Announcements

Homework 2 posted, due Feb 21
Last Time

Multiple message security

Statistical Secrecy

Unfortunately, for everything we’ve looked at so far,
\[ |\text{key}| \geq |\text{total information encrypted}| \]
Today

$|\text{key}| \geq |\text{total information encrypted}|$ is necessary for statistical security

Computational Security

Stream Ciphers and PRGs
Notation

For a probabilistic algorithm $\mathbf{A}$, we write $\mathbf{A}(x; r)$ to denote running $\mathbf{A}$ on input $x$, using randomness $r$

- When thought of as a function of its input and randomness, $\mathbf{A}$ is deterministic
Statistical Secrecy

**Definition:** A scheme \((\text{Enc}, \text{Dec})\) has statistical secrecy for \(n\) messages if \(\exists\) negligible function \(\epsilon\) s.t. \(\forall\) two sequences of messages \((m_0^{(i)})_{i \in [n]}, (m_1^{(i)})_{i \in [n]} \in \mathcal{M}^n\)

\[
\Delta\left[\left(\text{Enc}(K_\lambda, m_0^{(i)})\right)_{i \in [n]}, \left(\text{Enc}(K_\lambda, m_1^{(i)})\right)_{i \in [n]}\right] < \epsilon(\lambda)
\]
Theorem: Suppose \((\text{Enc}, \text{Dec})\) has plaintext space \(M_\lambda = \{0,1\}^{n(\lambda)}\) and key space \(K_\lambda = \{0,1\}^{t(\lambda)}\). Moreover, assume it is statistically secure for \(d\) messages. Then:

\[ t(\lambda) \geq d \cdot n(\lambda) \]

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

Use an encryption protocol to build a compression protocol

\[ \text{Goal: } |m'| < |m| \]
For Now: Easier Goal

\[ m' \leftarrow \text{Comp}(s,m) \]

\[ s \leftarrow \text{Setup()} \]

\[ m \leftarrow \text{Decomp}(s, m') \]

Goal: \(|m'| < |m|\)
The Protocol

Let $m_0$ be some message in $M_\lambda$

Setup():
- Choose random $k_0 \leftarrow K$
- Let $c_1 \leftarrow \text{Enc}(k_0, m_0)$, ..., $c_d \leftarrow \text{Enc}(k_0, m_0)$
- Output $(c_1, \ldots, c_d)$

Comp( $(c_1, \ldots, c_d)$, $(m_1, \ldots, m_d)$ ):
- Find $k, r_1, \ldots, r_d$ such that $c_i = \text{Enc}(k, m_i; r_i)$ $\forall i$
- If no such values exist, abort
- Output $k$
The Protocol

Let $m_0$ be some message in $M_{\lambda}$

$Com\overline{p}( (c_1,...,c_d), (m_1,...,m_d) )$:
• Find $k,r_1,...,r_d$ such that $c_i=Enc(k,m_i; r_i) \ \forall i$
• If no such values exist, abort
• Output $k$

$Decomp((c_1,...,c_d), k )$:
• Compute $m_i = Dec(k,c_i)$
• Output $(m_1,...,m_d)$
Analysis of Protocol

If $\textbf{Comp}$ succeeds, $\textbf{Decomp}$ must succeed by correctness

• Since $c_i = \text{Enc}(k,m_i; r_i)$, $\text{Dec}(k,c_i)$ must give $m_i$

Therefore, must figure out when $\textbf{Comp}$ succeeds

**Claim:** There exists a negligible function $\varepsilon(\lambda)$ such that, for any sequence of messages $m_1, \ldots, m_d$, $\textbf{Comp}$ succeeds with probability at least $1 - \varepsilon(\lambda)$

(Probability over the randomness used by $\text{Setup()}$)
Claim: There exists a negligible function $\varepsilon(\lambda)$ such that, for any sequence of messages $m_1, \ldots, m_d$, $\text{Comp}$ succeeds with probability at least $1 - \varepsilon(\lambda)$

Proof:
- Suppose $\text{Comp}$ succeeds with probability $1 - p$ for messages $m_1, \ldots, m_d$
- Let $A(c_1, \ldots, c_d)$ be the algorithm that runs $\text{Comp}((c_1, \ldots, c_d), (m_1, \ldots, m_d))$ and outputs 1 if $\text{Comp}$ succeeds

- If $c_i = \text{Enc}(k_0, m_i)$, then $\Pr[A(c_1, \ldots, c_d) = 1] = 1$
- If $c_i = \text{Enc}(k_0, m_0)$, then $\Pr[A(c_1, \ldots, c_d) = 1] = 1 - p$
- By statistical security of $\text{Enc}$, $p$ must be negligible
Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for any sequence of messages $m_1, \ldots, m_d$, Comp succeeds with probability at least $1 - \epsilon(\lambda)$.

