Announcements/Reminders

Last day to submit HW1
HW2 will be posted today
• Due September 29

PR1 Due October 6
Previously on COS 433...
Defining Pseudorandom Generator (PRG)

Syntax:
• Seed space $S_\lambda$
• Output space $X_\lambda$
• $G: S_\lambda \rightarrow X_\lambda$ (deterministic)

Correctness:
• $|s| = \log |S_\lambda|$, $|x| = \log |X_\lambda|$ polynomial in $\lambda$,
• $|X_\lambda| > 2 \times |S_\lambda|$
• Running time of $G$ polynomial in $\lambda$
Security of PRGs

Definition: \( G : S_\lambda \rightarrow X_\lambda \) is a secure pseudorandom generator (PRG) if:

- For all running in polynomial time, \( \exists \) negl \( \varepsilon \),

\[
\Pr[\text{(}G(s)\text{)}=1:s \leftarrow S_\lambda] - \Pr[\text{(}x\text{)}=1:x \leftarrow X_\lambda] \leq \varepsilon(\lambda)
\]
Security

Assume towards contradiction that there is a non-negligible $\varepsilon$ such that

$$|\Pr[W_0] - \Pr[W_1]| \geq \varepsilon,$$

non-negligible $W_b$: $b' = 1$ in IND-Exp$_b$
Security

Use to build . will run as a subroutine, and pretend to be 

\[ m_0, m_1 \in M_\lambda \]

\[ b \leftarrow \{0,1\} \]

\[ c \leftarrow x \oplus m_b \]

\[ 1 \oplus b \oplus b' \]

(either \( G(s) \) or truly random)
Insecure: Linear Feedback Shift Registers

In each step,
• Last bit of state is removed and outputted
• Rest of bits are shifted right
• First bit is XOR of subset of remaining bits
PRGs should be Unpredictable

More generally, it should be hard, given some bits of output, to predict subsequent bits

Definition: \( G : S_\lambda \rightarrow \{0,1\}^{n(\lambda)} \) is unpredictable if, for all polynomial time and any \( p = p(\lambda) \), there exists negligible \( \varepsilon \) such that:

\[
\left| \Pr[G(s)_{p+1} \leftarrow (G(s)_{[1,p]})] - \frac{1}{2} \right| \leq \varepsilon(\lambda)
\]
Linearity

Problem: LFSR’s are linear

\[
\text{state}' = \begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \cdot \text{state (mod 2)}
\]

output = (0 0 0 0 0 1) \cdot \text{state (mod 2)}
LFSR period

Period = number of bits before state repeats

After one period, output sequence repeats

Therefore, should have extremely long period
• Ideally almost $2^\lambda$
• Possible to design LFSR’s with period $2^\lambda - 1$
Today: Constructing Software PRGs
Hardware vs Software

PRGs based on LFSR’s are very fast in hardware

Unfortunately, not easily amenable to software
RC4

Fast software based PRG

Resisted attack for several years

No longer considered secure, but still widely used
RC4

State = permutation on \([256]\) plus two integers
- Permutation stored as 256-byte array \(S\)

**Init(16-byte k):**
- For \(i=0,\ldots,255\)
  \[ S[i] = i \]
- \(j = 0\)
- For \(i=0,\ldots,255\)
  \[ j = j + S[i] + k[i \text{ mod } 16] \text{ (mod 256)} \]
  Swap \(S[i]\) and \(S[j]\)
- Output \((S,0,0)\)
RC4

\textbf{GetBits} (S, i, j):

- \( i++ \mod 256 \)
- \( j+= S[i] \mod 256 \)
- \text{Swap} \( S[i] \) and \( S[j] \)
- \( t = S[i] + S[j] \mod 256 \)
- Output \((S, i, j), S[t]\)

\text{New state} \quad \text{Next output byte}
Insecurity of RC4

Second byte of output is slightly biased towards 0
• $\Pr[\text{second byte} = 0^8] \approx \frac{2}{256}$
• Should be $\frac{1}{256}$

Means RC4 is not secure according to our definition
• outputs 1 iff second byte is equal to $0^8$
• Advantage: $\approx \frac{1}{256}$

Not a serious attack in practice, but demonstrates some structural weakness
Insecurity of RC4

Possible to extend attack to actually recover the input $k$ in some use cases

- The seed is set to $(IV, k)$ for some initial value $IV$
- Encrypt messages as $RC4(IV,k) \oplus m$
- Also give $IV$ to attacker
- Cannot show security assuming RC4 is a PRG

Can be used to completely break WEP encryption standard
PRGs Today

LFSRs and RC4 should not be used for cryptographic purposes, though RC4 still widely used

As course goes on, will see more PRGs
Length Extension for PRGs

Suppose I give you a PRG $G: \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+1}$

On its own, not very useful: can only compress keys by 1 bit

But, we can use it to build PRGs with arbitrarily-long outputs!
Extending the Stretch of a PRG
Security Proof

Assume towards contradiction \( \text{that breaks big PRG} \)

