Advanced Encryption Standard (AES)

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University
History

- Clear a replacement for DES was needed
  - Have theoretical attacks that can break it
  - Have demonstrated exhaustive key search attacks
  - Block size small
  - Can use Triple-DES – but slow

- US NIST issued call for ciphers in 1997
  - 15 candidates accepted in Jun 98
  - 5 were shortlisted in Aug-99
  - Rijndael was selected as the AES in Oct 2000
  - Issued as FIPS PUB 197 standard in Nov 2001
AES Requirements

- Symmetric key block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Active life of 20-30 years (+ archival use)
- Provide full specification & design details
- Both C & Java implementations
- NIST have released all submissions & unclassified analyses
AES Evaluation Criteria

- **Initial criteria:**
  - Security – randomness, soundness, effort for practical cryptanalysis
  - Cost – computational efficiency, no licensing fee, small memory
  - Algorithm & implementation characteristics – flexibility, implementable in both software and hardware, simplicity

- **Final criteria**
  - General security – NIST relies on the cryptanalysis by crypto researchers
  - Ease of software & hardware implementation
  - Implementation attacks – finding keys based on implementation characteristics
    - Timing attacks: an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
    - Power analysis: the power consumed by a smart card at any particular time during the cryptographic operation is related to the instruction being executed and to the data being processed. For example, multiplication consumes more power than addition, and writing 1s consumes more power than writing 0s
  - Flexibility (encryption, decryption, keying, and other factors)
AES Shortlist

After testing and evaluation, shortlist in Aug 99:
- MARS (IBM) - complex, fast, high security margin
- RC6 (USA) - very simple, very fast, low security margin
- Rijndael (Belgium) - clean, fast, good security margin
- Serpent (Euro) - slow, clean, very high security margin
- Twofish (USA) - complex, very fast, high security margin

Then subject to further analysis & comment

Saw contrast between algorithms with
- Few complex rounds vs. many simple rounds
- which refined existing ciphers vs. new proposals
The AES Cipher - Rijndael

- Designed by Rijmen-Daemen in Belgium
- An iterative rather than Feistel Cipher
  - Processes data as block of 4 columns of 4 bytes
  - Operates on entire data block in every round
- Designed to be:
  - Resistant against known attacks
  - Speed and code compactness on many CPUs
  - Design simplicity
# AES Parameters

<table>
<thead>
<tr>
<th></th>
<th>4/16/128</th>
<th>6/24/192</th>
<th>8/32/256</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Size (words/bytes/bits)</strong></td>
<td>4/16/128</td>
<td>6/24/192</td>
<td>8/32/256</td>
</tr>
<tr>
<td><strong>Plaintext Block Size (words/bytes/bits)</strong></td>
<td>4/16/128</td>
<td>4/16/128</td>
<td>4/16/128</td>
</tr>
<tr>
<td><strong>Number of Rounds</strong></td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td><strong>Round Key Size (words/bytes/bits)</strong></td>
<td>4/16/128</td>
<td>4/16/128</td>
<td>4/16/128</td>
</tr>
<tr>
<td><strong>Expanded Key Size (words/bytes)</strong></td>
<td>44/176</td>
<td>52/208</td>
<td>60/240</td>
</tr>
</tbody>
</table>
# Advanced Encryption Standard (AES)

## Input

<table>
<thead>
<tr>
<th>128 bit plaintext</th>
<th>128 bit cipher key</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 88 31 e0</td>
<td>$k_0$ $k_4$ $k_8$ $k_{12}$</td>
</tr>
<tr>
<td>43 5a 31 37</td>
<td>$k_1$ $k_5$ $k_9$ $k_{13}$</td>
</tr>
<tr>
<td>f6 30 98 07</td>
<td>$k_2$ $k_6$ $k_{10}$ $k_{14}$</td>
</tr>
<tr>
<td>a8 8d a2 34</td>
<td>$k_3$ $k_7$ $k_{11}$ $k_{15}$</td>
</tr>
</tbody>
</table>

**Encryption Progress**

**Key Expansion**
Encryption Progress

- Encryption Progress
  - Substitute Bytes
  - Shift Rows
  - Mix Columns
  - Add Round Key

