Practical Constructions of Block Ciphers

Reading: K&L Section 6.2 (skipping 6.2.6)
Practical constructions of block ciphers

- There are methods to construct pseudorandom permutations from one-way functions.
  - One-way functions $\Rightarrow$ pseudorandom generators
    $\Rightarrow$ pseudorandom functions
    $\Rightarrow$ pseudorandom permutations
  - Extremely slow

- In practice, block ciphers are constructed using
  - Feistel networks (e.g., DES)
  - Substitution-permutation networks (e.g., AES)

- Block ciphers: "approximate" pseudorandom permutations with some fixed key length and block length.
The confusion-diffusion paradigm

- Introduced by Shannon. Suppose we want to design a 128-bit (keyed) random-looking permutation $F$.
- First, design an 8-bit (keyed) random-looking permutation $f$.
- To compute $F_k(x)$:
  - Divide the input block $x$ into sixteen 8-bit blocks $x_1, \ldots, x_{16}$.
  - Use the key $k$ to specify 16 permutations $f_{k_1}, \ldots, f_{k_{16}}$.
    (Derive a round key $\langle k_1, \ldots, k_{16} \rangle$ from the master key $k$.)
  - Let $x' = f_{k_1}(x_1) \parallel \cdots \parallel f_{k_{16}}(x_{16})$ (confusion).
  - Permute the 128 bits of $x'$ (diffusion).
  - Repeat the process several times (rounds).
Substitution-permutation networks

- A direct implementation of the confusion-diffusion paradigm.
- Hard to design a (keyed) random-looking permutation $f$.
- Design 16 (unkeyed, fixed) 8-bit permutations $f_1, \ldots, f_{16}$, which are called S-boxes and denoted by $S_1, \ldots, S_{16}$.
- To compute $F_k(x)$:
  - Divide the input block $x$ into 8-bit blocks $x_1, \ldots, x_{16}$.
  - Derive a round key $\langle k_1, \ldots, k_{16} \rangle$ from the master key $k$.
  - Let $x' = S_1(x_1 \oplus k_1) \| \cdots \| S_{16}(x_{16} \oplus k_{16})$ (key-mixing & substitution).
  - Permute the 128 bits of $x'$ (permutation).
  - Repeat the process several times (rounds), followed by a final key-mixing.
Substitution-permutation network

Key-mixing

Substitution

Permutation
In practice, all rounds use the same set of boxes, say \{S_1, S_2, S_3, S_4\}. 
Feistel Networks and Data Encryption Standard (DES)
Feistel Network/Cipher

- Proposed by Feistel (in 1970s). Suppose we want to design an $\ell$-bit (keyed) random-looking permutation $F$.
- First, design an $\ell/2$-bit (keyed) random-looking function $f$, which is not necessarily invertible.
- To compute $F_k(x)$:
  - Divide the input block $x$ into two halves $L$ and $R$.
  - Derive a round key $k_i$ (for round $i$) from master key $k$.
  - Let $x' = R \parallel L \oplus f_{k_i}(R)$.
  - Repeat the process several times (rounds).
  - (Typically there is a final swap of $L$ and $R$.)
Round $i$

If $\oplus k_i$ is not invertible

$\begin{align*}
L' & \quad R' \\
\downarrow & \quad \downarrow \\
L & \quad R
\end{align*}$
The Feistel Network Structure

Note: Read $F$ as $f$. 
Feistel Network/Cipher (Mathematical Description)

- Let $L_i$ and $R_i$ denote the output half-blocks of the $i$th round.
- So $L_{i-1}$ and $R_{i-1}$ are the input of the $i$th round.
- We have

  $$L_i := R_{i-1}$$
  $$R_i := L_{i-1} \oplus f_{k_i}(R_{i-1})$$

- The $i$th round can be viewed as a composite function $\mu \circ \phi_i$

  $$\phi_i : (x, y) \rightarrow (x \oplus f_{k_i}(y), y), \text{ where } x, y \text{ are half-blocks.}$$
  $$\mu : (x, y) \rightarrow (y, x).$$

- Note that $\phi_i^{-1} = \phi_i$ and $\mu^{-1} = \mu.$
• Assume 16 rounds.

