COS433/Math 473: Cryptography

Mark Zhandry
Princeton University
Spring 2020
Reminders

HW1 Due Feb 20\textsuperscript{th}
HW2 Due Feb 27\textsuperscript{th}

PR1 Due March 10\textsuperscript{th}
Previously on COS 433...
Theorem: No stateless \textit{randomized} encryption scheme can have perfect security for multiple messages.
Security Parameter $\lambda$

Additional input to system, dictates “security level”

Key, message, ciphertext size all polynomial in $\lambda$

Probability of adversary success is negligible in $\lambda$
Defining Encryption Again

**Syntax:**
- Key space $K_{\lambda}$
- Message space $M_{\lambda}$
- Ciphertext space $C_{\lambda}$
- $\text{Enc}: K_{\lambda} \times M_{\lambda} \rightarrow C_{\lambda}$ (potentially randomized)
- $\text{Dec}: K_{\lambda} \times C_{\lambda} \rightarrow M_{\lambda}$

**Correctness:**
- $|k|=\log|K_{\lambda}|$, $|m|=\log|M_{\lambda}|$, $|c|=\log|C_{\lambda}|$ polynomial in $\lambda$
- For all $\lambda$, $k \in K_{\lambda}$, $m \in M_{\lambda}$,
  $\Pr[ \Pr[\text{Dec}(k, \text{Enc}(k, m)) = m ] = 1$
Statistical Distance

Given two distributions $D_1$, $D_2$ over a set $X$, define

$$\Delta(D_1,D_2) = \frac{1}{2} \sum_x | \Pr[D_1=x] - \Pr[D_2=x] |$$

Observations:

$$0 \leq \Delta(D_1,D_2) \leq 1$$

$$\Delta(D_1,D_2) = 0 \iff D_1 \equiv D_2$$

$$\Delta(D_1,D_2) \leq \Delta(D_1,D_3) + \Delta(D_3,D_2)$$

($\Delta$ is a metric)
Another View of Statistical Distance

Theorem: $\Delta(D_1, D_2) \geq \varepsilon$ iff $\exists$ (potentially randomized) $A$ s.t.

$$|\Pr[A(D_1) = 1] - \Pr[A(D_2) = 1]| \geq \varepsilon$$

Terminology: for any $A$, $|\Pr[A(D_1) = 1] - \Pr[A(D_2) = 1]|$ is called the “advantage” of $A$ in distinguishing $D_1$ and $D_2$
Definition: A scheme \((\text{Enc}, \text{Dec})\) has statistical secrecy for \(d\) messages if \(\exists\) negligible \(\varepsilon\) such that \(\forall\) two sequences \((m_0^{(i)})_{i \in [d]}\) , \((m_1^{(i)})_{i \in [d]} \in M_{\lambda}^d\),

\[
\Delta \left[ \left( \text{Enc}(K_\lambda, m_0^{(i)}) \right)_{i \in [d]}, \ \left( \text{Enc}(K_\lambda, m_1^{(i)}) \right)_{i \in [d]} \right] < \varepsilon(\lambda)
\]

We will call such a scheme \(d\)-time statistically secure.
Limits of Statistical Security

**Theorem:** Suppose \((\text{Enc, Dec})\) has plaintext space \(M = \{0,1\}^n\) and key space \(K = \{0,1\}^t\). Moreover, assume it is \((d, 0.4999)\)-secure. Then:

\[ t \geq d \cdot n \]

In other words, the key must be at least as long as the total length of all messages encrypted.
Takeaway

If you don’t want to physically exchange keys frequently, you cannot obtain statistical security

So, now what?
Computational Security

We are ok if adversary takes a really long time

Only considered attack for adversaries that don’t take too long
Today: Continuation of Computational Security
Brute Force Attacks

Simply try every key until find right one

If keys have length $\lambda$, $2^\lambda$ is upper bound on attack

Applicable when easy to check if key is correct
• In case of perfect/statistical security, not possible
Crypto and P vs NP

What if P = NP?

