CS530
Cryptography
Bill Cheng

http://merlot.usc.edu/cs530-s10
Cryptography & Security

- Cryptography underlies many fundamental services
  - Confidentiality
  - Data integrity
  - Authentication

- Cryptography is *the* basic foundation of much of security
A Brief History

**Steganography:** "covered writing"
- Demaratus (5th century B.C.)
  - writing under wax on tablets
- German microdots (WWII)
- crucial flaw: Discovery yields knowledge
  - confidentiality through obscurity
- covert channels
  - Ex: timing channel

**Cryptography:** "secret writing"
- TASOIINRPSTO and TVCTUJUVUJPO
A Brief History (Cont...)

Two basic types of cryptography

- **Transposition** (TASOIINRNPSTO) or permutation
  - message broken up into units
  - units permuted in a seemingly random but reversible manner
  - Ex: wrap tape on rod
  - difficult to make it easily reversible only by intended receiver
  - exhibits same first-order (or mono-gram) statistics (but distort di-grams, tri-grams, etc.)

- **Substitution** (TVCTUJUVUJPO)
  - (cont...)
A Brief History (Cont...)

- Substitution (TVCTUJUVUJPO)
  - message broken up into units
  - units mapped into ciphertext
  - Ex: Caesar cipher
  - first-order statistics are isomorphic in simplest cases
    - note: for transposition, first-order statistics are identical
  - predominant form of encryption
How Much Security?

Monoalphabetic substitution cipher
- permutation on message units: letters
- 26! different permutations
- each permutation considered a key
- key space contains 26! = 4 \times 10^{26} keys
  - equal to number of atoms in a gallon of water
  - equivalent to a 88-bit key (more than DES!)

So why not use substitution ciphers?
- hard to remember 26-letter keys
  - but we can restrict ourselves to shorter keys
  - Ex: JULISCAERBDFGHKM, etc.
- remember: first-order statistics are isomorphic
  - vulnerable to simple cryptanalysis
1964 English Language Statistics

**frequency of single characters in English text**

<table>
<thead>
<tr>
<th>Character</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8.04</td>
</tr>
<tr>
<td>B</td>
<td>3.06</td>
</tr>
<tr>
<td>C</td>
<td>3.99</td>
</tr>
<tr>
<td>D</td>
<td>2.30</td>
</tr>
<tr>
<td>E</td>
<td>1.96</td>
</tr>
<tr>
<td>F</td>
<td>5.49</td>
</tr>
<tr>
<td>G</td>
<td>7.26</td>
</tr>
<tr>
<td>H</td>
<td>4.14</td>
</tr>
<tr>
<td>I</td>
<td>2.53</td>
</tr>
<tr>
<td>J</td>
<td>2.00</td>
</tr>
<tr>
<td>K</td>
<td>0.11</td>
</tr>
<tr>
<td>L</td>
<td>6.12</td>
</tr>
<tr>
<td>M</td>
<td>6.54</td>
</tr>
<tr>
<td>N</td>
<td>9.25</td>
</tr>
<tr>
<td>O</td>
<td>2.71</td>
</tr>
<tr>
<td>P</td>
<td>1.92</td>
</tr>
<tr>
<td>Q</td>
<td>1.73</td>
</tr>
<tr>
<td>R</td>
<td>0.99</td>
</tr>
<tr>
<td>S</td>
<td>0.19</td>
</tr>
<tr>
<td>T</td>
<td>0.09</td>
</tr>
<tr>
<td>U</td>
<td>1.28</td>
</tr>
<tr>
<td>V</td>
<td>1.28</td>
</tr>
<tr>
<td>W</td>
<td>1.28</td>
</tr>
<tr>
<td>X</td>
<td>1.28</td>
</tr>
<tr>
<td>Y</td>
<td>1.28</td>
</tr>
<tr>
<td>Z</td>
<td>1.28</td>
</tr>
</tbody>
</table>

