Outline

Introduction to cryptography
- Secret Key Cryptography (symmetric crypto)
- Modes of Operation of Encryption Algorithms
  - ECB, CBC, OFB, CFB, CTR
- Hashes and Message Authentication Codes
- Public Key Algorithms (asymmetric crypto)

Why, How, What?

Cryptography provides key building block for many network security services
- Security services:
  - Authentication, Confidentiality, Integrity, Access control, Non-repudiation, availability, key management, audit
- Cryptographic algorithms (building blocks):
  - Encryption: symmetric encryption (e.g., DES, AES), asymmetric encryption (e.g., RSA, El-Gamal)
  - Hashing functions
  - Message Authentication Code (e.g., HMAC + SHA1)
  - Digital signature functions (e.g., RSA, El-Gamal)
**Terminology**

- **Security services:**
  - Authentication, confidentiality, integrity, access control, non-repudiation, availability, key management, audit
- **Security attacks:**
  - Passive, active
- **Cryptography models:**
  - Symmetric (secret key), asymmetric (public key)
- **Cryptanalysis:**
  - Ciphertext only, known plaintext, chosen plaintext, chosen ciphertext, chosen text

**Security services**

- **Authentication:**
  - Assures the recipient of a message the authenticity of the claimed source
- **Access control:**
  - Limits the access to authorized users
- **Confidentiality:**
  - Protects against unauthorized release of message content
- **Integrity:**
  - Ensures that a message is received as sent
- **Non-repudiation:**
  - Protects against sender/receiver denying sending/receiving a message
- **Availability:**
  - Guarantees that the system services are always available when needed
- **Security audit:**
  - Keeps track of transactions for later use (diagnostic, alarms...)
- **Key management:**
  - Allows to negotiate, setup and maintain keys between communicating entities

**Security Attacks**

- **Security attacks:**
  - Interception (confidentiality)
  - Interruption (availability)
  - Modification (integrity)
  - Fabrication (authenticity)
- **Kent’s classification**
  - Passive attacks:
    - Release of message content
  - Active attacks:
    - Masquerade
    - Replay
    - Modification of message
    - Denial of service
Kerchoff's Principle

- The cipher should be secure even if the intruder knows all the details of the encryption process except for the secret key.
- "No security by obscurity"
  - Examples of systems that did not follow this rule and failed.

Securing Networks

- Where to put the security in a protocol stack?
- Practical considerations:
  - End to end security
  - No modification to OS

Some Building Blocks of Cryptography/Security

- Encryption algorithms
  - Block ciphers
- One-way hashing functions (={message digest, cryptographic checksum, message integrity check, etc.)
  - Input: variable length string
  - Output: fixed length (generally smaller) string
  - Desired properties:
    - Hard to generate a pre-image (input) string that hashes to a given string, second preimage, and collisions

- One-way functions
  - $y = f(x)$: easy to compute
  - $f^{-1}(y)$: much harder to reverse (it would take millions of years)
  - Example:
    - Multiplication of 2 large prime numbers versus factoring
  - Examples include:
    - Discrete exponentiation/discrete logarithms

- Protocols
  - Authentication, key management, etc.
Encryption

Basic Goal:
- Allow two entities (e.g., Alice, and Bob) to communicate over an insecure channel, such that an opponent (e.g., Oscar) cannot understand what is being communicated

Encryption Algorithms

- Block vs. Stream ciphers
  - Block ciphers:
    - Input: block of \( n \) bits; Output: block of \( n \) bits
    - Examples: AES, DES
  - Stream ciphers:
    - Input: stream of symbols; Output: stream of symbols
    - Examples: GSM AS, RC4
  - Block ciphers can be used to build stream ciphers (under some assumptions)
    - Examples: AES-CBC

Encryption Models

- Symmetric encryption (conventional encryption)
  - Encryption Key = Decryption Key
  - I.e., Decryption key can be derived from encryption key
  - E.g., AES, DES, FEAL, IDEA, BLOWFISH
- Asymmetric encryption
  - Encryption Key \neq Decryption Key
  - I.e., Decryption key cannot be derived from encryption key
  - E.g., RSA, Diffie-Hellman, ElGamal
Encryption Models

<table>
<thead>
<tr>
<th>Message source</th>
<th>Plaintext</th>
<th>Encryption Algorithm</th>
<th>Ciphertext</th>
<th>Decryption Algorithm</th>
<th>Plaintext</th>
<th>Message destination</th>
</tr>
</thead>
</table>