Goal: build adversary \( \text{that breaks } G \)
Problem?

\[ \text{seed} \]

\[ G \]

\[ \text{state} 1 \]

\[ \text{state} 2 \]

\[ \text{state} 3 \]

\[ \{0,1\}^{\lambda} \]

\[ \text{vs} \]

\[ \{0,1\}^{\lambda+1} \]

\[ \text{vs} \]

\[ \{0,1\} \]

\[ ? \]
Hybrid Arguments

Ubiquitous in crypto proofs

- distinguishes between two cases
  - Call them $H_0$ and $H_t$

Devise intermediate experiments $H_1, \ldots, H_{t-1}$ that “interpolate” between $H_0$ and $H_t$
  - Only change one thing at a time

Use triangle inequality to conclude that distinguishes $H_{i-1}$ and $H_i$
  - Use such a distinguisher to build
Proof by Hybrids

$H_0: \{0,1\}^\lambda$

Actual PRG evaluation
Security Proof

$H_t$: Truly Random Values

\{0,1\} \quad \{0,1\} \quad \{0,1\} \quad \{0,1\} \quad \ldots
Security Proof

\[ H_1 : \{0,1\}^\lambda \]
Security Proof

$H_2$: 

$\{0,1\} \rightarrow \{0,1\}^\lambda$ 

state$_2$ 

state$_3$ 

$\cdots$
Security Proof

$H_2$: 

\{0,1\} \quad \{0,1\} \quad \{0,1\} 

\begin{align*}
\text{state}_3
\end{align*} 

G 

\cdots
Security Proof

$H_+ :$

\begin{align*}
\{0,1\} & \quad \{0,1\} & \quad \{0,1\} & \quad \{0,1\} & \quad \cdots
\end{align*}
Security Proof

$H_0$ corresponds to pseudorandom $\mathbf{x}$

$H_t$ corresponds to truly random $\mathbf{x}$

Let $q_i = \Pr[\mathcal{H}_i(x) = 1 : x \leftarrow H_i]\$

By assumption, $|q_t - q_0| > \varepsilon$

Triangle ineq:

$$|q_t - q_0| \leq |q_1 - q_0| + |q_2 - q_1| + \ldots + |q_t - q_{t-1}|$$

$$\Rightarrow \exists i \text{ s.t. } |q_i - q_{i-1}| > \varepsilon/t$$
Security Proof

Analysis

• If \( y = G(s) \), then \( \text{sees } H_{i-1} \)
  \[ \Rightarrow \Pr[\text{outputs } 1] = q_{i-1} \]
  \[ \Rightarrow \Pr[\text{outputs } 1] = q_{i-1} \]

• If \( y \) is random, then \( \text{sees } H_i \)
  \[ \Rightarrow \Pr[\text{outputs } 1] = q_i \]
  \[ \Rightarrow \Pr[\text{outputs } 1] = q_i \]
Hybrids Recap

Useful whenever you can’t directly map between experiments

Only change one thing at a time, change corresponds to security of building block

• Not always obvious what hybrid sequence should be
Summary So Far

Stream ciphers = Encryption with PRG
- Secure encryption for arbitrary length, number of messages (though we did not completely prove it)

However, implementation difficulties due to having to maintain state
Multiple Message Security
Left-or-Right Experiment

$m_0, m_1 \in M$

$k \leftarrow K$

$c \leftarrow \text{Enc}(k,m_b)$

(Same $b$ for all queries)

$\text{LoR-Exp}_b(\text{Eve}, \lambda)$
LoR Security Definition

Definition: \((\text{Enc}, \text{Dec})\) has Left-or-Right indistinguishability if, for all \(\mathcal{A}\) running in polynomial time, \(\exists\) negligible \(\varepsilon\) such that:

\[ | \Pr[1\leftarrow\text{LoR-Exp}_0(\mathcal{A}, \lambda)] - \Pr[1\leftarrow\text{LoR-Exp}_1(\mathcal{A}, \lambda)] | \leq \varepsilon(\lambda) \]
Alternate Notion: CPA Security

What if adversary can additionally learn encryptions of messages of her choice?

Examples:
• Midway Island, WWII:
  • US cryptographers discover Japan is planning attack on a location referred to as “AF”
  • Guess that “AF” meant Midway Island
  • To confirm suspicion, sent message in clear that Midway Island was low on supplies
  • Japan intercepted, and sent message referencing “AF”
Alternate Notion: CPA Security

What if adversary can additionally learn encryptions of messages of her choice?

