Cryptography and Network Security

Block Cipher

Xiang-Yang Li

Modern Private Key Ciphers

- Stream ciphers
  - The most famous: Vernam cipher
  - Invented by Vernam, (AT&T, in 1917)
  - Process the message bit by bit (as a stream)
  - (Also known as the one-time pad)
  - Simply add bits of message to random key bits

Cont.

Plaintext  Key  Ciphertext  Key  Ciphertext

Pros and Cons

- Drawbacks
  - Need as many key bits as message, difficult in practice
  - (i.e., distribute on a magnetic tape or CD-ROM)

- Strength
  - Is unconditionally secure provided key is truly random

Key Generation

- Why not to generate keystream from a smaller (base) key?
  - Use some pseudo-random function to do this
  - Although this looks very attractive, it proves to be very difficult in practice to find a good pseudo-random function that is cryptographically strong

- This is still an area of much research

Block Ciphers

- The message is broken into blocks,
  - Each of which is then encrypted
  - (Like a substitution on very big characters - 64-bits or more)
Substitution and Permutation

- In his 1949 paper Shannon also introduced the idea of substitution-permutation (S-P) networks, which now form the basis of modern block ciphers
  - An S-P network is the modern form of a substitution-transposition product cipher
  - S-P networks are based on the two primitive cryptographic operations we have seen before

Substitution

- A binary word is replaced by some other binary word
- The whole substitution function forms the key
  - If use n bit words,
    - The key space is $2^n$
- Can also think of this as a large lookup table, with n address lines (hence $2^n$ addresses), each n bits wide being the output value
- Will call them s-boxes

Permutation

- A binary word has its bits reordered (permuted)
- The re-ordering forms the key
  - If use n bit words,
    - The key space is $n!$ (Less secure than substitution)
- This is equivalent to a wire-crossing in practice
  - (Though is much harder to do in software)
- Will call these p-boxes

Substitution-permutation Network

- Shannon combined these two primitives
- He called these mixing transformations
- A special form of product ciphers where
  - S-boxes
    - Provide confusion of input bits
  - P-boxes
    - Provide diffusion across s-box inputs
Confusion and Diffusion

- **Confusion**
  - A technique that seeks to make the relationship between the statistics of the ciphertext and the value of the encryption keys as complex as possible. Cipher uses key and plaintext.

- **Diffusion**
  - A technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext digit over many ciphertext digits.

Desired Effect

- **Avalanche effect**
  - A characteristic of an encryption algorithm in which a small change in the plaintext gives rise to a large change in the ciphertext.
  - Best: changing one input bit results in changes of approx half the output bits.

- **Completeness effect**
  - Where each output bit is a complex function of all the input bits.

Practical Substitution-permutation Networks

- **In practice we need to be able to decrypt messages, as well as to encrypt them, hence either:**
  - Have to define inverses for each of our S & P-boxes, but this doubles the code/hardware needed, or
  - Define a structure that is easy to reverse, so can use basically the same code or hardware for both encryption and decryption.

Feistel Cipher

- **Invented by Horst Feistel,** working at IBM Thomas J Watson research labs in early 70’s,

- The idea is to partition the input block into two halves, \(\text{l(i-1)}\) and \(\text{r(i-1)}\),
  - Use only \(\text{r(i-1)}\) in each round \(i\) (part) of the cipher
  - The function \(g\) incorporates one stage of the S-P network, controlled by part of the key \(k(i)\) known as the \(i\)th subkey

- **This can be described functionally as:**
  - \(L(i) = R(i-1)\)
  - \(R(i) = L(i) \oplus g(k(i), R(i-1))\)

- This can easily be reversed as seen in the above diagram, working backwards through the rounds.

