Computer Security
- Advanced Encryption Standard

Howon Kim
2019.4
Agenda

- Review
- AES Basics
- AES Encryption
  - ByteSub, ShiftRow
  - MixColumn, KeyExpansion, S-Box
- AES Decryption
  - Inverse ByteSub, InverseShiftRow
  - InverseMixColumn
Review

- Group
  - four axioms
  - abelian group

- Field
  - eleven axioms
  - four arithmetic operations

- Modular Arithmetic

- Divisors
  - prime
  - $\mathbb{Z}_n$
  - GCD: Euclid Algorithm
Review

- Finite Field GF($p^n$)

- Finite Fields of Our Interest
  - prime field: GF(p) $\hookrightarrow \mathbb{Z}_p$
  - binary field: GF($2^n$) $\hookrightarrow$ modular polynomial arithmetic

- Finding Multiplicative Inverses
  - Extended Euclidean Algorithm
Review: Block Ciphers

- Symmetric key ciphers
  - Stream ciphers
    - LFSR + Nonlinear Filter or Nonlinear Combiner
    - Bluetooth, GSM, …
  - Block ciphers
    - Convert blocks of plaintext to blocks of ciphertext.
      - Data Encryption Standard (DES): 64-bit blocks
      - Encrypt each block independently with the same key.
Agenda

- Review

- AES Basics
  - AES Encryption
    - ByteSub, ShiftRow
    - MixColumn, KeyExpansion, S-Box
  - AES Decryption
    - Inverse ByteSub, InverseShiftRow
    - InverseMixColumn
The Advanced Encryption Standard

- In 1997, NIST put out a call for candidates to replace DES.

- Requirements for the new algorithm
  - key size: 128, 192, 256 bits
  - block size: 128 bits
  - work on a variety of different hardware
    - 8-bit processors for smart cards
    - 32-bit architecture for PC
  - speed
  - cryptographic strength
The Advanced Encryption Standard

- 21 submissions
- 15 candidates after the first round evaluation
- 5 candidates after the second round evaluation
  - MARS from IBM
  - RC6 from RSA Lab.
  - Rijndael from Joan Daemen & Vincent Rijmen (from Belgium)
  - Serpent
  - Twofish
- Eventually, Rijndael was chosen. (Oct., 2000)
  - pronunciation: “Rain Doll” or “Rhine Dahl”
- Issued as FIPS PUB 197 standard in Nov-2001
AES: Design Criteria

- Resistance against all known attacks
- Speed and code compactness
  - on a wide range of platforms
- Design simplicity
AES Evaluation Criteria

- **Initial criteria**
  - security – effort to practical cryptanalysis
  - cost – computational
  - algorithm & implementation characteristics

- **Final criteria**
  - general security
  - software & hardware implementation ease
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)
AES Shortlist

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

- then subject to further analysis & comment

- saw contrast between algorithms with
  - few complex rounds versus many simple rounds
  - which refined existing ciphers versus new proposals
The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - treats data in 4 groups of 4 bytes
  - operates an entire block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity
Rijndael

- processes data as 4 groups of 4 bytes (state)
- has initial XOR operation
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiplication of groups)
  - add round key (XOR state with key material)
- has no mix columns in final round (that is, in 10\textsuperscript{th}/12\textsuperscript{th}/14\textsuperscript{th} round)
- all operations can be combined into XOR and table lookups - hence very fast & efficient
AES parameters

**Key size (words/bytes/bits)**
- 4/16/128
- 6/24/192
- 8/32/256

**Plaintext block size (words/bytes/bits)**
- 4/16/128
- 4/16/128
- 4/16/128

**Number of rounds**
- 10
- 12
- 14

**Round key size (words/bytes/bits)**
- 4/16/128
- 4/16/128
- 4/16/128

**Expanded key size (words/bytes)**
- 44/176
- 52/208
- 60/240

Ref: Understanding Cryptography by C.Paar
Agenda

- Review
- AES Basics

AES Encryption
- ByteSub, ShiftRow
- MixColumn, KeyExpansion, S-Box

AES Decryption
- Inverse ByteSub, InverseShiftRow
- InverseMixColumn
AES: The Basic Algorithm

