Information Security
CS 526
Topic 3

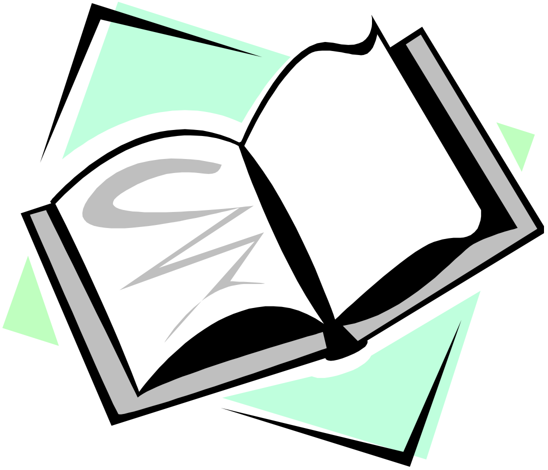
Ciphers and Cipher Security: Stream Ciphers, Block Ciphers, Perfect Secrecy, and IND-CPA Security

Announcements

- HW1 is out, due on Sept 10
  - Start early, late policy is 3 total late days for HW and Project
  - Team project has separate late days

Readings for This Lecture

- Required reading from wikipedia
  - Stream cipher
  - Pseudorandom number generator
  - Block Cipher
  - Block cipher modes of operation
  - Information theoretic security
  - Ciphertext Indistinguishability



Notation for Symmetric-key Encryption

- A symmetric-key encryption scheme is comprised of three algorithms
  - **Gen** the key generation algorithm
    - The algorithm must be probabilistic/randomized
    - Output: a key k
  - **Enc** the encryption algorithm
    - Input: key k, plaintext m
    - Output: ciphertext c := $\mathbf{Enc}_k(m)$
  - **Dec** the decryption algorithm
    - Input: key k, ciphertext c
    - Output: plaintext m := $\mathbf{Dec}_k(m)$

Requirement: $\forall k\ \forall m\ [\ \mathbf{Dec}_k(\mathbf{Enc}_k(m)) = \mathrm{m}\ ]$

Stream Ciphers (An Approximation of One-Time Pad)

- In One-Time Pad, a key is a random string of length at least the same as the message
- Stream ciphers:
  - Idea: replace "rand" by "pseudo rand"
  - Use Pseudo Random Number Generator
  - PRNG: $\{0,1\}^s \rightarrow \{0,1\}^n$
    - expand a short (e.g., 128-bit) random seed into a long (typically unbounded) string that "looks random"
  - Secret key is the seed
  - Basic encryption method: $E_{key}[M] = M \oplus PRNG(key)$

The RC4 Stream Cipher

- A proprietary cipher owned by RSA, designed by Ron Rivest in 1987.
- Became public in 1994.
- Simple and effective design.
- Variable key size (typical 40 to 256 bits),
- Output length unbounded
- Widely used (web SSL/TLS, wireless WEP).
- Extensively studied, not a completely secure PRNG, first part of output biased, when used as stream cipher, should use RC4-Drop[n]
  - Which drops first n bytes before using the output
  - Conservatively, set n=3072

Stream Ciphers = Cryptographically Strong Pseudo Random Number Generators

- Useful for cryptography, simulation, randomized algorithm, etc.
  - Stream ciphers, generating session keys
- The same seed always gives the same output stream
  - Why is this necessary for stream ciphers?
- Simulation requires uniform distributed sequences
  - E.g., having a number of statistical properties
- **Cryptographically secure pseudo-random number generator** requires unpredictable sequences
  - satisfies the "next-bit test“: given consecutive sequence of bits output (but not seed), next bit must be hard to predict
- Some PRNG’s are weak: knowing output sequence of sufficient length, can recover key.
  - Do not use these for cryptographic purposes

Security Properties of Stream Ciphers

- Under known plaintext, chosen plaintext, or chosen ciphertext, the adversary knows the key stream (i.e., PRNG(key))
  - Security depends on strength of PRNG
  - PRNG must be "unpredictable"
    - Precise definition is foundation of modern crypto
- How to break a stream cipher in a brute-force way?
- If the same key stream is used twice, then easy to break.
  - This is a fundamental weakness of stream ciphers; it exists even if the PRNG used in the ciphers is strong

Using Stream Ciphers in Practice

- In practice, one key is used to encrypt many messages
  - Example: Wireless communication
  - Solution: Use Initial vectors (IV).
  - $E_{key}[M] = [IV, M \oplus PRNG(key \,||\, IV)]$
    - IV is sent in clear to receiver;
    - IV needs integrity protection, but not confidentiality protection
    - IV ensures that key streams do not repeat, but does not increase cost of brute-force attacks
    - Without key, knowing IV still cannot decrypt
  - Need to ensure that IV never repeats! How?

