# Cryptography Review



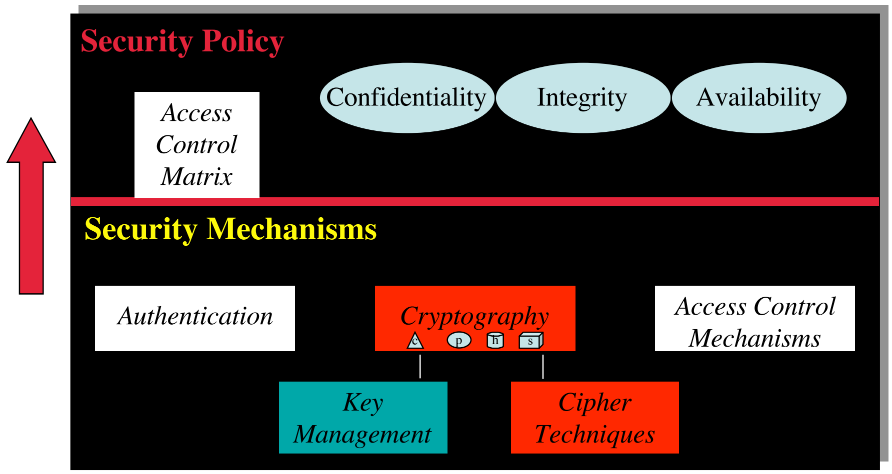
# Quiz 1

(Three minutes each)

- How can a TCP SYN attack cause a denial of service?
- How can an attacker detect what TCP-based services a server machine may be providing?

*Copyright © 2003 Jun Li.*
*All rights reserved.*

# Where Are We When Talking Cryptography?






# Cryptography

- Goal: keep enciphered info secret
  - A deep mathematical subject
- Usage: a cornerstone for secure communication
- Assumptions: attackers know the algorithm but not the key(s)
- Types: classical cryptosystems and public key cryptosystems

# Four Main Topics Covered

- Classical cryptography
- Public key cryptography
- Cryptographic checksum function
- Digital signature



# Definitions

- **Cryptography**: the art and science of concealing information
- **Cryptoanalysis**: code breaking
- **Cryptosystem**: basic component of cryptography
  - $(\mathcal{E}, \mathcal{D}, \mathcal{M}, \mathcal{K}, \mathcal{C})$
  - $\mathcal{M}$ : plaintexts
  - $\mathcal{K}$: keys
  - $\mathcal{C}$: ciphertexts
  - $\mathcal{E}$ : enciphering functions $\mathcal{M} \times \mathcal{K} \to \mathcal{C}$
  - $\mathcal{D}$ : deciphering functions $\mathcal{C} \times \mathcal{K} \to \mathcal{M}$

*Copyright © 2003 Jun Li.*
*All rights reserved.*

# Classical Cryptosystems

- Same key for encipherment and decipherment
  - Also called **single-key** cryptosystem
  - Or **symmetric** cryptosystem
- For all $\mathcal{E}_k \in \mathcal{E}$ there is $\mathcal{D}_k \in \mathcal{D}$ such that

$$\mathcal{D}_k = \mathcal{E}_k^{-1}$$

- Examples:
  - Transposition cipher
  - Substitution cipher
    - Vigenere cipher, One-time pad, etc.
  - DES: the combination of both



# Transposition Cipher

- Characters in plaintext are rearranged
  - Letters unchanged
- *Rail fence* cipher, as an example
  - "UNIV  OF  OREGON" becomes
    "UI O OEONV F RGN" or "UVFRON  ENIOOG"

```
UI O OEO      UVFRO
NV F RGN      N  EN
              IOOG
```

# Substitution Cipher

- Characters are changed
  - Caesar cipher for example, where letters are simply shifted
- Examples:
  - Vigenere cipher
  - One-time pad



# Vigenere Cipher

- Use a longer key to obscure the statistics
- The length of a key is called the **period** of the cipher
- A *tableau* is used to implement cipher
  - Table lookup for encipherment

| | |
|---|---|
| Key | `B ENCHBENC HBENC HBENCH` |
| Plaintext | `A LIMERICK PACKS LAUGHS` |
| Ciphertext | `B PVOLSMPM WBGXU SBYTJZ` |

# One-Time Pad

- A variant of the Vigenere cipher
- But key string is randomly chosen and *at least as long the message*!
  - No repetition
- Impossible to break! **Perfect secrecy :)**
  - Impossible to deploy either. :(



# DES: Data Encryption Standard

- A classical cryptosystem
- Bit-level
- Uses both transposition and substitution
  - Also referred as **product cipher**
- Encipherment unit: 64-bit blocks
  - Input, output and keys are all in 64b blocks

## AES: Advanced Encryption Standard

- DES is no longer as secure as designed in its early days
- 2001. NIST selects **Rijndael** as AES.