**frequency of 15 common digrams in English text (27% overall)**

<table>
<thead>
<tr>
<th>Digram</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN</td>
<td>1.81</td>
</tr>
<tr>
<td>AT</td>
<td>1.51</td>
</tr>
<tr>
<td>ED</td>
<td>1.32</td>
</tr>
<tr>
<td>EN</td>
<td>1.52</td>
</tr>
<tr>
<td>ER</td>
<td>1.36</td>
</tr>
<tr>
<td>ES</td>
<td>3.05</td>
</tr>
<tr>
<td>HE</td>
<td>2.30</td>
</tr>
<tr>
<td>IN</td>
<td>1.83</td>
</tr>
<tr>
<td>ON</td>
<td>1.28</td>
</tr>
<tr>
<td>OR</td>
<td>1.90</td>
</tr>
<tr>
<td>RE</td>
<td>1.22</td>
</tr>
<tr>
<td>ST</td>
<td>1.30</td>
</tr>
<tr>
<td>TE</td>
<td>1.28</td>
</tr>
<tr>
<td>TH</td>
<td>3.21</td>
</tr>
<tr>
<td>TI</td>
<td>1.28</td>
</tr>
</tbody>
</table>
Substitution Ciphers

Two basic types

- *symmetric-key* or conventional
  - single key used for both encryption and decryption
  - keys are typically short, because key space is densely filled
  - Ex: DES, 3DES, RC4, Blowfish, IDEA, etc

- *public-key* or asymmetric
  - two keys: one for encryption, one for decryption
  - keys are typically long, because key space is sparsely filled
  - Ex: RSA, El Gamal, DSA, etc
Conventional Cryptography

Stream cipher

- **stream cipher**: generates a (random or pseudorandom) keystream and applies it to a stream of plaintext with XOR
  - good for applications such as telnet
  - Ex: RC4
- **one-time pad**: if the keystream is truly randomly chosen and never used again, the stream cipher is a one-time pad
  - the one-time pad can be shown to be *theoretically unbreakable*
/* state information */
static uns8 state[256], x, y;

void rc4init(uns8 *key, uns16 length) {
    /* initialization */
    int i;
    uns8 t, j, k=0;
    for (i=256; i--; ) state[i] = i;
    for (i=0, j=0;
        i < 256;
        i++, j=(j+1)%length) {
        t = state[i];
        state[i] =
        state[k+= key[j] + t];
        state[k] = t;
    }
    x = y = 0;
}

uns8 rc4step() {
    /*
     * return next
     * pseudo-random
     * octet
     */
    uns8 t;
    t = state[y += state[++x]];
    state[y] = state[x];
    state[x] = t;
    return state[
        state[x]+state[y]
    ];
}
RC4 (Cont...)  

To generate a random byte, do:

\[
\begin{align*}
  i &:= 0 \\
  j &:= 0 \\
  \text{while GeneratingOutput:} \\
  &\quad i := (i + 1) \mod 256 \\
  &\quad j := (j + S[i]) \mod 256 \\
  &\quad \text{swap}(S[i], S[j]) \\
  &\quad \text{output } S[(S[i] + S[j]) \mod 256]
\end{align*}
\]

Key scheduling algorithm:

\[
\begin{align*}
  \text{for } i \text{ from 0 to 255} \\
  &\quad S[i] := i \\
  j &:= 0 \\
  \text{for } i \text{ from 0 to 255} \\
  &\quad j := (j + S[i] + \text{key}[i \mod l]) \mod 256 \\
  &\quad \text{swap}(S[i], S[j])
\end{align*}
\]
Conventional Cryptography (Cont...)

Block ciphers encrypt message in units called blocks

- DES: 8-byte key (56 key bits), 8-byte block
  - $2^{56} \approx 10^{17}$
  - Note: $2^{56} = 10^X$
    - $56 \log 2 = X \log 10$
    - $X = 56 \log 2 / \log 10$

- larger blocks make simple cryptanalysis useless (at least for short messages)
  - not enough samples for valid statistics
  - "octo-gram statistics needed"
Key and Block Size

Do larger keys make sense for an 8-byte block?