From this point forward, almost all crypto we will see depends on computational assumptions
Defining Encryption Yet Again

Syntax:
• Key space $K_\lambda$
• Message space $M_\lambda$
• Ciphertext space $C_\lambda$
• $\text{Enc}: K_\lambda \times M_\lambda \rightarrow C_\lambda$ (potentially randomized)
• $\text{Dec}: K_\lambda \times C_\lambda \rightarrow M_\lambda$

Correctness:
• $|k| = |K_\lambda|$, $|m| = |M_\lambda|$, $|c| = |C_\lambda|$ polynomial in $\lambda$
• $\text{Enc, Dec}$ running time polynomial in $\lambda$
• For all $\lambda$, $k \in K_\lambda$, $m \in M_\lambda$,
  $\Pr[ \Pr[\text{Dec}(k, \text{Enc}(k,m)) = m ] = 1$
Defining Security

Consider an attacker as a probabilistic efficient algorithm

Attacker gets to choose the messages

All attacker has to do is distinguish them
Security Experiment/Game
(One-time setting)

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

\[ k \leftarrow K_\lambda \]

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

\[ \text{IND-Exp}_b(\text{Charlie, } \lambda) \]
Security Definition

(One-time setting, concrete)

Definition: \((\text{Enc}, \text{Dec})\) has \((t, \varepsilon)\)-ciphertext indistinguishability if, for all running in time at most \(t\)

\[
\left| \Pr[1 \leftarrow \text{IND-Exp}_0(\cdot)] - \Pr[1 \leftarrow \text{IND-Exp}_1(\cdot)] \right| \leq \varepsilon
\]
Security Definition

(One-time setting, asymptotic)

Definition: \((\text{Enc, Dec})\) has ciphertext indistinguishability if, for all \(\mathcal{A}\) running in polynomial time, \(\exists\) negligible \(\epsilon\) s.t.

\[
\left| \Pr[1 \leftarrow \text{IND-Exp}_0(\mathcal{A}, \lambda)] - \Pr[1 \leftarrow \text{IND-Exp}_1(\mathcal{A}, \lambda)] \right| \leq \epsilon(\lambda)
\]
Construction with $|k| << |m|$

Idea: use OTP, but have key generated by some expanding procedure $G$

What do we want out of $G$?
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 \( \mathcal{A} \) running in polynomial time, \( \exists \) \( \text{negl} \varepsilon \),

\[
\left| \Pr[ \mathcal{A}(G(s))=1 : s \leftarrow S_{\lambda}] - \Pr[ \mathcal{A}(x)=1 : x \leftarrow X_{\lambda}] \right|\leq \varepsilon(\lambda)
\]
Secure PRG $\rightarrow$ Ciphertext Indistinguishability

\[ K_\lambda = S_\lambda \]
\[ M_\lambda = X_\lambda \text{ (assumed to be } \{0,1\}^n) \]
\[ C_\lambda = X_\lambda \]
\[ \text{Enc}(k,m) = \text{PRG}(k) \oplus m \]
\[ \text{Dec}(k,c) = \text{PRG}(k) \oplus c \]
Security?

Intuitively, security is obvious:
• $\text{PRG}(k)$ "looks" random, so should completely hide $m$
• However, formalizing this argument is non-trivial.

Solution: reductions
• Assume toward contradiction an adversary for the encryption scheme, derive an adversary for the PRG
Security

Assume towards contradiction that there is a \( m_0, m_1 \in M_\lambda \) and 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)
Security

Case 1: $x = \text{PRG}(s)$ for a random seed $s$

- “sees” $\text{IND-Exp}_b$ for a random bit $b$

$m_0, m_1 \in M_\lambda$

$b \leftarrow \{0,1\}$

$s \leftarrow S_\lambda$

$c \leftarrow \text{PRG}(s) \oplus m_b$

$b'$
Security

Case 1: \( x = \text{PRG}(s) \) for a random seed \( s \)

- “sees” \( \text{IND-Exp}_b \) for a random bit \( b \)

- \( \Pr[1 \oplus b \oplus b' = 1] = \Pr[b = b'] \)

  \[
  = \frac{1}{2} \Pr[b' = 1 \mid b = 1] \\
  + \frac{1}{2} (1 - \Pr[b' = 1 \mid b = 0]) \\
  = \frac{1}{2} (1 + \Pr[W_0] - \Pr[W_1]) \\
  = \frac{1}{2} (1 \pm \varepsilon)
  \]
Security