## Public Key Cryptography

- Use two different keys for encryption and decryption
- An entity has two keys: **a public key** and **a private key**
  - Hard to derive the private key from the public key
- Examples:
  - Diffie-Hellman
  - RSA
  - . . . . .

## Properties of Public Key

- Assuming *x* has a public key *e* and a private key *d*
- Message encrypted with *e* can **only** be decrypted **by *x*** using *d*
  - Useful to send an encrypted message to *x*
- If a message can be decrypted with *e*, then it must be encrypted **by *x*** using *d*
  - Useful to verify whether or not a message is from *x*



## Combine Confidentiality and Authentication

- For confidentiality, the message has to be encrypted with B's public key
  - So that B's private key has to be used to decrypt
  - But only B knows B's private key
- For origin authentication, the message has to be encrypted with A's private key
  - So that A's public key has to be used to decrypt
  - Everybody knows A's public key
- Question: can we switch the two above?

# Cryptographic Checksums

- Motivating question: How can Bob verify messages received from Alice is not changed?
- Answer: *digital signature*
  - Which relies on **cryptographic checksum function**
  - Digital signature will be covered later
- Cryptographic checksum function also has many other usages
  - Such as S/Key protocol (used in Authentication)



# Cryptographic Checksum Function

- Also called **strong hash function**
  - Or **strong one-way function**
- $h: A \rightarrow B$
  - For any $x \in$ A, $h(x)$ is easy to compute
  - For any y $\in$ B, computationally infeasible to find $x \in$ A such that $h(x) = y$
  - No collision pairs

# Prevention of Collision Pairs

- Statement A:
  - Computationally infeasible to find $x, x' \in$ A such that

    $$x \neq x' \text{ but } h(x) = h(x')$$

- Statement B:
  - Given any $x \in$ A, computationally infeasible to find another $x' \in$ A such that

    $$x \neq x' \text{ but } h(x) = h(x')$$

- Statement B is much harder than statement A.



# Keyed or Keyless Cryptographic Checksum

- A **keyed cryptographic checksum** requires a cryptographic key as part of hashing computation
  - E.g. DES-MAC
    - DES is in CBC mode (covered later)
    - Use last enciphered block output as the hash result
    - DES needs a key
- A **keyless cryptographic checksum** does not
  - MD2, MD4, MD5
  - SHA-1 (Secure Hash Algorithm)
  - Snefru
  - HAVAL

# Digital Signature

- A **digital signature** is a construct that authenticates both the ***origin*** and ***contents*** of a message in a manner that is provable to a disinterested third party.

- Provides a service of nonrepudiation



# Classical Signature

Let Cathy be a trusted third party

Alice shared a secret key $k_{A,C}$ with Cathy

Bob shared a secret key $k_{B,C}$ with Cathy

1. Alice → Bob: $\{m\}\ k_{A,C}$
2. Bob → Cathy: $\{m\}\ k_{A,C}$

    Cathy deciphers with $k_{A,C}$ and re-enciphers with $k_{B,C}$

3. Cathy → Bob: $\{m\}\ k_{B,C}$

    Bob then gets $m$

# Classical Signature Verification

- Verification question: is ***m*** created by Alice?

- Verification method:
  - Judge Takes the disputed messages $\{m\}k_{A,C}$ & $\{\{m\}\ k_{B,C}$
  - Ask Cathy to decrypt $\{m\}k_{A,C}$ using $k_{A,C}$ and $\{m\}k_{B,C}$ using $k_{B,C}$
  - And compare

$$\{\{m\}k_{A,C}\}\ k_{A,C} = \{\{m\}\ k_{B,C}\}\ k_{B,C}\ ?$$



# Public Key Signature

- Instead of using $\{m\}\ d_{Alice}$, Alice actually signs the message as

$$\{h(m)\}\ d_{Alice}$$

  where ***h*** is a cryptographic hash function
- And sends Bob

$$m\ \{h(m)\}\ d_{Alice}$$

- Q: how does Bob verifies the signature?