- 3DES: Key is 112 or 168 bits, but block is still 8 bytes long (64 bits)
- key space is larger than block space
- Q: how many possible keys are out there?
- A: equal to the size of the permutation space
- why?
  - each key can be think of as a way to map an input pattern to an output pattern
  - Q: how many different patterns are there?
  - A: $2^{64}$
  - remember, must be one-to-one mapping
- but how large is permutation space?
  - $2^{64} \times = ?$
  - use Stirling’s Formula: $n! \approx n^n e^{-n} \sqrt{2\pi n}$
CS530
Cryptanalysis
Bill Cheng

http://merlot.usc.edu/cs530-s10
Cryptanalysis

Cryptanalysis is the study of mathematical techniques for attempting to defeat cryptographic techniques and information security services

- a cryptanalyst is someone who engages in cryptanalysis

Cryptology is the study of cryptography and cryptanalysis

Six general types of cryptanalytic attacks:

- ciphertext-only attack
- known-plaintext attack
- chosen-plaintext attack
- adaptive-chosen-plaintext attack
- chosen-ciphertext attack
- adaptive-chosen-ciphertext attack

Another type of cryptanalytic attack

- purchase-key attack
Cryptanalytic Attacks

- **Ciphertext-only attack**
  - given ciphertexts
  - deduce plaintexts (or key)

- **Known-plaintext attack**
  - given plaintext-ciphertext pairs
  - deduce key

- **Chosen-plaintext attack**
  - the cryptanalyst has access to the encryption device
  - given plaintext→ciphertext pairs of the attacker’s choosing
  - deduce key

- **Adaptive-chosen-plaintext attack**
  - special case of a chosen-plaintext attack
  - the cryptanalyst can modify his choice based on the result of previous encryption
Cryptanalytic Attacks (Cont...) 

- **Chosen-ciphertext attack**
  - the cryptanalyst has access to the *decryption* device
  - given ciphertext → plaintext pairs of the attacker’s choosing
  - deduce key

- **Adaptive-chosen-ciphertext attack**
  - special case of a chosen-ciphertext attack
  - the cryptanalyst can modify his choice based on the result of previous decryption

- **Rubber-hose cryptanalysis** (or *purchase-key attack*)
  - the cryptanalyst threatens, blackmails, or tortures someone until they give him the key
  - often the best way to break an algorithm!
Attack on Protocols

- Until a protocol is proven to provide the service intended, the list of possible attacks can never be said to be complete
  - **known-key attack**: an adversary obtains some keys used previously to determine new keys
  - **replay attack**: an adversary records a communication session and replays the entire session, or a portion thereof, at some later point in time
  - **impersonation attack**: an adversary assumes the identity of one of the legitimate parties in a network
  - **dictionary attack**:
    - usually an attack against passwords
    - password is stored as image of unkeyed hash function
    - an adversary takes a list of possible passwords, hash all entries and compare with stored hash values
Attack on Protocols (Cont...)

- **forward search attack**: similar to the dictionary attack and is used to decrypt messages
- **interleaving attack**: involves some form of impersonation in an authentication protocol
  - A and B executes a security protocol
  - an adversary intercepts all messages and sends its own messages
  - man-in-the-middle attack
Finding Ethernet Address: Address Resolution (ARP)

Broadcast: who knows the Ethernet address for 128.125.51.41?

Broadcast: I do, it is 08-00-2c-19-dc-45
Man-in-the-middle Attack

- ARP weaknesses
  - accept additional ARP responses even if an ARP response has already been received
  - overwrite cached ARP value
  - accept ARP response even if no ARP request
  - fixes for this exist, but often not implemented or installed

- Man-in-the-middle attack
  - an attacker on LAN and send an ARP response to a host H to impersonate the gateway G
  - and tells G that its ethernet address is that of H
  - now the attacker can intercept and modify any packet that goes between H and G
CS530
DES
Bill Cheng

http://merlot.usc.edu/cs530-s10
Anatomy Of A Block Cipher

- DES: Data Encryption Standard
  - developed as Lucifer (one of a few) at IBM in 1970s
  - break message into 8-byte (64-bit) blocks
    - each block broken into 32-bit halves
    - initial permutation
    - 16 rounds of scrambling
    - final (reverse) permutation
  - Feistel Network structure
The Scrambling Function