- In practice link a number of these stages together (typically 16 rounds) to form the full cipher.
Data Encryption Standard

- Adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology
- Data are encrypted in 64-bit blocks using a 56-bit key
- The same algorithm is used for decryption.
- Subject to much controversy

History

- IBM LUCIFER 60’s
  - Uses 128 bits key
- Proposal for NBS, 1973
- Adopted by NBS, 1977
  - Uses only 56 bits key
  - Possible brute force attack
  - Design of S-boxes was classified
  - Hidden weak points in S-Boxes?
- Wiener (93) claim to be able to build a machine at $100,000 and break DES in 1.5 days

DES

- DES encrypts 64-bit blocks of data, using a 56-bit key
- The basic process consists of:
  - an initial permutation (IP)
  - 16 rounds of a complex key dependent calculation f
  - a final permutation, being the inverse of IP
- Function f can be described as:
  - \( L(i) = R(i-1) \)
  - \( R(i) = L(i-1) \oplus P(S( E(R(i-1)) \oplus K(i))) \)

Initial and Final Permutations

- Inverse Permutations

<table>
<thead>
<tr>
<th>40</th>
<th>48</th>
<th>16</th>
<th>64</th>
<th>52</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>47</td>
<td>15</td>
<td>55</td>
<td>33</td>
</tr>
<tr>
<td>38</td>
<td>46</td>
<td>14</td>
<td>54</td>
<td>32</td>
</tr>
<tr>
<td>37</td>
<td>45</td>
<td>13</td>
<td>53</td>
<td>31</td>
</tr>
<tr>
<td>36</td>
<td>44</td>
<td>12</td>
<td>52</td>
<td>30</td>
</tr>
<tr>
<td>35</td>
<td>43</td>
<td>11</td>
<td>51</td>
<td>29</td>
</tr>
<tr>
<td>34</td>
<td>42</td>
<td>10</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>33</td>
<td>41</td>
<td>9</td>
<td>49</td>
<td>27</td>
</tr>
</tbody>
</table>

Function f
Expansion Table

- Expands the 32 bit data to 48 bits
  - Result(i) = input array(i)

<table>
<thead>
<tr>
<th>i</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
</tr>
</tbody>
</table>

S-Boxes

- S-Box is a fixed 4 by 16 array
- Given 6-bits B = b1 b2 b3 b4 b5 b6,
  - Row r = b1 b6
  - Column c = b2 b3 b4 b5
  - S(B) = S(r, c) written in binary of length 4

Example

S-Box S1

<table>
<thead>
<tr>
<th>i</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Permutation Table

- The permutation after each round

<table>
<thead>
<tr>
<th>Round</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
</tr>
</tbody>
</table>

Subkey Generation

- Given a 64 bits key (with parity-check bit)
  - Discard the parity-check bits
  - Permute the remaining bits using fixed table P1
  - Let C0 D0 be the result (total 56 bits)
- Let Ci = Sh(θ1(Ci-1)), Di = Sh(θ2(Di-1)) and Ki be another permutation P2 of C0 D0 (total 56 bits)
  - Where cyclic shift one position left if i = 1, 2, 9, 16
  - Else cyclic shift two positions left

Permutation Tables

<table>
<thead>
<tr>
<th>Round</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
</tbody>
</table>

Permutation table P1

<table>
<thead>
<tr>
<th>Round</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
</tbody>
</table>
DES in Practice
- DEC (Digital Equipment Corp. 1992) built a chip with 50k transistors
  - Encrypt at the rate of 1G/second
  - Clock rate 250 Mhz
  - Cost about $300
- Applications
  - ATM transactions (encrypting PIN and so on)

Model
- Mode of use
  - The way we use a block cipher
  - Four have been defined for the DES by ANSI in the standard: ANSI X3.106-1983 modes of use
- Block modes
  - Splits messages in blocks (ECB, CBC)
- Stream modes
  - On bit stream messages (CFB, OFB)

Block Modes
- Electronic Codebook Book (ECB)
  - where the message is broken into independent 64-bit blocks which are encrypted
  - $C_i = \text{DES}_k (P_i)$
- Cipher Block Chaining (CBC)
  - again the message is broken into 64-bit blocks, but they are linked together in the encryption operation with an IV
  - $C_i = \text{DES}_k (P_i \oplus C_{i-1})$
  - $C_{i-1} = \text{IV} (\text{initial value})$

Stream Model
- Cipher Feedback (CFB)
  - where the message is treated as a stream of bits, added to the output of the DES, with the result being feedback for the next stage
  - $C_i = P_i \oplus \text{DES}_k (C_{i-1})$
  - $C_{i-1} = \text{IV} (\text{initial value})$

Cont.
- Output Feedback (OFB)
  - where the message is treated as a stream of bits, added to the message, but with the feedback being independent of the message
  - $C_i = P_i \oplus O_i$
  - $O_i = \text{DES}_k (O_{i-1})$
  - $O_{i-1} = \text{IV} (\text{initial value})$

DES Weak Keys
- With many block ciphers there are some keys that should be avoided, because of reduced cipher complexity
- These keys are such that the same sub-key is generated in more than one round, and they include:
Cont.