Symmetric encryption: Shared key
Asymmetric encryption: Public key

Symmetric vs. Asymmetric Algorithms

- Symmetric algorithms are much faster
  - In the order of a 1000 times faster
- Symmetric algorithms require a shared secret
  - Impractical if the communicating entities don't have another secure channel
- Both algorithms are combined to provide practical and efficient secure communication
  - E.g., establish a secret session key using asymmetric crypto and use symmetric crypto for encrypting the traffic

Attacks on Encrypted Messages

- Ciphertext only:
  - encryption algorithm, ciphertext to be decoded
- Known plaintext:
  - encryption algorithm, ciphertext to be decoded, pairs of (plaintext, ciphertext)
- Chosen plaintext:
  - encryption algorithm, ciphertext to be decoded, plaintext (chosen by cryptanalyst) + corresponding ciphertext
- Chosen ciphertext:
  - encryption algorithm, ciphertext to be decoded, ciphertext (chosen by cryptanalyst) + corresponding plaintext
- Chosen text:
  - encryption algorithm, ciphertext to be decoded, plaintext + corresponding ciphertext (both can be chosen by attacker)
Secret Key Cryptography
= Symmetric Cryptography
= Conventional Cryptography

Examples of Encryption Algorithms
- Advances Encryption Algorithm (AES)
  - Block size: 128 bits
  - Key size: 128/192/256
- Data Encryption Standard (DES) – not secure
  - Block size: 64 bits
  - Key size: 56 bits
- It is not recommended to use DES

Encryption Modes:
Electronic Codebook (ECB)
Counter (CTR)

- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)
  \[ C_i = P_i \text{ XOR } O_i \]
  \[ O_i = \text{DES}_{K_1}(i) \]
- Uses: high-speed network encryptions, random access to files

Symmetric Encryption Algorithms Internals

- Historical ciphers
  - Not necessary to understand all the details

Symmetric cryptosystems (conventional cryptosystems)

- Substitution techniques:
  - Caesar cipher
    - Replace each letter with the letter standing \( x \) places further
    - Example: \( x = 3 \)
      - plain: meet me after the toga party
      - cipher: phhw ph diwhu whk urjd edwub
    - Key space: 25
    - Brut force attack: try 25 possibilities
  - Monoalphabetic ciphers
    - Arbitrary substitution of alphabet letters
    - Key space: \( 26! > 6\times10^{26} > \text{key-space (DES)} \)
    - Attack if the nature of the plaintext is known (e.g., English text):
      - compute the relative frequency of letters and compare it to standard distribution for English (e.g., E:12.7, T:9, etc.)
      - compute the relative frequency of 2-letter combinations (e.g., TH)
Symmetric cryptosystems (Continued)

- **Multiple-Letter Encryption (Playfair cipher)**
  - Plaintext is encrypted two-letters at a time
  - Based on a 5x5 matrix
  - Identification of individual digraphs is more difficult (25x25 possibilities)
  - A few hundred letters of ciphertext allow to recover the structure of plaintext (and break the system)
  - Used during World War I & II
- **Polyalphabetic Ciphers (Vigenère cipher)**
  - 26 Caesar ciphers, each one denoted by a key letter
  - plain: unenciphered secret message
  - cipher: 6ITZTPSRMHRVTHPVQSMJ
  - Enhancement: auto-key (key = initial||plaintext)
- **Rotor machines: multi-round monoalphabetic substitution**
  - Used during WWII by Germany (ENIGMA) and Japan (Purple)

One-Time Pad

- Introduced by G. Vernam (AT&T, 1918), improved by J. Mauborgne
- **Scheme:**
  - Encryption: \( c = p \oplus k \)
  - \( c \): binary digit of plaintext, \( p \): plaintext, \( k \): key
  - Decryption: \( p = c \oplus k \)
  - Key is a random sequence of bits as long as the plaintext
- **One-Time Pad is unbreakable**
  - No statistical relationship between ciphertext and plaintext
- **Example (Vigenère One-Time Pad):**
  - Cipher: ANONYMOUSPILOTISPLASHTION
  - Plain-1 (with k1): NA MUSTARD WITH THE CANDLE
  - Plain-2 (with k2): MISS SCARLET WITH THE KNIFE
  - Share the same long key between the sender & receiver
Transposition/Permutation Techniques

- Based on permuting the plaintext letters
- Example: rail fence technique
  
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>a</td>
<td>t</td>
</tr>
<tr>
<td>e</td>
<td>r</td>
<td>t</td>
</tr>
<tr>
<td>y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
  
  etefeteoasat

- A more complex transposition scheme
  
  | Key: | 6212657 |
  | Plain: | attackpostpone |
  | | dustill |
  | Cipher: | TTHAAAPPNTTDOOAGADCOKHRLYPYXi |