In each round $i$, we have $L_i$ and $R_i$
- $L_{i+1} = R_i$ ← typical of Feistel networks
- $R_{i+1} = L_i \oplus f(R_i)$

f-function
- key is compressed and permuted to 48 bits (called subkeys, different for each round)
- $R_i$ (32-bits) is expanded and permuted to 48 bits

- 48 bits XOR’d, passed through S-boxes (to produce 32 bits), then permuted again
- irreversible
Feistel Network

**Encryption**

- 64-bit input
- 32-bit $L_n$
- 32-bit $R_n$
- $f$
- $K_n$
- 32-bit $L_{n+1}$
- 32-bit $R_{n+1}$
- 64-bit output

**Decryption**

- 64-bit output
- 32-bit $L_n$
- 32-bit $R_n$
- $f$
- $K_n$
- 32-bit $L_{n+1}$
- 32-bit $R_{n+1}$

→ $f$ only used in only one direction for both encryption and decryption
DES

**Twisted Ladder**
- **Input**: $m_1m_2 \ldots m_{63}m_{64}$
- **IP**: $64$
- **Ladder**
  - $L_0 \rightarrow R_0$
  - $L_1 \rightarrow R_1$
  - $L_{15} \rightarrow R_{15}$
- **Irregular Swap**: $R_{16} \rightarrow L_{16}$
- **IP^-1**: $64$
- **Output**: $c_1c_2 \ldots c_{63}c_{64}$

**Untwisted Ladder**
- **Input**: $64$
- **IP**: $64$
- **Ladder**
  - $L_0 \rightarrow R_0$
  - $R_1 \rightarrow k_1$
  - $L_2 \rightarrow k_2$
  - $R_3 \rightarrow k_3$
  - $L_4 \rightarrow k_4$
  - $R_{15} \rightarrow k_{16}$
- **IP^-1**: $64$
- **Output**: $c_1c_2 \ldots c_{63}c_{64}$
Key Compression

Reduction to 56 bits (no parity bits)

DES key (64 bits);

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

odd parity bits: 

Broken into halves

- each half is rotated by 1 or 2 bits
- 48 bits out of 56 selected

Why do this?

- use a different set of bits for each round
- not exactly symmetric
Data Expansion

- Data broken into 4-bit groups
- Each group expanded to 6 bits
- Why do this?
  - match subkey length
  - data diffusion occurs faster
Substitution Boxes (S-Boxes)

- This is the heart of DES
- 48 bit result broken into 6-bit units
- Each unit passed through an S-box
  - 6-bit input, 4-bit output
  - each S-box is a 4x16 array of 4-bit numbers
  - b1 and b6 specify row, b2 through b5 specify column
- End result passed through P-box
DES Properties

Desirable characteristics for a block cipher

- each bit of the ciphertext should depend on all bits of the key and all bits of the plaintext
- there should be no statistical relationship evident between plaintext and ciphertext
- altering any single plaintext or key bit should alter each ciphertext bit with probability $1/2$
- altering a ciphertext bit should result in unpredictable change to the recovered plaintext block

Empirically, DES satisfies all the above objectives
DES Weak Keys

- If generated subkeys are such that $k_1 = k_{16}$, $k_2 = k_{15}$, and so on, the encryption and decryption functions coincide
  - these are called weak keys (and also palindromic keys)
    - if K is a weak key, then $E_K(E_K(x)) = x$ for all $x$

- DES also has semi-weak keys
  - if $(K_1,K_2)$ is pair of semi-weak keys, then $E_{K_1}(E_{K_2}(x)) = x$ for all $x$

- DES has 4 weak keys and 6 pairs of semi-weak keys

<table>
<thead>
<tr>
<th>semi-weak keys (hexadecimal)</th>
<th>weak keys (hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01FE 01FE 01FE 01FE, FE01 FE01 FE01 FE01</td>
<td>0101 0101 0101 0101</td>
</tr>
<tr>
<td>1FE0 1FE0 0EF1 0EF1, E01F E01F F10E F10E</td>
<td>FFEF FFEF FFEF FFEF</td>
</tr>
<tr>
<td>01E0 01E0 01F1 01F1, E001 E001 F101 F101</td>
<td>1F1F 1F1F 1F1F 1F1F</td>
</tr>
<tr>
<td>1FFE 1FFE 0EFE 0EFE, EF1F EF1F EF0E EF0E</td>
<td>E0E0 E0E0 E0E0 E0E0</td>
</tr>
<tr>
<td>011F 011F 010E 010E, 1F01 1F01 0E01 0E01</td>
<td>E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1</td>
</tr>
<tr>
<td>E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1</td>
<td></td>
</tr>
</tbody>
</table>
Modes of DES Operation

What to do if message is longer than 8 bytes?