- **Weak keys**
  - The same sub-key is generated for every round
  - DES has 4 weak keys

- **Semi-weak keys**
  - Only two sub-keys are generated on alternate rounds
  - DES has 12 of these (in 6 pairs)

- **Demi-semi weak keys**
  - Have four sub-keys generated

None of these causes a problem since they are a tiny fraction of all available keys.
However they MUST be avoided by any key generation program.

Possible Techniques for Improving DES

- Multiple enciphering with DES
- Extending DES to 128-bit data paths and 112-bit keys
- Extending the key expansion calculation

Double DES?

- Using two encryption stages and two keys
  - \( C = E_{k_2}(E_{k_1}(P)) \)
  - \( P = D_{k_1}(D_{k_2}(C)) \)
- It is proved that there is no key \( k_3 \) such that
  - \( C = E_{k_2}(E_{k_1}(P)) = E_{k_3}(P) \)
- But Meet-in-the-middle attack

Meet-in-the-Middle Attack

- Assume \( C = E_{k_2}(E_{k_1}(P)) \)
- Given the plaintext \( P \) and ciphertext \( C \)
- Encrypt \( P \) using all possible keys \( k_1 \)
- Decrypt \( C \) using all possible keys \( k_2 \)
  - Check the result with the encrypted plaintext lists
  - If found match, they test the found keys again for another plaintext and ciphertext pair
  - If it turns correct, then find the keys
  - Otherwise keep decrypting \( C \)

Triple DES

- DES variant
- Standardized in ANSI X9.17 & ISO 8732 and in PEM for key management
- Proposed for general EFT standard by ANSI X9
- Backwards compatible with many DES schemes
- Uses 2 or 3 keys
IDEA:
- Developed by James Massey & Xuejia Lai at ETH originally in Zurich in 1990, then called IPES:
  - X Lai, J L Massey, S Murphy, "Markov Ciphers and Differential Cryptanalysis" in Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science, vol 547, pp 17-38,
  - name changed to IDEA in 1992

Basic Features
- Encrypts 64-bit blocks using a 128-bit key
- Based on mixing operations from different (incompatible) algebraic groups
  - XOR, + mod 2^16, X mod 2^16 + 1
  - On 16-bit sub-blocks, with no permutations used
- IDEA is patented in Europe & US, however non-commercial use is freely permitted
  - used in the public domain PGP (with agreement)
  - currently no attack against IDEA is known
    - Seem secure against differential cryptanalysis, brute force

Overview
- IDEA encryption works as follows:
  - Use 8 rounds
  - The 64-bit data is divided into: X₁, X₂, X₃, X₄
  - Each round
    - The sub-blocks are added (2,3), multiplied (1,4) with sub-keys
    - The results are XORed with input of MA structure,
      - It outputs two subblocks
      - Results are then XORed with 2,4 and 1,3 sub-blocks respectively
      - The second and third sub-blocks are swapped
    - Finally new sub-keys are combined with the sub-blocks

Sub-Keys
- Total need 52=6×8+4 sub-keys
  - First are directly from key in order
  - Left shift of 25 bits, and then next 8 sub-keys
  - Each sub-key is a sub-block of the original key

Decryption
- Much more complicated
  - It needs the inverse of the encryption key
    - For addition, multiplication
Decryption

- The process of decryption is essentially the same as encryption.
  - But with different selection of sub-keys.
- Basic Operations:
  - $K1.1^{-1}$ is the multiplicative inverse mod $2^{16}+1$.
  - $-K1.2$ is the additive inverse mod $2^{16}$.
  - The original operations are:
    - (x) bit-by-bit XOR.
    - + additional mod $2^{16}$ of 16-bit integers.
    - * multiplication mod $2^{16}+1$ (where 0 means $2^{16}$).