• A Feistel cipher with key $k$ and input block $x$ will output:

$$y = F_k(x) = \mu \circ \mu \circ \phi_{16} \circ \cdots \circ \mu \circ \phi_2 \circ \mu \circ \phi_1(x)$$

• The inverse $F_k^{-1}(y)$ will be:

$$F_k^{-1}(y) = \phi_1^{-1} \circ \mu^{-1} \circ \phi_2^{-1} \circ \cdots \circ \mu^{-1} \circ \phi_{16}^{-1} \circ \mu^{-1} \circ \mu^{-1}(y)$$

$$= \mu \circ \mu \circ \phi_1 \circ \mu \circ \phi_2 \circ \cdots \circ \mu \circ \phi_{16}(y)$$

• $F_k^{-1}$ is the same as $F_k$, but uses the round keys in the reverse order.
DES: The Data Encryption Standard

- Once most widely used block cipher in the world.
- Adopted by NIST in 1977.
- Based on the Feistel cipher structure with 16 rounds of processing.
- Block = 64 bits
- Key = 56 bits
- What is specific to DES is the design of the $f$ function and how the round keys are derived from the main key.
Design Principles of DES

- To achieve high degree of **confusion** and **diffusion**.

- Confusion: making the relationship between the encryption key and the ciphertext as complex as possible.

- Diffusion: making each plaintext bit affect as many ciphertext bits as possible.
DES Encryption Overview

1. **Initial Permutation**
   - 64-bit plaintext

2. **Round 1**
   - $K_1$: 48 bits
   - Permutated Choice 2
   - Left circular shift

3. **Round 2**
   - $K_2$: 48 bits
   - Permutated Choice 2
   - Left circular shift

4. **Round 16**
   - $K_{16}$: 48 bits
   - Permutated Choice 2
   - Left circular shift

5. **32-bit Swap**
   - 64 bits

6. **Inverse Initial Permutation**
   - 64-bit ciphertext
Round Key Generation

- Main key: 64 bits, but only 56 bits are used.
- 16 round keys (48 bits each) are generated from the main key by a sequence of permutations.
- Select and permute 56-bits using Permuted Choice One (PC1). Then divide them into two 28-bit halves.
- At each round:
  - Rotate each half separately by either 1 or 2 bits according to a rotation schedule.
  - Select 24-bits from each half & permute them (48 bits) by PC2.
  - This forms a round key.
### Permutated Choice One (PC1)

<table>
<thead>
<tr>
<th>57</th>
<th>49</th>
<th>41</th>
<th>33</th>
<th>25</th>
<th>17</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58</td>
<td>50</td>
<td>42</td>
<td>34</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>59</td>
<td>51</td>
<td>43</td>
<td>35</td>
<td>27</td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>3</td>
<td>60</td>
<td>52</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>63</td>
<td>55</td>
<td>47</td>
<td>39</td>
<td>31</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>62</td>
<td>54</td>
<td>46</td>
<td>38</td>
<td>30</td>
<td>22</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>61</td>
<td>53</td>
<td>45</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>13</td>
<td>5</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>
DES Round Structure

- $L$ & $R$ each has 32 bits.
- As in any Feistel cipher:
  
  $L_i := R_{i-1}$
  
  $R_i := L_{i-1} \oplus f_{k_i}(R_{i-1})$

- $f$ takes 32-bit $R$ and 48-bit round key $k_i$:
  - expands $R$ to 48-bits using expansion perm $E$
  - adds to the round key using XOR
  - shrinks to 32-bits using 8 $S$-boxes
  - finally permutes using 32-bit perm $P$
The DES $f$ function

The diagram shows the process of the DES $f$ function, which involves the following steps:

1. The input $R$ (32 bits) is first expanded to 48 bits using the expansion function $E$.
2. The 48-bit expanded value is then XORed with the 48-bit key $K$.
3. The result of the XOR operation is then passed through a series of substitution boxes labeled $S_1$ to $S_8$.
4. Finally, the output is permuted by the permutation function $P$ to produce the 32-bit output.
## The E Expansion Permutation

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>
The S-Boxes

- Eight S-boxes each map 6 to 4 bits
- Each S-box is a 4 x 16 table
  - each row is a permutation of 0-15
  - outer bits 1 & 6 of input are used to select one of the four rows/permutations
  - inner 4 bits of input are used to select a column
- All the eight boxes are different.
### Box $S_1$

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

- For example, $S_1(101010) = 6 = 0110$. 
P-Permutation
Avalanche Effect

- **Avalanche effect**: a key desirable property of any encryption algorithm:
  - A small change in the plaintext or in the key results in a significant change in the ciphertext.
  - (an evidence of high degree of diffusion and confusion)

- DES exhibits a strong avalanche effect
  - Changing 1 bit in the plaintext affects 34 bits in the ciphertext on average.
  - 1-bit change in the key affects 35 bits in the ciphertext on average.
Attacks on DES

- Brute-force key search
  - Needs only two plaintext-ciphertext samples
  - Trying 1 key per microsecond would take 1000+ years on average, due to the large key space size, \(2^{56} \approx 7.2 \times 10^{16}\).

- Differential cryptanalysis
  - Possible to find a key with \(2^{47}\) plaintext-ciphertext samples
  - Known-plaintext attack

- Linear cryptanalysis:
  - Possible to find a key with \(2^{43}\) plaintext-ciphertext samples
  - Known-plaintext attack
Attacks on DES

• **DES Cracker:**
  – A DES key search machine
  – containing 1536 chips
  – could search 88 billion keys per second
  – won RSA Laboratory’s **DES Challenge II-2** by successfully finding a DES key in 56 hours.
  – Cost: $250,000

• The vulnerability of DES is due to **its short key length**.

• Remedy: 3DES
Multiple Encryption with DES

• In 2001, NIST published the Advanced Encryption Standard (AES) to replace DES.

• But users in commerce and finance are not ready to give up on DES.

• As a temporary solution to DES’s security problem, one may encrypt a message (with DES) multiple times using multiple keys:
  – 2DES is not much securer than the regular DES
  – So, 3DES with either 2 or 3 keys is used
2DES

- Use two DES keys, say $k_1$, $k_2$.

- Encryption: $c := Enc_{k_2}(Enc_{k_1}(m))$

- Key length: $56 \times 2 = 112$ bits

- Would this thwart brute-force attacks?
Meet-in-the-Middle Attack on 2DES

\[ m \rightarrow \text{Enc}_{k_1} \rightarrow \text{Enc}_{k_2} \rightarrow c \]

- Given a known pair \((m, c)\), attack as follows:
  - Encrypt \(m\) with all \(2^{56}\) possible keys for \(k_1\).
  - Decrypt \(c\) with all \(2^{56}\) possible keys for \(k_2\).
  - Find two keys \(\tilde{k}_1, \tilde{k}_2\) such that \(\text{Enc}_{\tilde{k}_1}(m) = \text{Dec}_{\tilde{k}_2}(c)\).
  - Try \(\tilde{k}_1, \tilde{k}_2\) on another pair \((m', c')\): Is \(\text{Enc}_{\tilde{k}_1}(m') = \text{Dec}_{\tilde{k}_2}(c')\)?
  - If works, \((\tilde{k}_1, \tilde{k}_2) = (k_1, k_2)\) with high probability.
  - Takes \(\Theta \left(2^{56}\right)\) steps, not much more than attacking 1-DES.

