - $\mathcal{A}_r$ gets $r \in \{0,1\}^{l(n)}$ as its challenge (trying to tell if its random or $G(s)$)

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ is PRG

Goal: Prove that $\Pi =$PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π
  $(\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\mathrm{poly}(n))$
- Construct $\mathcal{A}_r$ that breaks G:
  - $\mathcal{A}_r$ gets $r \in \{0,1\}^{l(n)}$ as its challenge (trying to tell if its random or $G(s)$)
  - $\mathcal{A}_r$ runs $\mathcal{A}_c$ to get (m_0, m_1)

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\text{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\text{Enc}(k,m)$: $c = G(k) \oplus m$
- $\text{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\text{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \text{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \text{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\text{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

Goal: Prove that $\Pi =$PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π $(\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\text{poly}(n))$
- Construct $\mathcal{A}_r$ that breaks G:
  - $\mathcal{A}_r$ gets $r \in \{0,1\}^{l(n)}$ as its challenge (trying to tell if its random or $G(s)$)
  - $\mathcal{A}_r$ runs $\mathcal{A}_c$ to get (m_0, m_1)
  - $\mathcal{A}_r$ chooses $b \leftarrow \{0,1\}$ and sets $c = r \oplus m_b$ (challenge)

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ is PRG

Goal: Prove that Π =PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π $(\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\mathrm{poly}(n))$
- Construct $\mathcal{A}_r$ that breaks G:
  - $\mathcal{A}_r$ gets $r \in \{0,1\}^{l(n)}$ as its challenge (trying to tell if its random or $G(s)$)
  - $\mathcal{A}_r$ runs $\mathcal{A}_c$ to get (m_0, m_1)
  - $\mathcal{A}_r$ chooses $b \leftarrow \{0,1\}$ and sets $c = r \oplus m_b$ (challenge)
  - $\mathcal{A}_r$ gives c to $\mathcal{A}_c$ and gets bit b'

Security of PRG+OTP: Building $\mathcal{A}_r$

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Assumption: $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ is PRG

Goal: Prove that $\Pi =$PRG+OTP is secure

Proof:

- Assume there exists PPT $\mathcal{A}_c$ that breaks Π ($\Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n)] > 1/2 + 1/\text{poly}(n)$)
- Construct $\mathcal{A}_r$ that breaks G:
  - $\mathcal{A}_r$ gets $r \in \{0,1\}^{l(n)}$ as its challenge (trying to tell if its random or $G(s)$)
  - $\mathcal{A}_r$ runs $\mathcal{A}_c$ to get (m_0, m_1)
  - $\mathcal{A}_r$ chooses $b \leftarrow \{0,1\}$ and sets $c = r \oplus m_b$ (challenge)
  - $\mathcal{A}_r$ gives c to $\mathcal{A}_c$ and gets bit b'
  - $\mathcal{A}_r$ outputs 1 ("PRG") if $b = b'$ and 0 otherwise

Security of PRG+OTP: Analysis

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b = 1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow$ Gen, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathrm{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r$ WINS$]$ ($\Pr[PRG_{\mathcal{A}_r,G}(n) = 1]$)

Security of PRG+OTP: Analysis

PRG+OTP Encryption

- $\mathrm{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathrm{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathrm{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathrm{PrivK}^{eav}_{\mathcal{A},n}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathrm{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathrm{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathrm{PrivK}^{eav}_{\mathcal{A},n} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r \text{ WINS}]$ $(\Pr[PRG_{\mathcal{A}_r,G}(n) = 1])$

- Case 1: $r \leftarrow \{0,1\}^{l(n)}$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r \leftarrow \{0,1\}^{l(n)}$, this is just OTP

Security of PRG+OTP: Analysis

PRG+OTP Encryption

- Gen(1^n): $k \leftarrow \{0,1\}^n$
- Enc(k, m): $c = G(k) \oplus m$
- Dec(k, c): $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b = 1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},n}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow$ Gen, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},n} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r$ WINS$]$ ($\Pr[PRG_{\mathcal{A}_r,G}(n) = 1]$)

- Case 1: $r \leftarrow \{0,1\}^{l(n)}$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r \leftarrow \{0,1\}^{l(n)}$, this is just OTP
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c$ outputs $b' = b] = 1/2$

Security of PRG+OTP: Analysis

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b = 1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},n}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},n} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r \text{ WINS}]$ ($\Pr[PRG_{\mathcal{A}_r,G}(n) = 1]$)

- Case 1: $r \leftarrow \{0,1\}^{l(n)}$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r \leftarrow \{0,1\}^{l(n)}$, this is just OTP
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c \text{ outputs } b' = b] = 1/2$
- Case 2: $r = G(s)$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r = G(s)$, this is OTP+PRG