- Attack: letter/diagraph frequency
- Improvement: multiple-stage transposition

Today’s Block Encryption Algorithms

- Key size:
  
  - Too short => easy to guess

- Block size:
  
  - Too short easy to build a table by the attacker: (plaintext, ciphertext)
  - Minimal size: 64 bits

- Properties:
  
  - One-to-one mapping
  - Mapping should look random to someone who doesn’t have the key
  - Efficient to compute/reverse

- How:
  
  - Substitution (small chunks) & permutation (long chunks)
  - Multiple rounds
  
  $\rightarrow$ SPN (Substitution and Permutation Networks) and variants

Data Encryption Standard (DES)

- Developed by IBM for the US government
- Based on Lucifer (64-bits, 128-bits key in 1971)
- To respond to the National Bureau of Standards CFP
  
  - Modified characteristics (with help of the NSA):
    
    - 64-bits block size, 56 bits key length
    - Concerns about trapdoors, key size, sbox structure
  
  - Adopted in 1977 as the DES (FIPS PUB 46, ANSI X3.92) and reaffirmed in 1994 for 5 more years

- Replaced by AES (not secure today)
DES is based on Feistel Structure

\[ L_i = R_{i-1} \]
\[ R_i = L_i \oplus f(R_{i-1}, K_i) \]

Network Security

Cryptography Overview

One DES Round

Key (56 bits)

Key (56 bits)

S-Box Substitution

48-Bit Input

16-Bit Output

- S-Box: heart of DES security
- S-Box: 4x16 entry table
  - Input 6 bits:
    - 2 bits: determine the table (1/4)
    - 4 bits: determine the table entry
  - Output: 4 bits
- S-Boxes are optimized against Differential cryptanalysis
**Double/Triple DES**

- **Double DES**
  - Vulnerable to Meet-in-the-Middle Attack ([DH77])
  
- **Triple DES**
  - Used two keys $K_1$ and $K_2$
  - Compatible with simple DES ($K_1 = K_2$)
  - Used in ISO 8732, PEM, ANS X9.17

**Linear/Differential Cryptanalysis**

- Differential cryptanalysis
  - "Rediscovered" by E. Biham & A. Shamir in 1990
  - Based on a chosen-plaintext attack:
    - Analyze the difference between the ciphertexts of two plaintexts which have a known fixed difference
    - The analysis provides information on the key
  - 8-round DES broken with $2^{14}$ chosen plaintext
  - 16-round DES requires $2^{47}$ chosen plaintext
  - DES design took into account this kind of attacks
  - Linear cryptanalysis
  - Uses linear approximations of the DES cipher (M. Matsui 1993)
  - IDEA first proposal (PES) was modified to resist to this kind of attacks
  - GSM A3 algorithm is sensitive to this kind of attacks
  - SIM card secret key can be recovered => GSM cloning

**Breaking DES**

- Electronic Frontier Foundation built a "DES Cracking Machine" [1998]
  - Attack: brute force
  - Inputs: two ciphertext
  - Architecture:
    - PC
    - Array of custom chips that can compute DES
    - 24 search units/chip x 6 chips/board x 27 boards
  - Power:
    - Searches 92 billion keys per second
    - Takes 4.5 days for half the key space
  - Cost:
    - $130'000 (all the material: chips, boards, cooling, PC etc.)
    - $80'000 (development from scratch)
The Advanced Encryption Standard (AES) Cipher - Rijndael

- Designed by Rijmen-Daemen (Belgium)
- Key size: 128/192/256 bit
- Block size: 128 bit data
- Properties: **iterative** rather than **Feistel** cipher
  - Treats data in 4 groups of 4 bytes
  - Operates on an entire block in every round
- Designed to be:
  - Resistant against known attacks
  - Speed and code compactness on many CPUs
  - Design simplicity

AES

- State: 16 bytes structured in an array

\[
\begin{array}{cccc}
S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\
S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\
S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\
S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\
\end{array}
\]

- Each byte is seen as an element of \( F_2^{256} \)
  - Operations
    - Elements of \( F_2 \) are viewed as polynomials of degree 7 with coefficients \( \{0, 1\} \)
    - Addition: polynomials addition \( \oplus \)
    - Multiplication: polynomials multiplication modulo \( x^8 + x^4 + x^3 + x + 1 \)