Examples:
• Mines, WWII:
  • Allies would lay mines at specific locations
  • Wait for Germans to discover mine
  • Germans would broadcast warning message about the mines, encrypted with Enigma
  • Would also send an “all clear” message once cleared
CPA Experiment

\[ m_0, m_1 \in M \]

\[ m \in M \]

\[ c \leftarrow Enc(k,m) \]

\[ k \leftarrow K \]

\[ c \leftarrow Enc(k,m_b) \]

\[ c \leftarrow Enc(k,m) \]

\[ \text{Challenger} \]

\[ b \]

\[ b' \]

\[ \text{CPA-Exp}_b(\text{\texttt{204A}}) \]
Definition: \((\text{Enc}, \text{Dec})\) is CPA Secure if, for all \(\mathcal{A}\) running in polynomial time, \(\exists\) negligible \(\varepsilon\) such that:

\[
\left| \Pr[1 \leftarrow \text{CPA-Exp}_0(\mathcal{A}, \lambda)] - \Pr[1 \leftarrow \text{CPA-Exp}_1(\mathcal{A}, \lambda)] \right| \leq \varepsilon(\lambda)
\]
Generalized CPA Experiment

**Queries in any order**

- $m \in M \Rightarrow c \leftarrow Enc(k, m)
- $m_0, m_1 \in M \Rightarrow c \leftarrow Enc(k, m)
- $m \in M \Rightarrow c \leftarrow Enc(k, m)$

**Challenger**

- $b \leftarrow b'$

**G CPA-Exp**

$GCPA-Exp_b(\lambda, \lambda)$
GCPA Security Definition

Definition: \((\text{Enc}, \text{Dec})\) is Generalized CPA Secure if, for all \(\text{running in polynomial time}, \exists \) negligible \(\varepsilon\) such that:

\[
\left| \Pr[1 \leftarrow \text{GCPA-Exp}_0(\cdot, \lambda)] - \Pr[1 \leftarrow \text{GCPA-Exp}_1(\cdot, \lambda)] \right| \leq \varepsilon(\lambda)
\]
Theorem:

Left-or-Right indistinguishability

⇔

CPA-security

⇔

Generalized CPA-security
Proof

Generalized CPA-security $\rightarrow$ CPA-security
• Trivial: any adversary in the CPA experiment is also an adversary for the generalized CPA experiment that just doesn’t take advantage of the ability to make multiple challenge/LoR queries
Proof

Left-or-Right $\rightarrow$ Generalized CPA
- Assume towards contradiction that we have an adversary for the generalized CPA experiment
- Construct an adversary that runs as a subroutine, and breaks the Left-or-Right indistinguishability
Pr[1←LoR-Exp_b(\text{Wall-E}, \lambda)] = Pr[1←GCPA-Exp_b(\text{Charlie Brown}, \lambda)]
\[ \Pr[1 \leftrightarrow \text{LoR-Exp}_b(m, \lambda)] = \Pr[1 \leftrightarrow \text{GCPA-Exp}_b(b, \lambda)] \]
Proof

Left-or-Right $\rightarrow$ Generalized CPA

\[
\Pr[1\leftarrow \text{LoR-Exp}_0(\text{Robot}, \lambda)]
- \Pr[1\leftarrow \text{LoR-Exp}_1(\text{Robot}, \lambda)]

= \Pr[1\leftarrow \text{GCPA-Exp}_0(\text{Robot}, \lambda)]
- \Pr[1\leftarrow \text{GCPA-Exp}_1(\text{Robot}, \lambda)]
= \varepsilon
\]
Proof

(regular) CPA $\rightarrow$ Left-or-Right

• Assume towards contradiction that we have an adversary for the **LoR Indistinguishability**

• Hybrids!
Hybrid $i$:

If at most $i$ queries so far,
\[ k \leftarrow K \]
\[ c \leftarrow \text{Enc}(k, m_0) \]

If more than $i$ queries so far,
\[ c \leftarrow \text{Enc}(k, m_1) \]
Proof

(regular) CPA $\rightarrow$ Left-or-Right

• Hybrid $0$ is identical to $\text{LoR-Exp}_1(\lambda)$

• Hybrid $q$ is identical to $\text{LoR-Exp}_0(\lambda)$

• We know that $\text{distinguishes Hybrid } q$ and Hybrid $0$ with advantage $\epsilon$

  $\Rightarrow \exists i \text{ s.t. } \text{distinguishes Hybrid } i$ and Hybrid $i-1$ with advantage $\epsilon/q$
Pr[1←CPA-Exp_b(\lambda)] = Pr[1← in Hybrid i-b]
Proof

(regular) CPA $\rightarrow$ Left-or-Right

\[
\Pr[1 \leftarrow \text{CPA-Exp}_0(\phantom{\text{Exp}_0}, \lambda) ] \\
- \Pr[1 \leftarrow \text{CPA-Exp}_1(\phantom{\text{Exp}_1}, \lambda) ] \geq \epsilon/q
\]

= \Pr[1 \leftarrow \text{in Hybrid i} ] \\
- \Pr[1 \leftarrow \text{in Hybrid i-1} ] \geq \epsilon/q
Equivalences

Theorem:

Left-or-Right indistinguishability ⇔ CPA-security ⇔ Generalized CPA-security

Therefore, you can use whichever notion you like best.

Next time: how to construct
Announcements/Reminders

Last day to submit HW1
HW2 will be posted today
• Due September 29

PR1 Due October 6