Initial round

9 rounds

final rounds
## Encryption Progress – Substitute Bytes

<table>
<thead>
<tr>
<th>hex</th>
<th>y</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>f2</td>
<td>6b</td>
<td>6f</td>
</tr>
<tr>
<td>1</td>
<td>ca</td>
<td>82</td>
<td>c9</td>
<td>7d</td>
<td>fa</td>
<td>59</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>26</td>
<td>36</td>
<td>3f</td>
<td>f7</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>c7</td>
<td>23</td>
<td>c3</td>
<td>18</td>
<td>96</td>
<td>05</td>
</tr>
<tr>
<td>4</td>
<td>09</td>
<td>83</td>
<td>2c</td>
<td>1a</td>
<td>1b</td>
<td>6e</td>
<td>5a</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>d1</td>
<td>00</td>
<td>ed</td>
<td>20</td>
<td>fc</td>
<td>b1</td>
</tr>
<tr>
<td>6</td>
<td>d0</td>
<td>ef</td>
<td>aa</td>
<td>fb</td>
<td>43</td>
<td>4d</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>a3</td>
<td>40</td>
<td>8f</td>
<td>92</td>
<td>9d</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>cd</td>
<td>0c</td>
<td>13</td>
<td>ec</td>
<td>5f</td>
<td>97</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>81</td>
<td>4f</td>
<td>dc</td>
<td>22</td>
<td>2a</td>
<td>90</td>
</tr>
<tr>
<td>a</td>
<td>e0</td>
<td>32</td>
<td>3a</td>
<td>0a</td>
<td>49</td>
<td>06</td>
<td>24</td>
</tr>
<tr>
<td>b</td>
<td>e7</td>
<td>c8</td>
<td>37</td>
<td>6d</td>
<td>8d</td>
<td>d5</td>
<td>4e</td>
</tr>
<tr>
<td>c</td>
<td>ba</td>
<td>78</td>
<td>25</td>
<td>2e</td>
<td>1c</td>
<td>a6</td>
<td>b4</td>
</tr>
<tr>
<td>d</td>
<td>70</td>
<td>3e</td>
<td>b5</td>
<td>66</td>
<td>48</td>
<td>03</td>
<td>f6</td>
</tr>
<tr>
<td>e</td>
<td>e1</td>
<td>f8</td>
<td>98</td>
<td>11</td>
<td>69</td>
<td>d9</td>
<td>8e</td>
</tr>
<tr>
<td>f</td>
<td>8c</td>
<td>a1</td>
<td>89</td>
<td>0d</td>
<td>bf</td>
<td>e6</td>
<td>42</td>
</tr>
</tbody>
</table>

**Example:**
- **x**: 10
- **y**: Substitute Bytes Table
  - **d4** is mapped to **b8**.
Construction of S-Box and IS-Box

(a) Calculation of byte at row $y$, column $x$ of S-box

(b) Calculation of byte at row $y$, column $x$ of IS-box
S-Box Rationale

- The S-box is designed to be resistant to known cryptanalytic attacks
- The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input
- The nonlinearity is due to the use of the multiplicative inverse
## Encryption Progress – Shift Rows

<table>
<thead>
<tr>
<th></th>
<th>d4</th>
<th>e0</th>
<th>b8</th>
<th>1e</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>b4</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5d</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ae</td>
<td>f1</td>
<td>e5</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

- **Left rotate 0 byte**
- **Left rotate 1 byte**
- **Left rotate 2 bytes**
- **Left rotate 3 bytes**

---

<table>
<thead>
<tr>
<th></th>
<th>d4</th>
<th>e0</th>
<th>b8</th>
<th>1e</th>
</tr>
</thead>
<tbody>
<tr>
<td>b4</td>
<td>41</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5d</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ae</td>
<td>f1</td>
<td>e5</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Encryption Progress – Mix Columns

\[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02 \\
\end{bmatrix}
\cdot
\begin{bmatrix}
04 \\
66 \\
81 \\
e5 \\
\end{bmatrix}
= \begin{bmatrix}
\end{bmatrix}
\]
Mix Columns Rationale