Randomized vs. Deterministic Encryption

- Encryption can be randomized,
  - i.e., same message, same key, running the encryption algorithm twice results in two different ciphertexts
  - E.g, $\mathbf{Enc}_k[m] = (r, \mathrm{PRNG}[k||r] \oplus m)$, i.e., the ciphertext includes two parts, a randomly generated r, and a second part
- Decryption is deterministic in the sense that
  - For the same ciphertext and same key, running decryption algorithm twice always results in the same plaintext
- Each key induces a one-to-many mapping from plaintext space to ciphertext space
  - Corollary: ciphertext space must be equal to or larger than plaintext space

Block Ciphers

- An n-bit plaintext is encrypted to an n-bit ciphertext
  - $\mathcal{P}$: $\{0,1\}^n$
  - $\mathcal{C}$: $\{0,1\}^n$
  - $\mathcal{K}$: $\{0,1\}^s$
  - **E**: $\mathcal{K} \times \mathcal{P} \rightarrow \mathcal{C}$: E_k: a permutation on $\{0,1\}^n$
  - **D**: $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{P}$: D_k is E_k^{-1}
  - Block size: n
  - Key size: s

Data Encryption Standard (DES)

- Designed by IBM, with modifications proposed by the National Security Agency
- US national standard from 1977 to 2001
- De facto standard
- Block size is 64 bits;
- Key size is 56 bits
- Has 16 rounds
- Designed mostly for hardware implementations
  - Software implementation is somewhat slow
- Considered insecure now
  - Vulnerable to brute-force attacks
- Triple DES: $E_{k3}D_{k2}E_{K1}(M)$ has 112-bit strength, but slow

Advanced Encryption Standard

- Starting 1999: replace DES as the standard for block ciphers.
- **Aka. Rijndael** (invented by Joan Daemen and Vincent Rijmen)
- Designed to be efficient in both hardware and software across a variety of platforms.
- Block size: 128 bits
- Variable key size: **128, 192, or 256 bits.**
- No known exploitable algorithmic weaknesses
- Implementation may be vulnerable to timing attacks (largely due to CPU's architecture of using cache)
- Intel now has AES-NI, CPU-based implementation for AES
  - Timing-based side channel attacks no longer an issue

Need for Encryption Modes

- A block cipher encrypts only one block
- Needs a way to extend it to encrypt an arbitrarily long message
- Want to ensure that if the block cipher is secure, then the encryption is secure
- Aims at providing Semantic Security (**IND-CPA)** assuming that the underlying block ciphers are strong

Block Cipher Encryption Modes: ECB

- Message is broken into independent blocks;
- Electronic Code Book (ECB): each block encrypted separately.
- **Encryption: $c_i = E_k(x_i)$**
- **Decrytion: $x_i = D_k(c_i)$**

Properties of ECB

- Deterministic:
  - the same data block gets encrypted the same way,
    - reveals patterns of data when a data block repeats
  - when the same key is used, the same message is encrypted the same way