## Cipher Techniques

- Cipher techniques must be used wisely
  - Very sensitive to the environment
- A mathematically strong cryptosystem is vulnerable when used incorrectly
  - Examples include: precomputing the possible messages, misordered blocks, and statistical regularities.
- So we introduced block cipher and stream cipher, and try to strengthen both!



## Examples of Incorrect Cryptosystem Usage

- Precomputing the possible messages
- Misordered Blocks
- Statistical Regularities

# Precomputing Possible Messages

- Simmon's Attack: "*Forward search*" technique
- Alice will send Bob one of two messages: BUY or SELL, enciphered with $e_{Bob}$
  - Eve does not which one, but
  - Eve knows it's one of the two
- Eve precomputes the {"BUY"} $e_{Bob}$ and {"SELL"} $e_{Bob}$
- When Alice sends Bob a message, Eve intercept it and compare with the precomputed ciphertext
  - **Then Eve knows what's the plaintext!**

- Problem: the set of plaintext if small



# Misordered Blocks

- Denning: part of ciphertext can be deleted, replayed, or reordered

LIVE → 44 57 21 16 → 16 21 57 44 → EVIL

LIVE → 44 57 21 16 → 44 57 16 → LIE

- Each part can be signed
  - But if signed separately, will it work?

- Problem: each part is encrypted independently

## Statistical Regularities

- When each part is enciphered separately, the same plaintext will produce the same ciphertext
  - Thus regularity arises
  - Making cryptanalysis easy
- This type of encipherment is called **code book mode**



## So . . .

- How to use cipher techniques?

- Block Cipher
- Stream Cipher

# Block Cipher

- $E$: an encipherment algorithm
- $E_k(b)$: encipherment of msg $b$ with key $\boldsymbol{k}$
- Message $m = b_1 b_2 \ldots$,
  - where each $b_i$ is of fixed length
- **Block cipher :** $E_k(m) = E_k(b_1)\ E_k(b_2)\ \ldots$
- Q: is DES a block cipher?



# Block Cipher (cont'd)

- Multiple bits each time
  - Faster than stream cipher in software implementations
- But an identical plaintext block will produce an identical ciphertext block
  - If using the same key

*Copyright © 2003 Jun Li.*
*All rights reserved.*

# Strengthening Block Cipher

1. Insert extra bits into a block, often related to block position
   – Sequence number of a block
   – Bits from preceding ciphertext block
2. **Cipher block chaining** (**CBC**)
   – $c_0 = E_k(m_0 \oplus I)$
   – $c_i = E_k(\ m_i \oplus c_{i-1}\ )$ for $i > 0$



# (cont'd)

3. **Encrypt-Decrypt-Encrypt** (**EDE**)

$$c = E_k(D_{k'}(E_k(m)))$$

4. **Triple Encryption Mode**

$$c = E_k(E_{k'}(E_{k''}(m)))$$

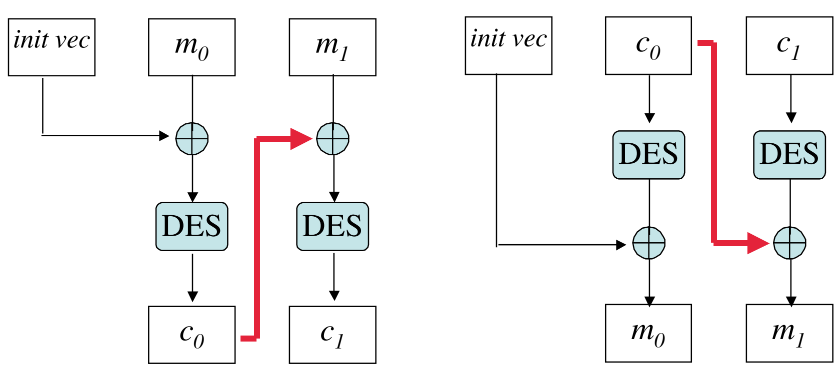
– Consider applying CBC, EDE, or triple Encryption to DES!