- Coefficients of a matrix based on a linear code with maximal distance between code words ensures a good mixing among the bytes of each column
- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits
Encryption Progress – Add Round Key

\[
\begin{array}{cccc}
a4 & 68 & 6b & 02 \\
9c & 9f & 5b & 6a \\
7f & 35 & ea & 50 \\
f2 & 2b & 43 & 49 \\
\end{array}
\oplus
\begin{array}{cccc}
a4 & 68 & 6b & 02 \\
9c & 9f & 5b & 6a \\
7f & 35 & ea & 50 \\
f2 & 2b & 43 & 49 \\
\end{array}
\]

Round Key

\[
\begin{array}{cccc}
04 & e0 & 48 & 28 \\
66 & cb & f8 & 06 \\
81 & 19 & d3 & 26 \\
e5 & 9a & 7a & 4c \\
\end{array}
\]
AddRoundKey Transformation

- The 128 bits of State are bitwise XORed with the 128 bits of the round key.
- Operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key.
  - Can also be viewed as a byte-level operation.

Rationale:

- Is as simple as possible and affects every bit of State.
- The complexity of the round key expansion plus the complexity of the other stages of AES ensure security.
Key Expansion (1/2)

- Function $g$
  - Rotate word
  - Substitute bytes
  - XOR with round constant

Cipher Key

Round Key 0
Key Expansion (2/2)

- **Function g**
  - Rotate word
    \[ W_3 = (K_{12}, K_{13}, K_{14}, K_{15}) \rightarrow (K_{13}, K_{14}, K_{15}, K_{12}) \]
  - Substitute bytes
    Replace each of these four bytes with the corresponding element in S-box
    Let \((K^*_{13}, K^*_{14}, K^*_{15}, K^*_{12})\) denote the result.
  - XOR with round constant
Key Expansion Rationale

- The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks.
- Inclusion of a round-dependent round constant eliminates the symmetry between the ways in which round keys are generated in different rounds.

The specific criteria that were used are:

- Knowledge of a part of the cipher key or round key does not enable calculation of many other round-key bits.
- An invertible transformation.
- Speed on a wide range of processors.
- Usage of round constants to eliminate symmetries.
- Diffusion of cipher key differences into the round keys.
- Enough nonlinearity to prohibit the full determination of round key differences from cipher key differences only.
- Simplicity of description.
AES Animation

  - Made by Enrique Zabala, Universidad ORT, Montevideo, Uruguay.
AES Encryption and Decryption

Figure 5.3 AES Encryption and Decryption
Mode of Operations

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University
How to use a block cipher?

- Block ciphers encrypt fixed size blocks
  - E.g. DES encrypts 64-bit blocks
- We need some way to encrypt a message of arbitrary length
  - E.g. a message of 1000 bytes
- NIST defines five ways to do it, Called **modes of operations**
  - Electronic codebook mode (ECB)
  - Cipher block chaining mode (CBC) – most popular
  - Output feedback mode (OFB)
  - Cipher feedback mode (CFB)
  - Counter mode (CTR)
Message Padding

- The plaintext message is broken into blocks,
  - E.g., $P_1$, $P_2$, $P_3$, ...

- The last block may be short of a whole block and needs padding.

- Possible padding:
  - Known non-data values (e.g. nulls)
  - Or a number indicating the size of the pad
  - Or a number indicating the size of the plaintext
  - The last two schemes may require an extra block.
Electronic Code Book (ECB)

- Simplest mode
- Plaintext is handled 64 bits at a time
- Each block is encrypted using the same key
- For a given key, there is a unique ciphertext for every 64-bit block of plaintext
- Each block of 64 plaintext bits is encoded independently using the same key
- Application: secure transmission of short pieces of information (e.g. a temporary encryption key)
Encryption and Decryption of ECB

(a) Encryption

Time = 1
\( P_1 \)

\( K \rightarrow \text{Encrypt} \)

\( C_1 \)

Time = 2
\( P_2 \)

\( K \rightarrow \text{Encrypt} \)

\( C_2 \)

(\( \ldots \) \( \ldots \) \( \ldots \))