- It is a known-plaintext attack.
3DES with 2 keys

- A straightforward implementation would be:
  \[ c := Enc_{k_1} \left( Enc_{k_2} \left( Enc_{k_1}(m) \right) \right) \]

- In practice: \[ c := Enc_{k_1} \left( Dec_{k_2} \left( Enc_{k_1}(m) \right) \right) \]
  - Also referred to as EDE encryption
  - Reason: if \( k_1 = k_2 \), then 3DES = 1DES.
    Thus, a 3DES software can be used as a single-DES.
  - No practical attacks are known.
  - Not recommended: key size 112 bits is shorter than the current minimum recommendation of 128 bits.
3DES with 3 keys

- Encryption: \( c := \text{Enc}_{k_3}\left(\text{Dec}_{k_2}\left(\text{Enc}_{k_1}(m)\right)\right) \).
- If \( k_1 = k_3 \), it becomes 3DES with 2 keys.
- If \( k_1 = k_2 = k_3 \), it becomes the regular DES.
- So, it is backward compatible with both 3DES with 2 keys and the regular DES.
- Some internet applications adopt 3DES with three keys, e.g. PGP and S/MIME.
AES: Advanced Encryption Standard

Finite field: The mathematics used in AES.
AES: Advanced Encryption Standard

• In 1997, NIST began the process of choosing a replacement for DES and called it the Advanced Encryption Standard.
• Requirements: block length of 128 bits, key lengths of 128, 192, and 256 bits.
• In 2000, Rijndael cipher (by Rijmen and Daemen) was selected.
• An iterated cipher, with 10, 12, or 14 rounds.
• Rijndael allows various block lengths.
• AES allows only one block size: 128 bits.
Structure of Rijndael

- $N_b$: block size (number of words). For AES, $N_b = 4$.
- $N_k$: key length (number of words).
- $N_r$: number of rounds, depending on $N_b$, $N_k$.
- Assume: $N_b = 4$, $N_k = 4$, $N_r = 10$.
- *state*: a variable of 4 words, holding the data block, viewed as a $4 \times 4$ matrix of bytes; each column is a word.
- Key schedule: $N_r + 1$ round keys $key_0, key_1, \ldots, key_{10}$ are computed from the main key $k$. 
**Rijndael algorithm** (input: plaintext $m$, key $k$)

1. $state \leftarrow m$
2. $\text{AddKey}(state, key_0)$
3. for $i \leftarrow 1$ to $N_r - 1$ do
   4. $\text{SubBytes}(state)$
   5. $\text{ShiftRows}(state)$
   6. $\text{MixColumns}(state)$
   7. $\text{AddKey}(state, key_i)$
8. $\text{SubBytes}(state)$
9. $\text{ShiftRows}(state)$
10. $\text{AddKey}(state, key_{N_r})$
11. return($state$)
AddKey($state, \ key_i$)

\[ state \leftarrow state \oplus key_i \]
SubBytes($state$)

- Each byte $z$ in $state$ is substituted with another byte according to a table.
ShiftRows\( (state) \)

- Left-shift row \( i \) circularly by \( i \) bytes, \( 0 \leq i \leq 3 \).

\[
\begin{pmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
m & n & o & p \\
\end{pmatrix} \rightarrow \begin{pmatrix}
a & b & c & d \\
f & g & h & e \\
k & l & i & j \\
p & m & n & o \\
\end{pmatrix}
\]
**MixColumns**(state)

- Operates on each column of the *state* matrix.
- View each column \( a = (a_0, a_1, a_2, a_3) \) as a polynomial with coefficients in \( \text{GF}(2^8) \):
  \[
a(x) = a_3x^3 + a_2x^2 + a_1x + a_0
  \]
- A fixed polynomial: \( c(x) = 03x^3 + 01x^2 + 01x + 02 \).
- The MixColumns operation maps each column
  \[
a(x) \mapsto a(x) \cdot c(x) \mod (x^4 + 1)
  \]
Rijndael Decryption

- Each step of Rijndael encryption is invertible.
Rijndael key schedule

- Round keys are derived from the main key
A Rijndael Animation by Enrique Zabala