# Three Common Modes of DES

- CBC : Cipher Block Chaining
- EDE: Encrypt-Decrypt-Encrypt
- Triple DES: DES-DES-DES



# CBC




Encipherment

Decipherment

# EDE

- Two 64-bit keys: $k$ and $k'$

$$c = DES_k(DES_{k'}^{-1}(DES_k(m)))$$



# Triple DES

- Three 64-bit keys: $k$, $k'$, and $k''$

$$c = DES_k(DES_{k'}(DES_{k''}(m)))$$

# Stream Cipher

- $E$: an encipherment algorithm
- $E_k(b)$: encipherment of msg $b$ with key $\boldsymbol{k}$
- Message $m = b_1 b_2 \ldots$, Key $k = k_1 k_2 \ldots$,
  - where each $b_i$ is of fixed length

- **Stream cipher :** $E_k(m) = E_{k1}(b_1)\ E_{k2}(b_2) \ldots$
- Q: is Vigenere a stream cipher?
  - Yes, and also a **periodic stream cipher**



# Types of Stream Ciphers

- Two types, depending on how keys are generated:
  - Synchronous stream cipher
  - Self-synchronous stream cipher

# Synchronous Stream Ciphers

- Generates bits of a key from **a particular source**
  - Not from the message itself
  - Hopefully the newly generated key is random and long
- Several techniques
  - LFSR (Linear feedback shift register)
  - NLFSR (Nonlinear feedback shift register)
  - Output feedback mode
  - Counter method



# LFSR (linear feedback shift register)

- An $n$-bit register $r = r_{n-1} \ldots r_0$ (a variable)
- An $n$-bit tap sequence $t = t_{n-1} \ldots t_0$ (a constant)
- Use $r_0$ as current key bit
- Right shift $r$, and $r_{n-1} = (r_{n-1} \bullet t_{n-1}) \oplus \ldots \oplus (r_0 \bullet t_0)$

| $t$=**1001** | current reg | key | new $r_{n-1}$ bit | new reg |
|---|---|---|---|---|
| | 0010 | **0** | 01⊕00⊕10⊕01=**0** | **0**001 |
| | 0001 | **1** | 01⊕00⊕00⊕11=**1** | **1**000 |
| | 1000 | | | |

The key stream can have a period of **$2^n$-1** (maximal value)

# NLFSR (nonlinear feedback shift register)

- New bit is a function of current register bits
  - No tap sequence used

| current reg | key | new $r_{n-1}$ bit | new reg |
|---|---|---|---|
| 0010 | 0 | $f(0,0,1,0)=0$ | 0001 |
| 0001 | 1 | $f(0,0,0,1)=0$ | 0000 |
| 0000 | | | |

$f = r_3$ or ($r_2$ and $r_0$)



# Output Feedback Mode

$m$: the message to encrypt

$E$: encipherment function

$k$: a cryptography key

$r$: a register

- $r = E_k(r)$
- $k_i = r_0$ ($r$'s rightmost bit)
- $c_i = m_i \oplus k_i$

# Counter Method

$m$: the message to encrypt

$E$: encipherment function

$k$: a cryptography key

$i_0$: initial value of a counter

- $k_i = (i+i_0)$'s rightmost bit (*for* $i$=0, 1, 2, . . . )
- $c_i = m_i \oplus k_i$



# Self-Synchronous Stream Ciphers

- Generate a key from the message itself
  - Could be from plaintext, could be from ciphertext
  - Also called **autokey cipher**

```
Key           XTHEBOYHASTHEBA
Plaintext     THEBOYHASTHEBAG
Ciphertext    QALFPNFHSLALFCT

Key           XQXBCQOVVNGNRTT
Plaintext     THEBOYHASTHECAT
Ciphertext    QXBCQOVVNGNRTTM
```

# (cont'd)

- If using plaintext, key selection is an issue
  - Key will display same statistical regularities as it's derived from plaintext
- If using ciphertext, weak
  - A character in ciphertext = *f* (*X*, a previous chacter in ciphertext)



# Cipher Feedback Mode

*m*: the message to encrypt

*E*: encipherment function

*k*: a cryptography key

*r*: a register

- $x = E_k(r)$
- $r = x_{n-1}\, r_{n-1} \ldots r_1$
- $c_i = m_i \oplus x_0$ ($x_0$ is $x$'s rightmost bit)

# Authentication

- Authentication is the binding of an identity to a subject, which is acting on behalf of an entity
  - Or, the binding of an identity to an entity

- How?
  - What the entity knows (e.g.passwords)
  - What the entity has (e.g. a badge)
  - What the entity is (e.g. fingerprints)
  - Where the entity is (e.g. in front of a particular terminal)
  - . . . .