Time = \( N \)
\( P_N \)

\( K \rightarrow \text{Encrypt} \)

\( C_N \)

(b) Decryption

\( K \rightarrow \text{Decrypt} \)

\( P_1 \)

\( C_1 \)

\( K \rightarrow \text{Decrypt} \)

\( P_2 \)

\( C_2 \)

(\( \ldots \) \( \ldots \) \( \ldots \))

\( K \rightarrow \text{Decrypt} \)

\( P_N \)

\( C_N \)
Strength and Weakness of ECB

Strength

- Simple
- Efficient

Weakness

- the encrypted message blocks are independent
- message repetitions may show in ciphertext
  - if aligned with message block
  - particularly with data such as graphics
  - or with messages that change very little, which become a code-book analysis problem
Cipher Block Chaining (CBC)

- The plaintext is broken into blocks: $P_1, P_2, P_3, ...$
- Each plaintext block is XORed (chained) with the previous ciphertext block before encryption:
  \[ C_i = E_K(C_{i-1} \oplus P_i) \]
  \[ C_0 = IV \]
- Use an Initial Vector (IV) to start the process
- Decryption
  \[ P_i = C_{i-1} \oplus D_K(C_i) \]
- Application: general block-oriented transmission
Encryption and Decryption of CBC

(a) Encryption

(b) Decryption
Strength and Weakness of CBC

- **Strength**
  - The encryption of a block depends on the current and **all** blocks before it.
  - Thus, repeated plaintext blocks are encrypted differently.

- **Weakness**
  - Modifying Ciphertext Blocks
  - Rearranging Ciphertext Blocks
Modifying Ciphertext Blocks

- If an attacker changes a ciphertext block, say $C_1$, the attacker knows the predictable effect on $P_2$
  - Because $P_2 = C_1 \oplus D_K(C_2)$
  - Changing the $i$-th bit of $C_1$ will change the $i$-th bit of $P_2$

Example: Suppose $P_2$ is A’s salary.

If A changes one bit of $C_1$, the corresponding bit of $P_2$ will change.

Note that:
1. Changing $C_1$ will also changes $P_1$
2. There is no predictable effect on $P_1$ by changing $C_1$
Rearranging Ciphertext Blocks

- Suppose A knows the plaintext IV, P_1, P_2, P_3, ... and the corresponding ciphertext C_1, C_2, C_3, ..., A can deduce the modified plaintext.
  - A can also deduce D_K(C_i), because D_K(C_i) = C_{i-1} \oplus P_i
  - Let D_1, D_2, D_3, ... denote D_K(C_1), D_K(C_2), D_K(C_3), ..., respectively
  - If A swaps C_1 and C_2, A can deduce the modified plaintext P_1 and P_2
Designing Stream Ciphers

- ECB and CBC are not stream ciphers
- To encrypt $P_1, P_2, P_3, ..., $ we want to use $E_K$ to generate a different keys for each $P_i$:
  
  $K_1, K_2, K_3, ...$

- Then encrypt $P_i$ as $C_i = P_i \oplus K_i$
- Three different ways to generate $K_1, K_2, K_3, ...$
  - Cipher Feedback (CFB) Mode
  - Output Feedback (OFB) Mode
  - Counter Mode (CTR)
Cipher Feedback Mode (CFB)

- The plaintext is a sequence of segments of $s$ bits ($s \leq$ block size):
  $$P_1, P_2, P_3, \ldots,$$
- Encryption is used to generate a sequence of keys, each of $s$ bits:
  $$K_1, K_2, K_3, \ldots$$
- The ciphertext is $C_1, C_2, C_3, \ldots$, where
  $$C_i = P_i \oplus K_i$$
- How to generate the key stream?
  - The input to the block cipher is a shift register $x$, whose value at stage $i$ is denoted as $x_i$
  - Initially, $x_1 = \text{an initial vector (IV)}$
    - For $i > 1$, $x_i = \text{shift-left-s-bits}(x_{i-1}) | C_{i-1}$
    - Then, $K_i = \text{s-most-significant-bits}(E_K(x_i))$
  - Example value of $s$ is 8, and example value of shift register size is 64 for DES.
Encryption and Decryption of CFB

(a) Encryption

(b) Decryption
Strength and Weakness of CFB

- **Strength**
  - Compared with ECB and CBC:
    - The block cipher is used as a stream cipher.
    - Appropriate when data arrives in bits/bytes.
    - $s$ can be any value; a common value is $s = 8$.
    - A ciphertext segment depends on the current and all preceding plaintext segment.
  - Compared with OFB:
    - Can decrypt at any point. No need to start from the beginning. This makes CFB mode ideal for applications like decrypting an encrypted randomly accessed file.