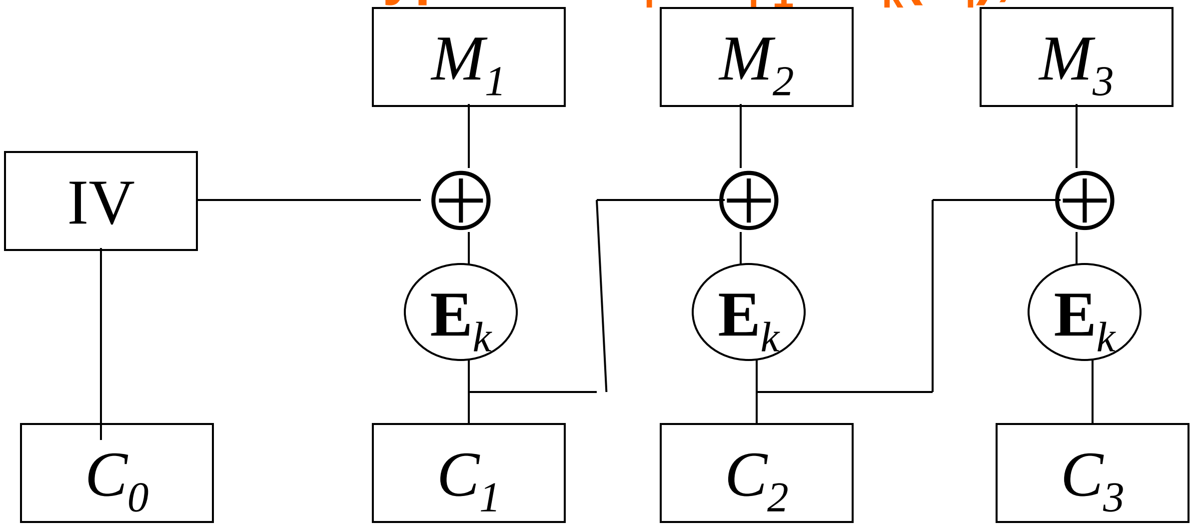
- Usage: not recommended to encrypt more than one block of data

DES Encryption Modes: CBC

- Cipher Block Chaining (CBC):
  - Uses a random Initial Vector (IV)
  - Next input depends upon previous output

Encryption: $C_i = E_k(M_i \oplus C_{i-1})$, **with** $C_0 = IV$

Decryption: $M_i = C_{i-1} \oplus D_k(C_i)$, **with** $C_0 = IV$



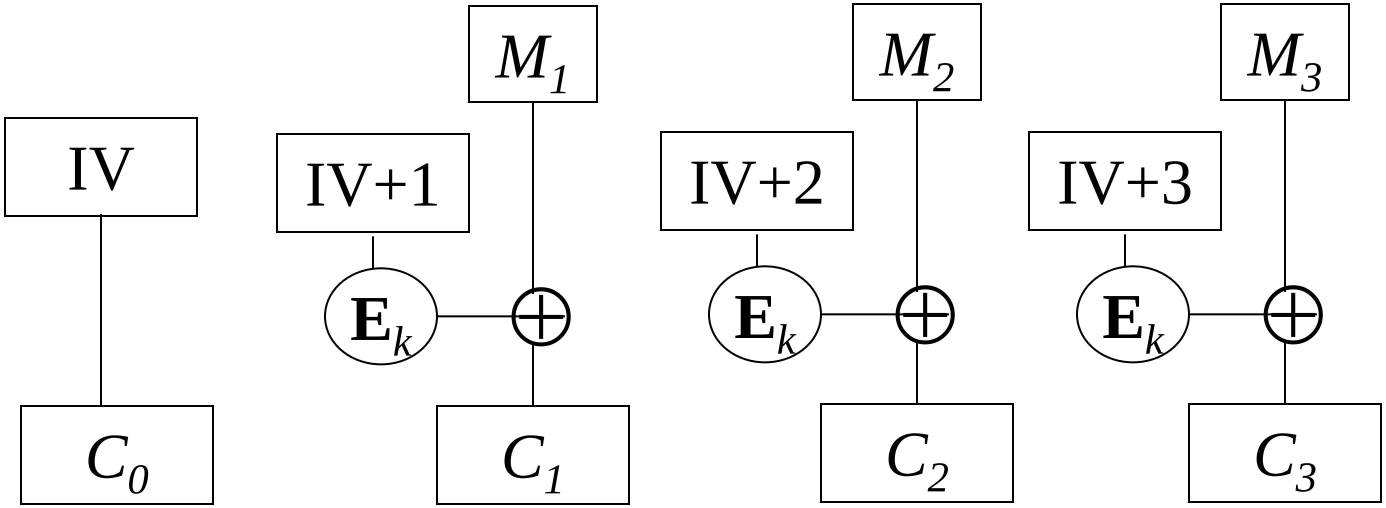
Properties of CBC

- Randomized encryption: repeated text gets mapped to different encrypted data.
  - IV must be randomly chosen to get this benefit

- Each ciphertext block depends on all preceding plaintext blocks.
- Usage: IV must be random, needs **integrity** but not confidentiality
  - The IV is not secret (it is part of ciphertext)
  - The adversary cannot control the IV

Encryption Modes: CTR

- Counter Mode (CTR): Defines a stream cipher using a block cipher
  - Uses a random IV, known as the counter
  - Encryption: C_0=IV, $C_i = M_i \oplus E_k[IV+i]$
  - Decryption: IV=C_0, $M_i = C_i \oplus E_k[IV+i]$