Electronic Codebook (ECB)
- each block encrypted in isolation
- vulnerable to block replay (same input $\Rightarrow$ same output)

\[
\begin{align*}
\text{Encryption with secret key} \\
n_1 &\xrightarrow{E} c_1 \\
n_2 &\xrightarrow{E} c_2 \\
n_3 &\xrightarrow{E} c_3 \\
n_4 &\xrightarrow{E} c_4 \\
n_5 &\xrightarrow{E} c_5 \\
n_6 &\xrightarrow{E} c_6
\end{align*}
\]

Cipher Block Chaining (CBC)
- each plaintext block XOR’d with previous ciphertext before encryption
- easily incorporated into decryption
- what if prefix is always the same? IV!
Cipher Block Chaining (CBC)

Encryption with secret key:

\[ m_1 \rightarrow E \rightarrow c_1 \]
\[ m_2 \rightarrow E \rightarrow c_2 \]
\[ m_3 \rightarrow E \rightarrow c_3 \]
\[ m_4 \rightarrow E \rightarrow c_4 \]
\[ m_5 \rightarrow E \rightarrow c_5 \]
\[ m_6 \rightarrow E \rightarrow c_6 \]

Decryption with secret key:

\[ c_1 \rightarrow D \rightarrow m_1 \]
\[ c_2 \rightarrow D \rightarrow m_2 \]
\[ c_3 \rightarrow D \rightarrow m_3 \]
\[ c_4 \rightarrow D \rightarrow m_4 \]
\[ c_5 \rightarrow D \rightarrow m_5 \]
\[ c_6 \rightarrow D \rightarrow m_6 \]
Modes of DES Operation (Cont...)

Stream cipher
- *stream cipher*: generates a (random or pseudorandom) keystream and applies it to a stream of plaintext with XOR
- *one-time pad*: if the keystream is truly randomly chosen and never used again, the stream cipher is a one-time pad

Cipher Feedback (CFB)
- for encrypting character-at-a-time (or less)
- chains as in CBC
- also needs an IV
  - must be unique

Output Feedback (OFB)
- like CFB, but some bits of output fed back into input stream
OFB vs CFB

**OFB (simplified):** \( v_i = E(k, v_{i-1}) \) and \( c_i = m_i \oplus v_i \)

**CFB (simplified):** \( c_i = m_i \oplus E(k, c_{i-1}) \)
OFB vs CFB (Cont...)

OFB (simplified)

Alternate view:
DES Variants and Applications

- Crypt: Unix hash function for passwords
  - uses variable expansion permutations
  - add a 12-bit salt (to modify DES)
    - to mitigate the precomputed dictionary attack
  - encrypt the number 0

- DES with key-dependent S-boxes
  - cannot be done blindly
Variants and Applications (Cont...)

3DES: Encrypt using DES 3x
- two and three-key types

encryption:
\[ m \rightarrow E \rightarrow D \rightarrow E \rightarrow c \]

decryption:
\[ c \rightarrow D \rightarrow E \rightarrow D \rightarrow m \]

- inner and outer-CBC modes

CBC on the outside:
Variants and Applications (Cont...)

inner and outer-CBC modes (cont...)

CBC on the inside:

\[
\begin{align*}
\text{m}_1 & \rightarrow E \rightarrow C_1 \\
\text{m}_2 & \rightarrow E \rightarrow C_2 \\
\text{m}_3 & \rightarrow E \rightarrow C_3 \\
\text{m}_4 & \rightarrow E \rightarrow C_4 \\
\text{m}_5 & \rightarrow E \rightarrow C_5 \\
\text{m}_6 & \rightarrow E \rightarrow C_6
\end{align*}
\]
Variants and Applications (Cont...)