- **Weakness**
  - A corrupted ciphertext segment during transmission will affect at most $(b/s)+1$ plaintext segments.
Output Feedback Mode (OFB)

- Very similar to CFB in structure
- But $K_{i-1}$ rather than $C_{i-1}$ is feedback to the next stage
- How to generate the key stream?
  - As in CFB, the input to the block cipher is a shift register $x$, whose value at stage $i$ is denoted as $x_i$
  - Initially, $x_1 = $ an initial vector (IV)
    - For $i>1$, $x_i = \text{shift-left-}s\text{-bits}(x_{i-1}) \mid K_{i-1}$
    - Then, $K_i = s$-most-significant-bits($E_K(x_i)$)
Encryption and Decryption of OFB

(a) Encryption

(b) Decryption
Strength and Weakness of OFB

**Strength**

- Keys can be pre-computed. Encryption and decryption is only an XOR in real-time.
- Fast parallel encryption/decryption: blocks can be processed (encrypted or decrypted) in parallel; good for high speed links
- More resistant to transmission errors; a bit error in a ciphertext segment affects only the decryption of that segment.

**Weakness**

- IV should be generated randomly each time and sent with the ciphertext.
  - Insecure if different data are encrypted with the same key and the initial counter: attacker can get the XOR of two plaintext blocks by XORing the two corresponding ciphertext blocks.
Counter Mode (CTR)
Strength and Weakness of CTR

- **Strength:**
  - Has OFB advantage:
    - Keys can be pre-computed. Encryption and decryption is only an XOR in real-time.
    - Fast parallel encryption/decryption: blocks can be processed (encrypted or decrypted) in parallel; good for high speed links
    - More resistant to transmission errors; a bit error in a ciphertext segment affects only the decryption of that segment.
  - Has CFB advantage:
    - Can decrypt at any point. No need to start from the beginning. This makes CTR mode ideal for applications like decrypting an encrypted randomly accessed file.

- **Weakness**
  - Has OFB weakness: IV should be generated randomly each time and sent with the ciphertext.
    - Insecure if different data is encrypted with the same key and the initial counter: attacker can get the XOR of two plaintext blocks by XORing the two corresponding ciphertext blocks.
Feedback Characteristics of Modes of Operation

(a) Cipher block chaining (CBC) mode

(b) Cipher feedback (CFB) mode

(c) Output feedback (OFB) mode

(d) Counter (CTR) mode
# Block Cipher Modes of Operation

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>Typical Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic Codebook (ECB)</td>
<td>Each block of plaintext bits is encoded independently using the same key.</td>
<td>• Secure transmission of single values (e.g., an encryption key)</td>
</tr>
<tr>
<td>Cipher Block Chaining (CBC)</td>
<td>The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.</td>
<td>• General-purpose block-oriented transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Authentication</td>
</tr>
<tr>
<td>Cipher Feedback (CFB)</td>
<td>Input is processed ( s ) bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.</td>
<td>• General-purpose stream-oriented transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Authentication</td>
</tr>
<tr>
<td>Output Feedback (OFB)</td>
<td>Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.</td>
<td>• Stream-oriented transmission over noisy channel (e.g., satellite communication)</td>
</tr>
<tr>
<td>Counter (CTR)</td>
<td>Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.</td>
<td>• General-purpose block-oriented transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Useful for high-speed requirements</td>
</tr>
</tbody>
</table>
Homework

- Textbook Chapter 5, problems:
  - 5.4 and 5.9.

- Textbook Chapter 6, problems:
  - 6.1, 6.3, 6.5 and 6.9.