Claim: There exists a negligible function $\epsilon(\lambda)$ such that, for a random sequence of messages $m_1, \ldots, m_d$, Comp succeeds with probability at least $1 - \epsilon(\lambda)$.

(Probability over the randomness used by Setup() and the random choices of $m_1, \ldots, m_d$.)
Next step: Removing Setup

We know:

\[
\Pr[\text{Comp succeeds: } (c_1, \ldots, c_d) \leftarrow \text{Setup}(), \quad m_i \leftarrow M_\lambda] \geq 1 - \epsilon(\lambda)
\]

Therefore, there must exist some \((c_1^*, \ldots, c_d^*)\) such that

\[
\Pr[\text{Comp succeeds: } m_i \leftarrow M_\lambda] \geq 1 - \epsilon(\lambda)
\]

Define: \(M_\lambda' = \{(m_1, \ldots, m_d) : \text{Comp succeeds}\}\)

* Note that \(|M_\lambda'| \geq (1 - \epsilon(\lambda)) \cdot |M_\lambda|^d*
The Protocol

Find $k, r_1, \ldots, r_d$ such that $c_i^* = \text{Enc}(k, m_i; r_i)$ $\forall i$

For each $i$,
Let $m_i \leftarrow \text{Dec}(k, c_i^*)$
Output $(m_1, \ldots, m_d)$

By previous analysis,
• Alice always successfully compresses
• Bob always successfully decompresses
Final Touches

Can compress messages in $M'_\lambda$ into keys in $K_\lambda$

Therefore, it must be that $|M'_\lambda| \leq |K_\lambda|$

Meaning $t = \log |K_\lambda|$
\[
\geq \log |M'_\lambda| \\
\geq \log \left[ (1-\varepsilon(\lambda)) |M_\lambda|^{d} \right] \\
= d \log |M_\lambda| + \log \left[ 1-\varepsilon(\lambda) \right] \\
\geq dn - 2\varepsilon(\lambda) \\
\geq dn \text{ (as long as } \varepsilon(\lambda) < \frac{1}{2})
Takeaway

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

So, now what?
Computational Security

If it takes a billion years to decrypt the message, that’s ok

How long is okay?
• Practitioner: $2^{80}$, $2^{128}$, or maybe $2^{256}$ computational steps
• Theorist?
Defining Security

Consider an attacker as a probabilistic polynomial time algorithm (Turing machine)
• Cobham's thesis: captures anything computable in the real world

Attacker gets to choose the messages

All attacker has to do is distinguish them

Our impossibility gives an exponential-time algorithm → attack doesn’t apply in this setting
Defining Security

Efficiency of algorithms

• Doesn’t make sense to allow \((\text{Enc}, \text{Dec})\) to run in superpoly time, but not the adversary
• Therefore, all algorithms are also poly time (efficient)
Security Experiment/Game

\[ \lambda \]
\[ b \]

Challenger

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

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

\[ b' \]

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

Definition: \((\text{Enc}, \text{Dec})\) has ciphertext indistinguishability if, for all probabilistic polynomial time (PPT) , there exists a negligible function \(\varepsilon\) such that

\[
\left| \Pr[1 \leftarrow \text{IND-Exp}_0(\cdot, \lambda)] - \Pr[1 \leftarrow \text{IND-Exp}_1(\cdot, \lambda)] \right| \leq \varepsilon(\lambda)
\]
Construction with $|k| \ll |m|$

Idea: use OTP, but have key generated by some expanding function $G$
What Do We Want Out of $G$?