Case 2: \( \mathbf{x} \) is truly random

- “sees” OTP encryption

\[
\begin{align*}
    m_0, m_1 &\in M_\lambda \\
    b &\leftarrow \{0,1\} \\
    x &\leftarrow M_\lambda \\
    c &\leftarrow x \oplus m_b \\
    b' &\leftarrow .
\end{align*}
\]
Security

Case 2: \( \times \) is truly random

- “sees” OTP encryption
- Therefore \( \Pr[b'=1 \mid b=0] = \Pr[b'=1 \mid b=1] \)
- \( \Pr[1 \oplus b \oplus b'=1] = \Pr[b=b'] \)
  \[
  = \frac{1}{2} \Pr[b'=1 \mid b=1] \\
  + \frac{1}{2} (1 - \Pr[b'=1 \mid b=0])
  \\
  = \frac{1}{2}
  \]
Security

Putting it together:

- \( \Pr[(G(s))=1:s\leftarrow\{0,1\}^\lambda] = \frac{1}{2}(1 \pm \varepsilon(\lambda)) \)

- \( \Pr[(x)=1:x\leftarrow\{0,1\}^n] = \frac{1}{2} \)

- Absolute Difference: \( \frac{1}{2}\varepsilon, \Rightarrow \text{Contradiction!} \)
Security

**Thm:** If $G$ is a secure PRG, then $(Enc, Dec)$ is has ciphertext indistinguishability
An Alternate Proof: Hybrids

Idea: define sequence of “hybrid” experiments “between” \textsf{IND-Exp}_0 and \textsf{IND-Exp}_1

In each hybrid, make small change from previous hybrid

Hopefully, each small change is undetectable

Using triangle inequality, overall change from \textsf{IND-Exp}_0 and \textsf{IND-Exp}_1 is undetectable
An Alternate Proof: Hybrids

**Hybrid 0: IND-Exp₀**

\[ \begin{align*}
  & m₀, m₁ \in M_λ \\
  & c \\
  & b' \\
  \end{align*} \]

\[ \begin{align*}
  & k \leftarrow S_λ \\
  & c \leftarrow G(k) \oplus m₀ \\
  \end{align*} \]
An Alternate Proof: Hybrids

Hybrid 1:

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

\[ x \leftarrow M_\lambda \]

\[ c \leftarrow x \oplus m_0 \]

\[ b' \]
An Alternate Proof: Hybrids

Hybrid 2:

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

\[ x \leftarrow M_\lambda \]

\[ c \leftarrow x \oplus m_1 \]

\[ b' \]
An Alternate Proof: Hybrids

Hybrid 3: IND-Exp₁

\[ \begin{align*} m_0, m_1 & \in M_\lambda \\ k & \leftarrow S_\lambda \\ c & \leftarrow G(k) \oplus m_1 \end{align*} \]
An Alternate Proof: Hybrids

\[ | \Pr[b' = 1 : \text{IND-Exp}_0] - \Pr[b' = 1 : \text{IND-Exp}_1] | \]
\[ = | \Pr[b' = 1 : \text{Hyb 0}] - \Pr[b' = 1 : \text{Hyb 3}] | \]
\[ \leq | \Pr[b' = 1 : \text{Hyb 0}] - \Pr[b' = 1 : \text{Hyb 1}] | \]
\[ + | \Pr[b' = 1 : \text{Hyb 1}] - \Pr[b' = 1 : \text{Hyb 2}] | \]
\[ + | \Pr[b' = 1 : \text{Hyb 2}] - \Pr[b' = 1 : \text{Hyb 3}] | \]

If \(|\Pr[b' = 1 : \text{IND-Exp}_0] - \Pr[b' = 1 : \text{IND-Exp}_1]| \geq \varepsilon\),
Then for some \(i = 0, 1, 2\),
\(|\Pr[b' = 1 : \text{Hyb } i] - \Pr[b' = 1 : \text{Hyb } i+1]| \geq \varepsilon/3\)
An Alternate Proof: Hybrids