Security of PRG+OTP: Analysis

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$;
  if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$.
  He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r \text{ WINS}]$ ($\Pr[PRG_{\mathcal{A}_r,G}(n) = 1]$)

- Case 1: $r \leftarrow \{0,1\}^{l(n)}$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r \leftarrow \{0,1\}^{l(n)}$, this is just OTP
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c \text{ outputs } b' = b] = 1/2$
- Case 2: $r = G(s)$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r = G(s)$, this is OTP+PRG
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c$ outputs $b' = b] = \Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] \geq 1/2 + 1/\mathrm{poly}(n)$

PRG+OTP Encryption

- $\mathsf{Gen}(1^n)$: $k \leftarrow \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: $c = G(k) \oplus m$
- $\mathsf{Dec}(k,c)$: $m = G(k) \oplus c$

$PRG_{\mathcal{D},G}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$. If $b=0$, he chooses $r \leftarrow \{0,1\}^{l(n)}$; if $b=1$, he chooses $s \leftarrow \{0,1\}^n$, and computes $r = G(s)$. He gives r to $\mathcal{D}$.
- On input r, the distinguisher $\mathcal{D}$ outputs a guess b'
- $PRG_{\mathcal{D},G}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

$\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$
- The challenger chooses $k \leftarrow \mathsf{Gen}$, $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{eav}_{\mathcal{A},\Pi} = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Need to analyze $\Pr[\mathcal{A}_r \text{ WINS}]$ ($\Pr[PRG_{\mathcal{A}_r,G}(n) = 1]$)

- Case 1: $r \leftarrow \{0,1\}^{l(n)}$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r \leftarrow \{0,1\}^{l(n)}$, this is just OTP
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c \text{ outputs } b' = b] = 1/2$
- Case 2: $r = G(s)$
  - $\mathcal{A}_c$ receives $c = r \oplus m_b$ with $r = G(s)$, this is OTP+PRG
  - $\Pr[\mathcal{A}_r(r) = 1] = \Pr[\mathcal{A}_c \text{ outputs } b' = b] = \Pr[PrivK^{eav}_{\mathcal{A}_c,\Pi}(1^n) = 1] \geq 1/2 + 1/\mathrm{poly}(n)$
- Summing these together, we get

$$\begin{aligned}\Pr[PRG_{\mathcal{A}_r,G}(1^n) = 1] &\geq 1/2 \cdot 1/2 + 1/2 \cdot (1/2 + 1/\mathrm{poly}(n)) \\ &= 1/2 + 1/(2\mathrm{poly}(n))\end{aligned}$$

Contradiction!

Outline

Where Are We Now

- Features of PRG+OTP encryption
  - Can encrypt messages of arbitrary length, just need PRG with enough stretch.
  - Achieve security against an eavesdropper

Where Are We Now

- Features of PRG+OTP encryption
  - Can encrypt messages of arbitrary length, just need PRG with enough stretch.
  - Achieve security against an eavesdropper
- Limitations of PRG+OTP encryption
  - Can only see one encryption
  - If see two, can tell whether they are equal

CPA Security Intuition

CPA Security

- $\mathcal{A}$ is allowed to request encryptions (under key k) of any messages of its choice.
- $\mathcal{A}$ still cannot learn any information about encrypted message when seeing challenge ciphertext c.

Why We Need CPA Security - A Historical Motivation

British Mines:

- British would bury a mine at specific latitude, longitude
- When Germans would find the mine, they would encrypt location and send back to HQ
- British intercepted these ciphertexts and used them to break security for German military comm's

Why We Need CPA Security - A Historical Motivation

British Mines:

- British would bury a mine at specific latitude, longitude
- When Germans would find the mine, they would encrypt location and send back to HQ
- British intercepted these ciphertexts and used them to break security for German military comm's

Battle of Midway:

- US forces intercepted and partially decrypted Japanese message
- Learned that Japan was going to attack location "AF" - wanted to confirm that this was Midway Island
- US sent out a message that "Midway is low on water" making sure Japanese intercepted it
- Japanese forces send message "AF is low on water" to HQ

Oracles in Cryptography

Mythology: An oracle is a person who knows answers to difficult questions.

Oracles in Cryptography

Mythology: An oracle is a person who knows answers to difficult questions.

Cryptography: An oracle evaluates a function without revealing its internal details.

Oracles in Cryptography

Mythology: An oracle is a person who knows answers to difficult questions.

Cryptography: An oracle evaluates a function without revealing its internal details.

- Example: Can give oracle access to $\mathrm{Enc}_k(\cdot)$. This allows caller to encrypt m of its choice without learning k.