AES Outline

1. Initialize State \( \leftarrow x \odot \text{RoundKey} \);
2. For each of the \( N-1 \) rounds:
   1. \( \text{SubBytes(State)} \);
   2. \( \text{ShiftRows(State)} \);
   3. \( \text{MixColumns(State)} \);
   4. \( \text{AddRoundKey(State)} \);
3. Last round:
   1. \( \text{SubBytes(State)} \);
   2. \( \text{ShiftRows(State)} \);
   3. \( \text{AddRoundKey(State)} \);
4. Output y \( \leftarrow \text{State} \)
### Implementation Aspects

- Can be efficiently implemented on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is a simple byte shifting
  - add round key works on byte XORs
  - mix columns requires matrix multiply in GF($2^8$) which works on byte values, can be simplified to use a table lookup

- Can be efficiently implemented on 32-bit CPU
  - redefine steps to use 32-bit words
  - can pre-compute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 16Kb to store tables

- Designers believe this very efficient implementation was a key factor in its selection as the AES cipher

### Hashing Functions and Message Digests

**Goal:**
- Input: long message
- Output: short block (called hash or message digest)
- Desired properties:
  - Pre-image: Given a hash it is computationally infeasible to find a message that produces it
  - Second preimage
  - Collision

**Examples:**
- [Secure Hash Algorithm (SHA-1, SHA-2) by NIST](http://www.slavasoft.com/quickhash/links.htm)
- MD2, MD4, and MD5 by Ron Rivest ([RFC1319, 1320, 1321](http://archive.is/8s5mH))
- SHA-1: output 160 bits
- SHA-2: output 256-384-512 believed to be more secure than others
- SHA-3: ongoing competition with objective of 2012 ([timeline](http://csrc.nist.gov/groups/ST/hash/timeline.html))
Birthday Attacks

- Is a 64-bit hash secure?
  - Brute force: 1ns per hash => $10^{13}$ seconds over 300 thousand years
- But by Birthday Paradox it is not
- Example: what is the probability that at least two people out of 23 have the same birthday? $P > 0.5$
- Birthday attack technique
  - Opponent generates $2^{64}$ variations of a valid message all with essentially the same meaning
  - Opponent also generates $2^{64}$ variations of a desired fraudulent message
    - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
    - have user sign the valid message, then substitute the forgery which will have a valid signature
- Need to use larger MACs

Message Digest 5 (MD5) by R. Rivest [RFC1321]

- Input: message of arbitrary length
- Output: 128-bit hash
- Message is processed in blocks of 512 bits (padding if necessary)
- Security: not recommended
  - Designed to resist the Birthday attack
  - Collisions where found in MD5, SHA-0, and almost found for SHA-1
  - MD5 considered harmful today: creating a rogue CA certificate, Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, December 30, 2008

Applications of Hashing Functions

- Authentication: how?
- Encryption: how?
- Message Authentication Codes
Message Authentication Code (MAC) Using an Encryption Algorithm

Also called Message Integrity Code (MIC)

- Goal:
  - Detect any modification or forgery of the content by an attacker
- Some techniques:
  - Simple techniques have flaws
  - Use CBC mode, send only the last block (residue) along with the plaintext message
  - For confidentiality + integrity:
    - Use two keys (one for CBC encryption and one for CBC residue computation)
    - Append a cryptographic hash to the message before CBC encryption
  - New technique: use a Nested MAC technique such as HMAC

HMAC

\[
\text{HMAC}_K(x) = \text{SHA-1}(\text{opad} | \text{SHA-1}(\text{ipad} | x))
\]

- \( \text{ipad} = 3636...36 \)
- \( \text{opad} = 5C5C...5C \)

- HMAC can be combined with any hashing function
- Proven to be secure under some assumptions...

Public Key Systems
Asymmetric cryptosystems
- Invented by Diffie and Hellman [DH76], Merkle
  - When DES was proposed for standardization
  - Asymmetric systems are much slower than the symmetric ones (~1000 times)
- Advantages:
  - does not require a shared key
  - simpler security architecture (no-need to a trusted third party)

RSA Cryptosystem [RSA78]
- $E(M) = M^e \mod n = C$ (Encryption)
- $D(C) = C^d \mod n = M$ (Decryption)
- RSA parameters:
  - $n = pq$, $\varphi(n) = (p-1)(q-1)$ (public, chosen)
  - $e$ with $\gcd(q(n), e) = 1$, $1 < e < \varphi(n)$ (public, chosen)
  - $d = e^{-1} \mod \varphi(n)$ (private, calculated)
- $D(E(M)) = M^{ed} \mod n = M^{e\varphi(n)+1} = M$ (Euler's theorem)