Suppose \( \text{ distinguishes } \text{Hybrid 0 from Hybrid 1} \) with advantage \( \frac{\epsilon}{3} \)

\[
\begin{align*}
  k \leftarrow S_{\lambda} \\
  m_0, m_1 \in M_{\lambda} \\
  c \leftarrow G(k) \oplus m_0 \\
  b' \\

  x \leftarrow M_{\lambda} \\
  m_0, m_1 \in M_{\lambda} \\
  c \leftarrow x \oplus m_0 \\
  b'
\end{align*}
\]
An Alternate Proof: Hybrids

Suppose \( \text{dis} \) distinguishes Hybrid 0 from Hybrid 1 with advantage \( \varepsilon/3 \) \( \Rightarrow \) Construct

\[
\begin{align*}
&m_0, m_1 \in M_{\lambda} \\
&x \leftarrow x \oplus m_0
\end{align*}
\]

(either \( G(s) \) or truly random)
An Alternate Proof: Hybrids

Suppose an adversary distinguishes Hybrid 0 from Hybrid 1 with advantage $\frac{\epsilon}{3} \Rightarrow \text{Construct}$

If is given $G(s)$ for a random $s$, sees Hybrid 0

If is given $x$ for a random $x$, sees Hybrid 1

Therefore, advantage of is equal to advantage of which is at least $\frac{\epsilon}{3} \Rightarrow \text{Contradiction!}$
An Alternate Proof: Hybrids

Suppose \( \mathcal{A} \) distinguishes Hybrid 1 from Hybrid 2 with advantage \( \varepsilon/3 \)

\[
\begin{align*}
x & \leftarrow M_\lambda \\
m_0, m_1 \in M_\lambda & \quad c \leftarrow x \oplus m_0 \\
& \quad b' \\
m_0, m_1 \in M_\lambda & \quad c \leftarrow x \oplus m_1 \\
& \quad b'
\end{align*}
\]
An Alternate Proof: Hybrids

Suppose distinguishes Hybrid 1 from Hybrid 2 with advantage $\epsilon(\lambda)/3$.

Impossible by OTP security.
Suppose \( \text{distinguishes Hybrid 2 from Hybrid 3} \) with advantage \( \varepsilon/3 \).
How do we build PRGs?
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
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
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
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
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
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
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
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
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

\[
\begin{array}{cccccc}
0 & 0 & 0 & 1 & 0 & \ldots
\end{array}
\]
Linear Feedback Shift Registers

Are LFSR’s secure PRGs?
Linear Feedback Shift Registers

Are LFSRs secure PRGs?

No!

First $n$ bits of output = initial state

Write $x = x_1, \ldots, x_n$, $x'$

Initialize LFSB to have state $x_1, \ldots, x_n$

Run LFSB for $|x|$ steps, obtaining $y$

Check if $y = x$
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 $p=p(\lambda)$, $\exists$ negligible $\varepsilon$ such that:

$$\left| \Pr[G(s)_{p+1} \leftrightarrow (G(s)_{[1,p]})] - \frac{1}{2} \right| \leq \varepsilon(\lambda)$$
PRGs should be Unpredictable

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

**Theorem:** $G$ is unpredictable iff it is pseudorandom
Proof

Pseudorandomness $\rightarrow$ Unpredictability

Assume towards contradiction $s.t.$

$$\left| \Pr[G(s)_{p+1} \leftarrow (G(s)_{[1,p]}) ] - \frac{1}{2} \right| > \varepsilon$$
Proof

Pseudorandomness $\rightarrow$ Unpredictability

Construct

\[ x_{[1, p]} \]

$\oplus$ b

$\oplus$ $x_{p+1}$
Proof

Pseudorandomness $\rightarrow$ Unpredictability

Analysis:

- If $x$ is random, $\Pr[1 \oplus b \oplus x_{p+1} = 1] = \frac{1}{2}$
- If $x$ is pseudorandom,
  \[
  \Pr[1 \oplus b \oplus x_{p+1} = 1] = \Pr[G(s)_{p+1} \leftarrow (G(s)_{[1,p]})] > (\frac{1}{2} + \varepsilon) \text{ or } < (\frac{1}{2} - \varepsilon)
  \]
Proof

Unpredictability $\implies$ Pseudorandomness

Assume towards contradiction $\exists$ s.t.