Oracles in Cryptography

Mythology: An oracle is a person who knows answers to difficult questions.

Cryptography: An oracle evaluates a function without revealing its internal details.

- Example: Can give oracle access to $\mathsf{Enc}_k(\cdot)$. This allows caller to encrypt m of its choice without learning k.
- Notation:
  - We write $\mathcal{A}^{\mathcal{O}(\cdot)}$ to indicate a party $\mathcal{A}$ given oracle access to some function $\mathcal{O}$.
  - Calls to $\mathcal{O}$ cost 1 computation step
  - Oracles are a useful tool in security definitions and proofs

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Defining CPA-Secure Encryption

Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following game between an adversary $\mathcal{A}$ and a challenger:

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

Definition: An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ is CPA-secure if for all PPT $\mathcal{A}$ it holds that

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

Observations

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

- Adversary can query $\mathsf{Enc}_k(\cdot)$ oracle on any messages of her choice (even m_0 and m_1)

Observations

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

- Adversary can query $\mathsf{Enc}_k(\cdot)$ oracle on any messages of her choice (even m_0 and m_1)
- This captures her ability to get people to encrypt messages for her

Observations

$\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n)$

- The challenger chooses $k \leftarrow \mathsf{Gen}(1^n)$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}(1^n)$ outputs m_0, m_1 such that $|m_0| = |m_1|$.
- The challenger chooses $b \leftarrow \{0,1\}$, computes $c \leftarrow \mathsf{Enc}_k(m_b)$ and gives c to $\mathcal{A}$
- $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ outputs a guess bit b'
- We say that $\mathsf{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1$ (i.e., $\mathcal{A}$ wins) if $b' = b$.

- Adversary can query $\mathsf{Enc}_k(\cdot)$ oracle on any messages of her choice (even m_0 and m_1)
- This captures her ability to get people to encrypt messages for her
- She still cannot learn any information about encrypted message.

Benefits of CPA-Security

- Can encrypt many messages
  - $\mathcal{A}$ gets to see encryptions of many messages of its choice, still cannot break security of challenge
  - Can show that this means that seeing many ciphertexts doesn't help break any of them

Benefits of CPA-Security

- Can encrypt many messages
  - $\mathcal{A}$ gets to see encryptions of many messages of its choice, still cannot break security of challenge
  - Can show that this means that seeing many ciphertexts doesn't help break any of them
- Can encrypt arbitrarily long messages
  - Break message m into n-bit blocks, $m = m_1||m_2||\cdots||m_\ell$
  - To encrypt m separately encrypt each m_i.
  - Secure since this is just encrypting many messages

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

Key Idea

What if encryption (and decryption) could generate a different OTP for each ciphertext?

How to Construct CPA-Secure Encryption

- Recall that PRG+OTP encryption allowed us to encrypt long messages.
- But, it still revealed if same message was encrypted many times.

Key Idea

What if encryption (and decryption) could generate a different OTP for each ciphertext?

Note: We need to produce enough OTP's for as many encryptions as $\mathcal{A}$ wants. So, can't just pre-generate them all.

Outline

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	
0001	
⋮	
1111	

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	
0001	
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
$\vdots$	
1111	

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
⋮	⋮
1111	1101

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$

What Is a Random Function?

Consider a function $f : \{0,1\}^n \to \{0,1\}^n$

x	f(x)
0000	1010
0001	0001
$\vdots$	$\vdots$
1111	1101

Choosing a random function:

- Choose each value $f(x)$ independently and uniformly at random from $\{0,1\}^n$
- This is the same as choosing a uniformly random function from the set of all n-bit to n-bit functions

Why Random Functions Are Useful for Crypto

- Key feature

Why Random Functions Are Useful for Crypto

- Key feature
    - If haven't queried value of f at x, $f(x)$ is uniformly random.

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions
    - Writing down a random f from $\mathcal{F}_n$ requires $\log |\mathcal{F}_n| = n2^n$ bits

Why Random Functions Are Useful for Crypto

- Key feature
  - If haven't queried value of f at x, $f(x)$ is uniformly random.
  - Informally, this gives you 2^n OTPs
- Just one problem
  - Evaluating random functions is terribly inefficient
  - Can't even efficiently specify a random function
    - Each cell in function table has 2^n possibilities
    - There are 2^n cells
    - Thus, there are $|\mathcal{F}_n| = (2^n)^{(2^n)} = 2^{n2^n}$ possible functions
    - Writing down a random f from $\mathcal{F}_n$ requires $\log|\mathcal{F}_n| = n2^n$ bits

Question:

How can we get the benefits of a random function without paying the overhead?