Definition: $G: \{0,1\}^* \rightarrow \{0,1\}^*$ is a secure pseudorandom generator (PRG) if:
- $G$ is computable in polynomial time
- $G$ applied to $\lambda$ bit strings produces strings of length $s(\lambda) > \lambda$
- $G$ is deterministic
- For all PPT $\mathcal{A}$, $\exists$ negl $\epsilon$ such that:

$$\left| \Pr[ \mathcal{A}(G(s))=1:s \leftarrow \{0,1\}^\lambda ] - \Pr[ \mathcal{A}(x)=1:x \leftarrow \{0,1\}^{s(\lambda)} ] \right| \leq \epsilon(\lambda)$$
Secure PRG → Ciphertext Indistinguishability

\[ K_\lambda = \{0,1\}^\lambda \]
\[ M_\lambda = \{0,1\}^{s(\lambda)} \]
\[ C_\lambda = \{0,1\}^{s(\lambda)} \]

\[ \text{Enc}(k,m) = \text{PRG}(k) \oplus m \]
\[ \text{Dec}(k,c) = \text{PRG}(k) \oplus c \]
Security?

Intuitively, security is obvious:
• 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 PPT such that

\[ |\Pr[W_0] - \Pr[W_1]| \geq \varepsilon(\lambda), \text{ non-negligible} \]

\( W_b: b' = 1 \) in IND-Exp_b
Security

Use \( m_0, m_1 \in \mathcal{M}_\lambda \) to build \( \cdot \). \( \cdot \) will run \( \cdot \) as a subroutine, and pretend to be \( \cdot \).

\[ \begin{align*}
  b & \leftarrow \{0, 1\} \\
  c & \leftarrow x \oplus m_b \\
  1 \oplus b \oplus b' & \leftarrow \text{(either } G(s) \text{ or truly random)}
\end{align*} \]
Security

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

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

$\begin{align*}
\text{m}_0, \text{m}_1 &\in M_\lambda \\
b &\leftarrow \{0,1\} \\
s &\leftarrow K_\lambda \\
c &\leftarrow \text{PRG}(s) \oplus \text{m}_b \\
b' &\leftarrow \text{IND-Exp}_b(b) 
\end{align*}$
Security

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

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

- $\Pr[1 \oplus b \oplus b' = 1] = \Pr[b = b']$

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

Case 2: $\mathbf{x}$ is truly random

- $\mathbf{x}$ “sees” OTP encryption

$m_0, m_1 \in \mathcal{M}_\lambda$  

\[
\begin{align*}
&b \leftarrow \{0,1\} \\
&x \leftarrow \{0,1\}^{s(\lambda)} \\
&c \leftarrow x \oplus m_b \\
&b' \downarrow
\end{align*}
\]
Security

Case 2: $\times$ is truly random

• 🪤 “sees” OTP encryption

• Therefore $\Pr[b'=1 \mid b=0] = \Pr[b'=1 \mid b=1]$

• $\Pr[1\oplus b \oplus b'=1] = \Pr[b=b']$

\[
= \frac{1}{2} \Pr[b'=1 \mid b=1 ]
\]

\[
+ \frac{1}{2} \left( 1 - \Pr[b'=1 \mid b=0]\right)
\]

\[
= \frac{1}{2}
\]
Security

Putting it together:

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

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

- Absolute Difference: $\frac{1}{2}\varepsilon(\lambda)$, non-negligible
  $\Rightarrow$ Contradiction!
An Alternate Proof: Hybrids

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

In each hybrid, make small change from previous hybrid

Hopefully, each small change is undetectable

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

Hybrid 0: \( \text{IND-Exp}_0 \)

\[
\begin{align*}
\text{Hybrid 0: } & \text{IND-Exp}_0 \\
m_0, m_1 \in M_\lambda & \quad k \leftarrow K_\lambda \\
& \quad c \leftarrow G(k) \oplus m_0 \\
& \quad b' \\
\end{align*}
\]
An Alternate Proof: Hybrids

Hybrid 1:

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

\[ x \leftarrow \{0,1\}^{s(\lambda)} \]

\[ c \leftarrow x \oplus m_0 \]

\[ b' \]
An Alternate Proof: Hybrids

Hybrid 2:

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

\[ x \leftarrow \{0,1\}^{s(\lambda)} \]

\[ c \leftarrow x \oplus m_1 \]

\[ b' \]
An Alternate Proof: Hybrids

Hybrid 3: IND-Exp$_1$

$m_0, m_1 \in M_\lambda$

$k \leftarrow K_\lambda$

$c \leftarrow G(k) \oplus m_1$

$b'$
An Alternate Proof: Hybrids

\[
\left| \Pr[b'=1 : \text{IND-Exp}_0] - \Pr[b'=1 : \text{IND-Exp}_1] \right| \\
= \left| \Pr[b'=1 : \text{Hyb}_0] - \Pr[b'=1 : \text{Hyb}_3] \right| \\
\leq \left| \Pr[b'=1 : \text{Hyb}_0] - \Pr[b'=1 : \text{Hyb}_1] \right| \\
+ \left| \Pr[b'=1 : \text{Hyb}_1] - \Pr[b'=1 : \text{Hyb}_2] \right| \\
+ \left| \Pr[b'=1 : \text{Hyb}_2] - \Pr[b'=1 : \text{Hyb}_3] \right|
\]

If \( |\Pr[b'=1:\text{IND-Exp}_0]-\Pr[b'=1:\text{IND-Exp}_1]| \geq \varepsilon(\lambda) \),

Then for some \( i=0,1,2, \)

\( |\Pr[b'=1:\text{Hyb}_i]-\Pr[b'=1:\text{Hyb}_{i+1}]| \geq \varepsilon(\lambda)/3 \)
An Alternate Proof: Hybrids

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

\[
k \leftarrow K_\lambda
\]

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

\[
c \leftarrow G(k) \oplus m_0
\]

\[
x \leftarrow \{0,1\}^{s(\lambda)}
\]

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

\[
c \leftarrow x \oplus m_0
\]

\[
b'
\]
Suppose \( \hat{\epsilon} \) distinguishes Hybrid 0 from Hybrid 1 with advantage \( \epsilon(\lambda)/3 \) \( \Rightarrow \) Construct

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

\[ c \leftarrow x \oplus m_0 \]

\[ b' \]
An Alternate Proof: Hybrids

Suppose $\text{ distinguishing Hybrid 0 from Hybrid 1 with advantage } \varepsilon(\lambda)/3 \implies \text{ Construct}$

If $\text{ is given } G(s) \text{ for a random } s, \text{ sees Hybrid 0}$
If $\text{ is given } x \text{ for a random } x, \text{ sees Hybrid 1}$

Therefore, advantage of $\text{ is equal to advantage of } \varepsilon(\lambda)/3 \implies \text{ Contradiction!}$
An Alternate Proof: Hybrids

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

\[
x \leftarrow \{0,1\}^{s(\lambda)}
\]

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

\[
c \leftarrow x \oplus m_0
\]

\[
b' \downarrow
\]

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

\[
c \leftarrow x \oplus m_1
\]

\[
b' \downarrow
\]
An Alternate Proof: Hybrids

Suppose distinguishes Hybrid 1 from Hybrid 2 with advantage \( \varepsilon = \frac{\lambda}{3} \), \( m_0, m_1 \in M_\lambda \)

\( c \leftarrow \{0,1\}^s(\lambda) \)

\( x \leftarrow \{0,1\}^{s(\lambda)} \)

\( x \oplus m_1 \)

Impossible by OTP security
An Alternate Proof: Hybrids

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

Proof essentially identical to Hybrid 0/Hybrid 1 case
PRG Discussion

Do we need to restrict to PPT? YES!

Reason:
• Impossibility for statistically secure encryption gives exponential-time adversary. Apply reduction to get exponential-time.
PRG Discussion

Do we need to restrict to PPT?

YES!

Reason:
• More obvious: Brute force attack

For each \( s \) in \( \{0,1\}^\lambda \):
    If \( G(s) = x \), output 1
    If no \( s \) found, output 0

(either \( G(s) \) or truly random)
PRG Discussion

For each $s$ in $\{0,1\}^\lambda$:
- If $G(s) = x$, output 1
- If no $s$ found, output 0

If $x = G(s)$ for random $s$, will always output 1

If $x$ is truly random, will output 1 with probability at most $2^{\lambda-s(\lambda)} \leq \frac{1}{2}$
- $2^{s(\lambda)}$ possible values of $x$, only $\leq 2^s$ possible values of $G(s)$
“Bergofsky Principle”

(Not a real principle)

(False) Bergofsky Principle: you can brute force any cryptosystem to learn the key

Can you think of an example that contradicts this?
Brute Forcing OTP?