Modular Arithmetic
- Modular addition:
  - E.g., $3 + 5 = 1 \mod 7$
- Modular multiplication:
  - E.g., $3 \times 4 = 5 \mod 7$
- Modular exponentiation:
  - E.g., $3^3 = 6 \mod 7$
- Group, Rings, Finite/Galois Fields ...
Prime Numbers Generation

- Density of primes (prime number theorem):
  \( \pi(x) \sim x/\ln(x) \)
- Sieve of Eratosthenes
- Try if any number less than \( \sqrt{n} \) divides \( n \)
- Based on Fermat's Little Theorem but does not detect Carmichael numbers
  \( b^n \equiv 1 \mod n \)  \( \text{if there exists } t \in \mathbb{N} \text{ s.t. } \gcd(b, n) = 1 \text{ and } b^t \equiv 1 \mod n \text{ then } n \text{ does not pass Fermat's test for half its relatively prime with } b \)
- Solovay-Strassen primality test
  \( \text{If } n \text{ is not prime at least } 50\% \text{ of } b \text{ fail to satisfy the following:} \)
  \( b^{(n-1)/2} \equiv \left\lfloor \frac{n}{2} \right\rfloor \mod n \)
- Rabin-Miller primality test
  \( \text{If } n \text{ is not prime then it is not pseudoprime to at least } 75\% \text{ of } b < n: \)
  Pseudoprime: \( n = 2, n = 4 \mod p \) or \( n = 1 \mod p \) for some \( p \)
  Probabilistic test, deterministic if the Generalized Riemann Hypothesis is true
- Deterministic polynomial time primality test [Agrawal, Kayal, Saxena'2002]

Use of RSA

- Encryption (A wants to send a message to B):
  A uses the public key of B and encrypts \( M \) (i.e., \( E_B(M) \))
  Since only B has the private key, only B can decrypt \( M \)
  (i.e., \( M = D_B(M) \))

- Digital signature (A want to send a signed message to B):
  Based on the fact that \( E_A(D_A(M)) = D_A(E_A(M)) \)
  A encrypts \( M \) using its private key (i.e., \( D_A(M) \)) and sends it to B
  B can check that \( E_A(D_A(M)) = M \)
  Since only A has the decryption key, only can generate this message

Diffie-Hellman Key Exchange

- Based on the difficulty of computing discrete logarithms
- Works also in extension Galois fields: \( GF(p^q) \)
### Attack on Diffie-Hellman Scheme: Public Key Integrity

**Man-in-the-Middle Attack**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>g^x</td>
<td>g^y</td>
</tr>
</tbody>
</table>

Shared key: \( K_{A} = g^x \)

Shared key: \( K_{B} = g^y \)

Message encrypted using \( K_{A} \)

Decrypt using \( K_{A} \); Decrypt using \( K_{B} \)

- Need for a mean to verify the public information: certification
- Another solution: the Interlock Protocol (Rivest & Shamir 1984)

### El Gamal Scheme

**Parameters:**

- \( p \): prime number (public, chosen)
- \( g \): random number (public, chosen)
- \( x \): random number (private, chosen)
- \( y = g^x \mod p \) (public, computed)

**Encryption of message \( M \):**

- Choose random \( k < p-1 \)
- \( a = g^k \mod p \)
- \( b = y^k M \mod p \)

**Decryption:**

- \( M = b^a \mod p = b^{g^x} \mod p = b^y \)

**Message signature:**

- Choose random \( k \) relatively prime with \( p-1 \)
- Find \( c \): \( M = (xa + kb) \mod (p-1) \) (extended Euclid algorithm)
- \( signature(M) = (a, b) \)
- Verify signature: \( y^a \mod p = g^x \mod p \)

### Knapsack

- Introduced by R. Merkle
- Based on the difficulty of solving the Knapsack problem in polynomial time (Knapsack is an NP-complete problem)
  - \( a = (a_1, a_2, \ldots, a_n) \) (seq. Int)
  - \( x = (x_1, x_2, \ldots, x_n) \) (seq. Bits)
  - \( S = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \)
- \( a \) is \( m \)-bit, such that \( a > a_1 + \ldots + a_n \)
- \( w \) is relatively prime with \( m \)
- One-round Knapsack was broken by A. Shamir in 1982
- Several variations of Knapsack were broken
Others

- Elliptic Curve Cryptography (ECC)
- Zero Knowledge Proof Systems

Building Security Services

- Confidentiality:
  - Use an encryption algorithm
  - Generally a symmetric algorithm
- Integrity:
  - MAC algorithm
- Access control:
  - Use access control tables
- Authentication
  - Use authentication protocols
- Non-repudiation
  - Digital signatures