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that:

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ such that:

- $F_k(\cdot)$ is efficiently computable

PRF Goals

Construct an efficient, keyed function
$F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ such that:

- $F_k(\cdot)$ is efficiently computable
- For a random key $k \leftarrow \{0,1\}^n$, $F_k(\cdot)$ looks like a *random function* from n bits to n bits (to someone who doesn't know k).

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

Definition: F is a secure PRF if for all PPT distinguishers $\mathcal{D}$, it holds that

$$\Pr[PRF_{\mathcal{D},F}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$$

PRF Definition

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a deterministic, keyed, poly-time function.

$PRF_{\mathcal{D},F}(n)$

- The challenger chooses $b \leftarrow \{0,1\}$.
  If $b = 0$, he chooses $f \leftarrow \mathcal{F}_n$ and gives $\mathcal{D}$ an oracle $\mathcal{O} = f$.
  if $b = 1$, he chooses $k \leftarrow \{0,1\}^n$, and gives $\mathcal{D}$ an oracle $\mathcal{O} = F_k$.
- With access to oracle $\mathcal{O}$, the distinguisher $\mathcal{D}$ outputs a bit b'
- $PRF_{\mathcal{D},F}(n) = 1$ (i.e., $\mathcal{D}$ wins) if $b' = b$

Definition: F is a secure PRF if for all PPT distinguishers $\mathcal{D}$, it holds that

$$\Pr[PRF_{\mathcal{D},F}(n) = 1] \leq 1/2 + \mathrm{negl}(n)$$

$\mathcal{D}$ cannot distinguish between oracle access to a random function and oracle access to a PRF (for a key k he doesn't know).

Observations

Observations:

- $\mathcal{D}$ can make polynomially many queries to $\mathcal{O}$
- $\mathcal{D}$ can choose its queries adaptively based on results of earlier queries
- The set of polynomially many evaluations of $F_k(\cdot)$ must look random
- Clearly, this is not possible if $\mathcal{D}$ knows k

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1
  - If $\mathcal{O} = f$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability $1/2^n$

An Example

Example: Is the following F a secure PRF?

$$F_k(x) = k \oplus x$$

Pseudorandomness:

- If k is random, $F_k(x) = k \oplus x$ is random when evaluated once
- But, consider $F_k(x_1) = k \oplus x_1$ and $F_k(x_2) = k \oplus x_2$:

$$F_k(x_1) \oplus F_k(x_2) = (k \oplus x_1) \oplus (k \oplus x_2) = (x_1 \oplus x_2)$$

- Given oracle $\mathcal{O}$ (either f or F_k), $\mathcal{D}$ evaluates $y_1 = \mathcal{O}(x_1)$ and $y_2 = \mathcal{O}(x_2)$ and outputs 1 (PRF) if $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ and 0 if not.
  - If $\mathcal{O} = F_k$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability 1
  - If $\mathcal{O} = f$, then $(y_1 \oplus y_2) = (x_1 \oplus x_2)$ with probability $1/2^n$
- So, $\mathcal{D}$ always outputs 1 when $\mathcal{O} = F_k$ and outputs 1 with probability $1/2^n$ when $\mathcal{O} = f$.

$$\Pr[\mathcal{D} \text{ WINS}] = \Pr[b = 1] \cdot 1 + \Pr[b = 0] \cdot (1 - 1/2^n) > 1/2$$

Variants of PRFs

- Pseudorandom permutation (PRP)

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.
  - In a *strong PRP*, we give $\mathcal{D}$ access to oracles for both f and f^{-1}. $\mathcal{D}$ still should not be able to distinguish from a PRP from a random permutation even using both oracles.

Variants of PRFs

- Pseudorandom permutation (PRP)
  - Recall that a *permutation* is a function that is one-to-one and onto with same domain and range (it shuffles the domain)
  - A PRP is a PRF where F_k is a permutation, and for security we compare to the case where f is a random permutation
- Strong PRP
  - Note that a permutation is always *invertible*. For every permutation f, there is a permutation f^{-1}.
  - In a *strong PRP*, we give $\mathcal{D}$ access to oracles for both f and f^{-1}. $\mathcal{D}$ still should not be able to distinguish from a PRP from a random permutation even using both oracles.
  - In applied crypto, this is often called a *blockcipher*.

Relationship Between PRG and PRF

Goals:

- Clearly, PRG and PRF have similar goals
- Both construct random-looking objects
- Both use this to "create randomness"

Relationship Between PRG and PRF

Goals:

- Clearly, PRG and PRF have similar goals
- Both construct random-looking objects
- Both use this to “create randomness”

Relationships:

- Not hard to show that a PRF can be used to build a PRG
- In fact, PRG can also be used to build a PRF
- But, important to remember the differences in functionalities and security definitions