Properties of CTR

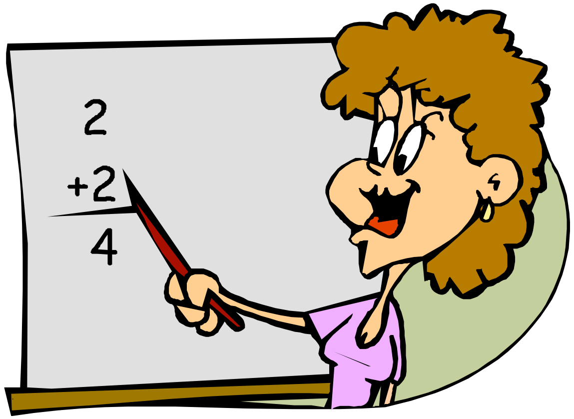
- Gives a stream cipher from a block cipher
- Randomized encryption:
  - when starting counter is chosen randomly
- Random Access: encryption and decryption of a block can be done in random order, very useful for hard-disk encryption.
  - E.g., when one block changes, re-encryption only needs to encrypt that block. In CBC, all later blocks also need to change

Encryption Schemes Commonly User

- RC4 (with new IV for each message)
- AES+ECB (Only in very restricted scenarios)
- AES+CBC
- AES+CTR

- Are they secure?
- We need probability theory to study that.

Begin Math



Random Variable

Definition

A **discrete random variable, X,** consists of a finite set $\mathcal{X}$, and a probability distribution defined on $\mathcal{X}$. The probability that the random variable **X** takes on the value x is denoted **Pr**[**X** =x]; sometimes, we will abbreviate this to **Pr**[x] if the random variable **X** is fixed. It must be that

$$0 \leq \Pr[x] \quad \text{for all } x \in \mathcal{X}$$

$$\sum_{x \in \mathcal{X}} \Pr[x] = 1$$

Example of Random Variables

- Let random variable $\mathbf{D_1}$ denote the outcome of throwing one die (with numbers 0 to 5 on the 6 sides) randomly, then $\mathcal{D}$={0,1,2,3,4,5} and $\mathbf{Pr}[\mathbf{D_1}=i] = 1/6$ for $0 \le i \le 5$
- Let random variable $\mathbf{D_2}$ denote the outcome of throwing a second such die randomly
- Let random variable $\mathbf{S_1}$ denote the sum of the two dice, then $\mathcal{S}$ ={0,1,2,…,10}, and

  $\mathbf{Pr}[\mathbf{S_1}=0] = \mathbf{Pr}[\mathbf{S_1}=10] = 1/36$

  $\mathbf{Pr}[\mathbf{S_1}=1] = \mathbf{Pr}[\mathbf{S_1}=9] = 2/36 = 1/18$

  …
- Let random variable $\mathbf{S_2}$ denote the sum of the two dice modulo 6, what is the distribution of $\mathbf{S_2}$?

Relationships between Two Random Variables

Definitions

Assume **X** and **Y** are two random variables, then we define:

- joint probability: **Pr**[x, y] is the probability that **X** takes value x and **Y** takes value y.
- conditional probability: **Pr**[x|y] is the probability that **X** takes value x given that **Y** takes value y.

$$\mathbf{Pr}[x|y] = \mathbf{Pr}[x, y] \; / \; \mathbf{Pr}[y]$$

- independent random variables: **X** and **Y** are said to be independent if **Pr**[x,y] = **Pr**[x]P[y], for all $x \in \mathcal{X}$ and all $y \in \mathcal{Y}$.

Examples

- Joint probability of $\mathbf{D_1}$ and $\mathbf{D_2}$
  for $0 \le i,j \le 5$, $\mathbf{Pr}[\mathbf{D_1}=i, \mathbf{D_2}=j] = ?$

- Are $\mathbf{D_1}$ and $\mathbf{D_2}$ independent?

- Suppose $\mathbf{D_1}$ is plaintext and $\mathbf{D_2}$ is key, and $\mathbf{S_1}$ and $\mathbf{S_2}$ are ciphertexts of two different ciphers, which cipher would you use?

Examples to think after class

- What is the joint probability of $\mathbf{D_1}$ and $\mathbf{S_1}$?
- What is the joint probability of $\mathbf{D_2}$ and $\mathbf{S_2}$?