Decryption Sub-Keys

<table>
<thead>
<tr>
<th>Round</th>
<th>Encryption Keys</th>
<th>Decryption Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>K1.1 K1.2 K1.3 K1.4 K1.5 K1.6</td>
<td>K9.1 K9.2 K9.3 K9.4 K9.5 K9.6</td>
</tr>
<tr>
<td>2</td>
<td>K2.1 K2.2 K2.3 K2.4 K2.5 K2.6</td>
<td>K8.1 K8.2 K8.3 K8.4 K8.5 K8.6</td>
</tr>
<tr>
<td>3</td>
<td>K3.1 K3.2 K3.3 K3.4 K3.5 K3.6</td>
<td>K7.1 K7.2 K7.3 K7.4 K7.5 K7.6</td>
</tr>
<tr>
<td>4</td>
<td>K4.1 K4.2 K4.3 K4.4 K4.5 K4.6</td>
<td>K6.1 K6.2 K6.3 K6.4 K6.5 K6.6</td>
</tr>
<tr>
<td>5</td>
<td>K5.1 K5.2 K5.3 K5.4 K5.5 K5.6</td>
<td>K5.1 K5.2 K5.3 K5.4 K5.5 K5.6</td>
</tr>
<tr>
<td>6</td>
<td>K6.1 K6.2 K6.3 K6.4 K6.5 K6.6</td>
<td>K4.1 K4.2 K4.3 K4.4 K4.5 K4.6</td>
</tr>
<tr>
<td>7</td>
<td>K7.1 K7.2 K7.3 K7.4 K7.5 K7.6</td>
<td>K3.1 K3.2 K3.3 K3.4 K3.5 K3.6</td>
</tr>
<tr>
<td>8</td>
<td>K8.1 K8.2 K8.3 K8.4 K8.5 K8.6</td>
<td>K2.1 K2.2 K2.3 K2.4 K2.5 K2.6</td>
</tr>
<tr>
<td></td>
<td>Output K9.1 K9.2 K9.3 K9.4</td>
<td>K1.1 K1.2 K1.3 K1.4 K1.5 K1.6</td>
</tr>
</tbody>
</table>

CAST-128

- By Carlisle Adams, Stafford Tavares
  - Defined in RFC 2144.
  - Use key size varying from 40 to 128 bits.
  - Structure of Feistel network.
  - 16 rounds on 64-bit data block.
  - Four primitive operations:
    - Addition, subtraction (mod $2^{32}$).
    - Bitwise exclusive-OR.
    - Left-circular rotation.

Skipjack and Clipper

- Skipjack:
  - Used in Clipper escrowed encryption scheme (US govt).
  - Skipjack is a block cipher, 64-bit data.
  - Hardware only implementation.
  - 80-bit key (escrowed in 2 halves).
  - 32 round.
  - All design details and descriptions are classified.
  - Has been very considerable debate over its use.
  - Attack by Matt Blaze (ATT) on the LEAF component of the Clipper protocol for secure phone communications.

Blowfish Scheme

- Developed by Bruce Schneier.
  - Fast, compact, simple and variably secure.
  - Two basic operations: addition, XOR.
  - Key ranges from 32 bits to 448 bits.
  - Similar to Feistel scheme.
  - The sub-key and s-boxes are complicated.
  - So not suitable when key changes often.
  - Function g is very simple, unlike DES.

RC5

- Developed by R. Rivest.
  - Suitable for hardware or software.
  - Fast, simple, low memory, data-dependent rotations.
  - Adaptable to processors of different word length.
    - A family of algorithms determined by word length, number of rounds, size of secret key.
  - Decryption and encryption are not the same.
  - With little variations.
  - Primitive operations:
    - Addition, XOR, left circular rotation.
Characteristics

- Key features of advanced sym block cipher
  - Variable key length
  - Mixed operators
  - Data dependent rotation
  - Key dependent rotation
  - Key dependent S-boxes
  - Lengthy key schedule algorithm
  - Variable function $F$
  - Variable of number of rounds
  - Operation on both halved data each round