# Authentication Process

- Obtain authentication info from an entity
- Analyze the info
- Determine whether or not the info is associated with the entity

- For the purpose of analysis, the entity's info must be stored and managed
  - An authentication system

# Authentication System

- ***A***: the set of authentication info with which entities prove their identities
- ***C***: the set of complementary info that the system stores and uses to validate the authentication info
- ***F***: the set of complementation functions that for $f \in F$, $f: A \rightarrow C$
- ***L***: the set of authentication functions that for $l \in L$, $l: A \times C \rightarrow$ { **true**, **false**}
- ***S***: the set of selection functions that enable an entity to create/alter authentication and complementary info



# Authentication Systems

- Password
- Challenge-Response
  - One-time password
    - S/Key
  - Hardware-supported challenge-response
- Biometrics
- Location
- Etc.

*Copyright © 2003 Jun Li.*
*All rights reserved.*

# Passwords

- A **password** is information associated with an entity that confirms the entity's identity
  - Simplest example: some sequence of characters
  - e.g., *login, su,* etc. in Unix
- *C* may not be the same as *A*
  - Mostly because *C* must be protected
  - e.g., `/etc/passwd` (or shadow password files) in Unix
- *F*
  - $f \in F$ is based upon DES in Unix
- *S*
  - e.g., `passwd` command in Unix



# Challenge-Response

- Fundamental flaw of passwords: reusability
  - Can be replayed if known before
  - What if every time one uses different authentication information
- In a challenge-response authentication system
  - User *U* and System *S* share a secret function ***f***
  - *S* sends a random message m (**challenge**)
  - *U* replies with $r=f(m)$ (**response**)
  - *S* validates *r* by computing it separately

# One-Time Password

- **One-time password**: a password that is invalidated as soon as it is used
- Also a challenge-response mechanism
  - Challenge: the number of authentication attempt
  - Response: the one-time password



# S/Key

- $h$: a one-way hash function
- $k$: an initial seed chosen by the user

keys: $h(k)=k_1$, $h(k_1)=k_2$, . . . , $h(k_{n-1})=k_n$
passwds: $p_1=k_n$, $p_2 = k_{n-1}$, . . ., $p_{n-1} = k_2$, $p_n = k_1$

If Eve intercepts $p_i$ , we know $\mathbf{p_i = h(p_{i+1})}$, and $h$ is a one-way hash function, so $p_{i+1}$ cannot be derived from $p_i$.

## S/Key Authentication Protocol

- User Matt supplies his name to the server
- The server replies with the number $i$ stored in the *skeykeys* file
- Matt supplies password $p_i$
- Server computes $h(p_i)$ and compares it with the stored password $p_{i-1}$. If match,
  - Authentication succeeds
  - $i \leftarrow i+1$, $p_{i-1} \leftarrow p_i$



## Hardware-Supported Challenge-Response Procedures

- *Token* device
  - System sends a challenge
  - User enters it into the device (PIN maybe needed)
  - The device returns a response, by hashing (or enciphering) the challenge
  - The user sends the response over

## (cont'd)

- Temporally based device
  - Every 60 seconds, a different number displayed
  - The system knows what number to be displayed for a user
  - When the user logs in, he enters the number currently shown
    - Followed by a fixed password
  - e.g., RSA SecureID card



## Biometrics

- As old as humanity
- Fingerprints
- Voices
- Eyes
- Faces
- Keystrokes
- Combinations

# Location

- Anna is logging from Russia
  - But we know she is now working at California
- Dennis and MacDoran's scheme: use Global Positioning System (GPS)
  - An entity obtains a **location signature** using GPS
  - Transmits it
  - The System uses a **location signature sensor** (**LSS**) to obtain a similar location signature
  - Compare the two signatures to authenticate