Say I see the ciphertext

\[ \text{ciphertext: AKFLRKATEOMH} \]

I try all keys, and I see that

\[ \begin{align*}
\text{key:} & \quad \text{ARMLPAAABOQU} \\
\text{message:} & \quad \text{attackatdawn}
\end{align*} \]

Is this the right key/message? Who knows...

\[ \begin{align*}
\text{key:} & \quad \text{ARMLPAAABUVX} \\
\text{message:} & \quad \text{attackatdusk}
\end{align*} \]
When Is Brute Force Possible?

Possible:
- PRGs
- Shift cipher (unlikely two shifts give valid English)
- Substitution cipher
- Encryption where $|key| < |message|$

Impossible:
- OTP
- Anything else?
“Bergofsky Principle”

(Not a real principle)

If you want to find a secret from a finite set, and if given a candidate secret from that set it is possible to tell if the secret is correct, then you can always find the secret given enough time.
Do PRGs Exist?

If $P = NP$, the answer is NO

• Language $L = \{x: \exists s, G(s) = x\}$

• $s$ is an efficiently verifiable witness that $x \in L$

• Therefore, $L \in NP$

• If $P = NP$, can decide $L$ in polynomial time

  $\Rightarrow$ break PRG security of $G$
Do PRGs Exist?

Therefore, we need to at least assume $P \neq NP$

Fortunately, most people believe $P \neq NP$

But, huge open question:
   Can we build PRGs assuming $P \neq NP$?

Big difficulty:
   $P \neq NP$ is a worst case assumption, whereas PRGs are average case
Does Crypto Exist?

In most cases, crypto requires **P≠NP**
• We can usually efficiently check if a key is correct
• Therefore, if **P=NP**, we can also find the key in efficiently
• There are some exceptions: notable example is OTP

However, most crypto seems to require something stronger

Again, **P≠NP** is worst case, whereas most crypto definitions are average case

**P≠NP** is necessary but not sufficient for most crypto
Assumptions in Cryptography

For most crypto, will need to make certain computational assumptions
• E.g. that $G$ is a PRG

Obviously, unsatisfying state of affairs

To gain confidence in assumption, need to perform extensive cryptanalysis
Assumptions in Cryptography

To gain confidence in assumption, need to perform extensive cryptanalysis
• Expensive and time consuming

Ideally, we would only do this once
• Don’t want to have to perform cryptanalysis every time we design a new scheme
Provable Security

Major goal in cryptography:

Use one component (e.g. PRG) for many cryptographic tasks (e.g. Encryption), with security proof assuming just the security of the component

When we say to “prove” security, we mean relative to the assumption that building block is secure

Exactly what this means should generally be clear from context
Back To PRGs

PRGs usually allow for streaming arbitrarily long sequences of random bits
Stream Ciphers

Use “streaming” PRG to encrypt messages
Stream Ciphers

Use “streaming” PRG to encrypt messages

In this way, can encrypt arbitrarily long messages

But remember, stream ciphers are really just OTP’s, so still cannot encrypt twice with the same part of keystream

Instead, encrypt like we did with the one-time pad
Multiple Messages with Stream Ciphers
Multiple Messages with Stream Ciphers
Limitations of Stream Ciphers

Just like with OTP, need to be careful because communication may be asynchronous
  • Keep a different key/state for each direction of communication

Here, even bigger problem cause by out of order messages
Multiple Messages with Stream Ciphers
Multiple Messages with Stream Ciphers
Multiple Messages with Stream Ciphers
Multiple Messages with Stream Ciphers

\[ \text{state}_{n-1} \oplus \text{state}_n \oplus \text{key} \]

1  2  3  \ldots  n

\[ \text{state}_{n-1} \rightarrow \text{state}_n \]

Alice

Minion

\[ n \rightarrow n \]
Multiple Messages with Stream Ciphers

Bob needs to either:
- Store entire keystream until he receives message 1
- Or re-compute keystream from scratch every message
Multiple Messages with Stream Ciphers

Basic stream ciphers as described unsuitable for many applications

• Memory/computationally constrained devices

Solution?

• Self-synchronizing stream cipher
Self-Synchronizing Stream Ciphers

“state” is just (key, last several ciphertext bits seen)

Thus, you can always decrypt if the last several ciphertext bits were correct
Summary

Computational assumptions crucial to cryptography

First building block: PRGs
• Use to build stream ciphers

Security proofs by reduction
• Hybrid arguments
Next Time

Some design principles behind PRGs