- What is the conditional probability $\mathbf{Pr}[\mathbf{S_1}=s \mid \mathbf{D_1}=i]$ for $0 \le i \le 5$ and $0 \le s \le 10$?
- What is the conditional probability $\mathbf{Pr}[\mathbf{D_1}=i \mid \mathbf{S_2}=s]$ for $0 \le i \le 5$ and $0 \le s \le 5$?

- Are $\mathbf{D_1}$ and $\mathbf{S_1}$ independent?
- Are $\mathbf{D_1}$ and $\mathbf{S_2}$ independent?

Bayes' Theorem

If P[y] > 0 then

$$P[x \mid y] = \frac{P[x]P[y \mid x]}{P[y]}$$

$$P[y] = \sum_{x \in X} P[x, y] = \sum_{x \in X} P[x] p[y \mid x]$$

Corollary

X and Y are independent random variables iff P[x|y] = P[x], for all x $\in$ X and all y $\in$ Y.

Example: What is **Pr**[$\mathbf{D_1}$=1 | $\mathbf{S_1}$=3] ?

End Math

Shannon (Information-Theoretic) Security = Perfect Secrecy

- Basic Idea: Ciphertext should reveal no "information" about Plaintext
- Intuition:
  - Ciphertext should be independent from the plaintext

That is,

$\forall$ message m, $\forall$ ciphertext c

$$\Pr[\mathbf{PT}=m \wedge \mathbf{CT}=c] = \Pr[\mathbf{PT}=m] \Pr[\mathbf{CT}=c]$$

Or, equivalently

$\forall$ message m1, m2, $\forall$ ciphertext c

$$\Pr[\mathbf{CT}=c \mid \mathbf{PT}=m1] = \Pr[\mathbf{CT}=c \mid PT=m2]$$

Example for Information Theoretical Security

- Consider an example of encrypting the result of a 6-side dice (1 to 6).
  - Method 1: randomly generate K=[1..6], ciphertext is result + K.
    - What is plaintext distribution? After seeing that the ciphertext is 3, what could be the plaintext. After seeing that the ciphertext is 12, what could be the plaintext?
  - Method 2: randomly generate K=[1..6], ciphertext is (result + K) mod 6.
    - Same questions.
    - Can one do a brute-force attack?

Perfect Secrecy

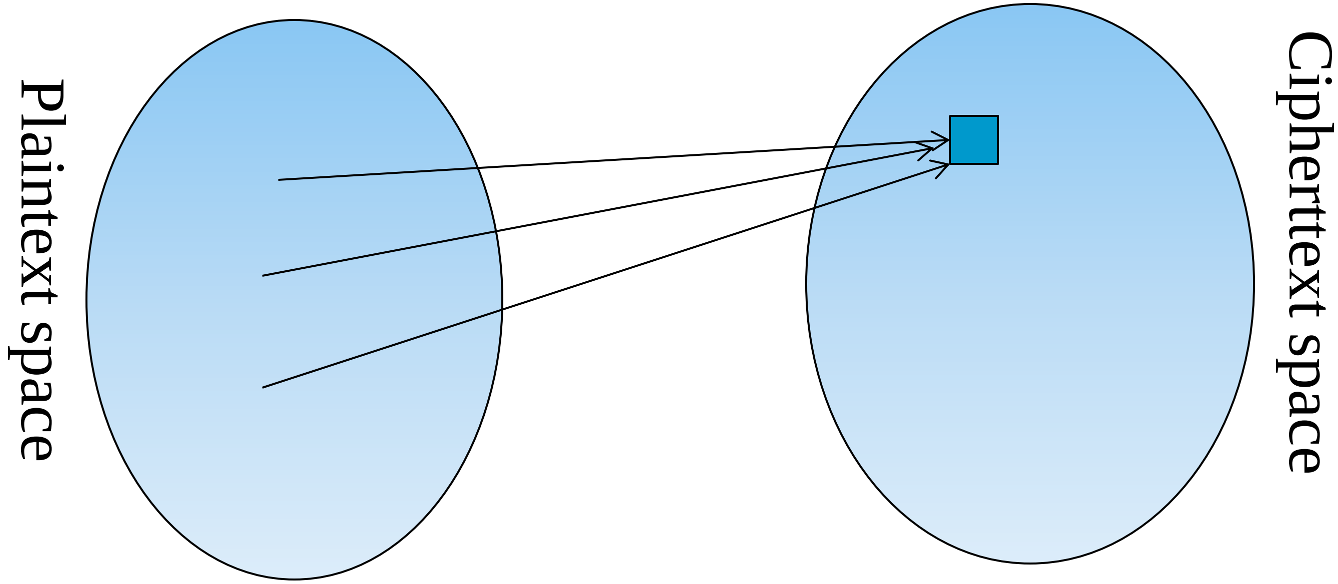
- Fact: When keys are uniformly chosen in a cipher, the cipher has perfect secrecy iff. the number of keys encrypting M to C is the same for any (M,C)
  - This implies that
    $\forall c \forall m_1 \forall m_2 \Pr[\mathbf{CT}=c \mid \mathbf{PT}=m_1] = \Pr[\mathbf{CT}=c \mid \mathbf{PT}=m_2]$