$$\left| \Pr[ (G(s))=1 : s \leftarrow \{0,1\}^\lambda ] - \Pr[ (x)=1 : x \leftarrow \{0,1\}^t ] \right| > \epsilon$$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Hybrids:
$H_i$: $x_{[1,i]} \leftarrow G(s)$, $x_{[i+1,t]} \leftarrow \{0,1\}^{t-i}$

$H_0$: truly random $\mathbf{x}$
$H_t$: pseudorandom $\mathbf{t}$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Hybrids:
$H_i: x_{[1,i]} \leftarrow G(s), x_{[i+1,t]} \leftarrow \{0,1\}^{t-i}$

$\Pr[O(x)=1: x\leftarrow H_s] - \Pr[O(x)=1: x\leftarrow H_0] > \epsilon$

Let $q_i = \Pr[O(x)=1: x\leftarrow H_i]$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Hybrids:
$H_i$: $x_{[1,i]} \leftarrow G(s)$, $x_{[i+1,t]} \leftarrow \{0,1\}^{t-i}$

$\left| q_t - q_0 \right| > \varepsilon$

Let $q_i = \Pr[\text{Q}(x)=1:x \leftarrow H_i]$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Hybrids:
$H_i: \ x_{[1,i]} \leftarrow G(s), \ x_{[i+1,t]} \leftarrow \{0,1\}^{t-i}$

By triangle inequality, there must exist an $i$ s.t.

$$|q_i - q_{i-1}| > \varepsilon/t$$

Can assume wlog that

$$q_i - q_{i-1} > \varepsilon/t$$
Proof

Unpredictability \(\rightarrow\) Pseudorandomness

Construct

\[
y = G(s)_{[1,i-1]} \\
b \leftarrow \{0,1\} \\
y' \leftarrow \{0,1\}^{t-i} \\
x = y||b||y' \\
b' \\
1 \oplus b \oplus b'
\]
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Analysis:
• If $b = G(s)_i$, then $\$ sees $H_i$

  $\Rightarrow$ outputs 1 with probability $q_i$

  $\Rightarrow$ outputs $b = G(s)_i$ with probability $q_i$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Analysis:
• If $b = 1 \oplus G(s)_i$, then
  
  Define $q'_i$ as $\Pr[\text{outputs 1}]$
  
  $\frac{1}{2}(q'_i + q_i) = q_{i-1} \Rightarrow q'_i = 2q_{i-1} - q_i$
  
  $\Rightarrow$ outputs $G(s)_{[1,i]}$ with probability
  
  $1 - q'_i = 1 + q_i - 2q_{i-1}$
Proof

Unpredictability $\rightarrow$ Pseudorandomness

Analysis:

- $\Pr[\text{outputs } G(s)_i]$

  $= \frac{1}{2} (q_i) + \frac{1}{2} (1 + q_i - 2q_{i-1})$

  $= \frac{1}{2} + q_i - q_{i-1}$

  $> \frac{1}{2} + \varepsilon/t$
Any ideas?
Linearity

LFSR’s are linear:

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\end{bmatrix} \cdot \text{state}
\]

\[
\text{output} = (0 \ 0 \ 0 \ 0 \ 0 \ 1) \cdot \text{state}
\]
Linearity

LFSR’s are linear:
- Each output bit is a linear function of the initial state (that is, $G(s) = A \cdot s \pmod{2}$)

Any linear $G$ cannot be a PRG
- Can check if $x$ is in column-span of $A$ using linear algebra
Introducing Non-linearity

Non-linearity in the output:

Non-linear feedback:
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 $2^\lambda - 1$
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 $i=0,...,255$
  \[ S[i] = i \]
• $j = 0$
• For $i=0,...,255$
  \[ j = j + S[i] + k[i \mod 16] \mod 256 \]
  Swap $S[i]$ and $S[j]$
• Output $(S,0,0)$
RC4

GetBits(S,i,j):

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

New state    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
Summary

Stream ciphers = secure encryption for arbitrary length, number of messages
(though we did not completely prove it)

However, implementation difficulties due to having to maintaining state
Reminders

HW1 Due Feb 20th
HW2 Due Feb 27th

PR1 Due March 10th