3DES (cont...)
- inner-CBC mode for 3DES is more efficient, but less secure
  - more efficient because of possible pipelining
  - under some attacks inner-CBC mode is significantly weaker than outer-CBC mode; against other attacks based on block size, inner-CBC mode appears stronger (please note that this is different from what the textbook says)
- main reason for EDE is backwards compatibility with single-key DES

Why not 2DES? (EE, DE, or ED)
- (cont...)
Why not 2DES? (EE, DE, or ED)
- turns out 2DES is not much more secure than DES
- meet-in-the-middle attack

\[ K_1 \rightarrow E \rightarrow D \rightarrow C \]

\[ K_2 \rightarrow \]

\[ D_0 \rightarrow E_0 \]
\[ D_1 \rightarrow E_1 \]
\[ D_2 \rightarrow E_2 \]
\[ D_{2^{56}-1} \rightarrow E_{2^{56}-1} \]

Matching!
Attacks on DES

No known systematic attack (for 16 rounds)

- is DES "closed" (that is, a group)?
  - does there exist a $K_3$ such that $E_{K_1}(E_{K_2}(P)) = E_{K_3}(P)$
  - if it were, double encryption would be useless
  - DES is not closed

- is DES "pure"?
  - does there exist a $K_4$ such that $E_{K_1}(E_{K_2}(E_{K_3}(P))) = E_{K_4}(P)$
  - if it were, triple encryption would be useless
  - unfortunately, don’t know if DES is pure

- does DES has a skeleton/allpass key?
  - not likely because DES is symmetric

Brute force attacks only

- try all $2^{56}$ keys!
Lucifer Goes Standard

Lucifer is one of the IBM ciphers
- generally regarded in 1970s as one of the strongest cryptosystems

Heading toward standardization as DES
- NSA managed to get key size reduced to 56 bits (from 64), yielding $10^{17}$ keys
- also apparently changed S-boxes
- why (or why not) do this?
  - NSA does not trust IBM
  - who trusts NSA?
Certification of DES

Had to be recertified every ~5 years

- 1983: Recertified routinely
- 1987: Recertified after NSA tried to promote secret replacement algorithms
  - withdrawal would mean lack of protection
  - lots of systems then using DES
- 1993: Recertified after continued lack of alternative
CS530
AES
Bill Cheng

http://merlot.usc.edu/cs530-s10
Enter AES

1997: Call for candidates for Advanced Encryption Standard (AES)
- fifteen candidates whittled down to five
- criteria: Security, but also efficiency
  - compare Rijndael with Serpent (which is generally regarded as more secure but less efficient)
- 2000: Rijndael selected as AES
Structure of Rijndael

 Unlike DES, operates on whole bytes for efficiency of software implementations

 Key sizes: 128/192/256 bits

 Variable rounds: 9/11/13 rounds

 Rounds are not Feistel networks

 Round structure

 - run block through S-box (8x32)
 - permute result into 4x4/4x6/4x8 array of bytes
 - multiply each byte by 1, 2, or 3 in GF($2^8$)
   - addition in GF($2^8$) is done through XOR
 - mix subkey into result

 Addition in GF($2^8$) is done through XOR.
Security of Rijndael

- Based on arithmatic in GF(2^8)
- Key size is enough
- Immune to linear or differential analysis
- But Rijndael is a very structured cipher
  - S-box consists of byte reciprocals in GF(2^8)
    - finite field \( \mathbb{Z}_2[x] / (x^8 + x^4 + x^3 + x + 1) \)
    - permutations are regular
- Attack on Rijndael’s algebraic structure
  - breaking can be modeled as equations
    - only need to know a single plaintext/cipher text pair
    - \(~8,000\) quadratic equations with \(~1,600\) variables
      (also in GF(2^8))
Impact of Attacks on Rijndael

- Currently of theoretical interest only
  - reduces complexity of attack to about $2^{100}$ (after the system of quadratic equations are solved)
  - also applicable to Serpent (complexity is about $2^{200}$)

- Still, uncomfortably close to feasibility
  - DES is already insecure against brute force
  - Schneier (somewhat arbitrarily) sets limit at $2^{80}$

- Certainly usable pending further results