- One-time pad has perfect secrecy when limited to messages over the same length (Proof?)

The "Bad News" Theorem for Perfect Secrecy

- Question: OTP requires key as long as messages, is this an inherent requirement for achieving perfect secrecy?
- Answer. Yes. Perfect secrecy implies that
  key-length ≥ msg-length

Proof:



- Implication: Perfect secrecy difficult to achieve in practice

Towards Computational Security

- Perfect secrecy is too difficult to achieve.

- Computational security uses two relaxations:
  - Security is preserved only against **efficient** (computationally bounded) adversaries
    - Adversary can only run in feasible amount of time
  - Adversaries can potentially succeed with some **very small probability** (that we can ignore the case it actually happens)

Defining Security

- Desire “semantic security”, i.e., having access to the ciphertext does not help adversary to compute any function of the plaintext.
  - Difficult to use
- Equivalent notion: Adversary cannot distinguish between the ciphertexts of two plaintexts

Towards IND-CPA Security:

- Ciphertext Indistinguishability under a Chosen-Plaintext Attack: Define the following IND-CPA experiment :
  - Involving an Adversary and a Challenger
  - Instantiated with an Adversary algorithm *A*, and an encryption scheme Π = (Gen, Enc, Dec)

Challenger		Adversary
$k \leftarrow \text{Gen}()$	$\xrightarrow{\text{Enc}_k[]}$	
	$\xleftarrow{m_0,\ m_1}$	chooses $m_0, m_1 \in \mathcal{M}$
$b \leftarrow_R \{0,1\}$	$\xrightarrow{C=\mathbf{Enc}_k[m_b]}$	
	$\xleftarrow{b' \in \{0,1\}}$	

Adversary wins if b=b'

The IND-CPA Experiment Explained

- A k is generated by Gen()
- Adversary is given oracle access to $Enc_k(\cdot)$,
  - Oracle access: one gets its question answered without knowing any additional information
- Adversary outputs a pair of equal-length messages m_0 and m_1
- A random bit b is chosen, and adversary is given $Enc_k(m_b)$
  - Called the challenge ciphertext
- Adversary does any computation it wants, while still having oracle access to $Enc_k(\cdot)$, and outputs b'
- Adversary wins if b=b'

Intuition of IND-CPA security

- Perfect secrecy means that any plaintext is encrypted to a given ciphertext with the same probability, i.e., given any pair of M_0 and M_1, the probabilities that they are encrypted into a ciphertext C are the same
  - Hence no adversary can tell whether C is ciphertext of M_0 or M_1.

- IND-CPA means
  - With bounded computational resources, the adversary cannot tell which of M_0 and M_1 is encrypted in C

Computational Security vs. Information Theoretic Security

- If a cipher has only computational security, then it can be broken by a brute force attack, e.g., enumerating all possible keys
  - Weak algorithms can be broken with much less time
- How to prove computational security?
  - Assume that some problems are hard (requires a lot of computational resources to solve), then show that breaking security means solving the problem
- Computational security is foundation of modern cryptography.

Security of Ciphers

- Stream ciphers can be used to achieve IND-CPA security when the underlying PRNG is cryptographically strong
- ECB does not have semantic security
  - How to break the semantic security (IND-CPA) of a block cipher with ECB?
- CBC is proven to provide IND-CPA assuming that the block cipher is secure and that IV's are randomly chosen
- CTR is proven IND-CPA secure assuming that block cipher is secure and that IV's are randomly chosen

Coming Attractions …

- Cryptography: Message Authentication Code and Cryptographic Hash Functions