- AES is a byte-oriented cipher
- The state $A$ (i.e., the 128-bit data path) can be arranged in a 4x4 matrix:

\[
\begin{array}{cccc}
A_0 & A_4 & A_8 & A_{12} \\
A_1 & A_5 & A_9 & A_{13} \\
A_2 & A_6 & A_{10} & A_{14} \\
A_3 & A_7 & A_{11} & A_{15} \\
\end{array}
\]

with $A_0, \ldots, A_{15}$ denoting the 16-byte input of AES

Ref: Understanding Cryptography by C.Paar
AES: The Basic Algorithm

- Each round consists of four basic steps (layers).
  - **ByteSub Transformation (BS)**
    - Byte-by-byte substitution
    - Non-linear layer is for resistance to DC and LC attacks.
  - **ShiftRow Transformation (SR)**
    - Byte-by-byte permutation
    - For diffusion of the bits over multiple rounds.
  - **MixColumn Transformation (MC)**
    - Substitution using arithmetic over $GF(2^8)$
    - Purpose similar to ShiftRow
  - **AddRoundKey (ARK)**
    - The round key is XORed with the result of the above layers.

- Each layers are invertible. (Different from DES)
AES: The Layers

- 128-bit blocks are grouped into 16 blocks of 8 bits each.
  - These are arranged into a 4*4 matrix, which we call the **State** array.
AES: overall architecture

- **Round function for rounds 1, 2, ..., \( r-1 \):**

  Byte sub layer consists of 16 S-Boxes (which are identical). Only non-linear elements in AES

  permutation of the data on a byte level

  **Diffusion**

  - **Byte Substitution**
  - **ShiftRows**
  - **MixColumn**

  matrix operation which combines ("mixes")

  - **Key Addition**

- **Note:** In the last round, the MixColumn transformation is omitted

Ref: Understanding Cryptography by C. Paar
1. ByteSub Transformation

- a simple substitution of each byte
- uses one table (S-box) of 16x16 bytes containing a permutation of all possible 256 8-bit values
- each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte \{95\} is replaced by row 9 column 5 byte
  - which is the value \{2A\}
- S-box is constructed using a defined transformation of the values in GF(2^8)
- designed to be resistant to all known attacks
• The Byte Substitution layer consists of 16 **S-Boxes** with the following properties:

  The S-Boxes are
  
  • **identical**
  
  • the only **nonlinear** elements of AES, i.e.,
    \[ \text{ByteSub}(A_i) + \text{ByteSub}(A_j) \neq \text{ByteSub}(A_i + A_j), \text{ for } i,j = 0,\ldots,15 \]
  
  • **bijective**, i.e., there exists a one-to-one mapping of input and output bytes
    \[ \Rightarrow \text{S-Box can be uniquely reversed} \]

• In software implementations, the S-Box is usually realized as a lookup table
Diffusion Layer

The Diffusion layer

• provides diffusion over all input state bits

• consists of two sublayers:
  • **ShiftRows Sublayer**: Permutation of the data on a byte level
  • **MixColumn Sublayer**: Matrix operation which combines ("mixes") blocks of four bytes

• performs a linear operation on state matrices $A$, $B$, i.e.,
  $$\text{DIFF}(A) + \text{DIFF}(B) = \text{DIFF}(A + B)$$
Key Addition Layer

- **Inputs:**
  - 16-byte state matrix \( C \)
  - 16-byte subkey \( k_i \)

- **Output:** \( C \oplus k_i \)

- The subkeys are generated in the key schedule

Ref: Understanding Cryptography by C.Paar
1. ByteSub Transformation

- Each of the bytes in the matrix is changed to another byte by the following table (S-box).

<table>
<thead>
<tr>
<th></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>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>
<td>C5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2B</td>
<td>FE</td>
<td>D7</td>
<td>AB</td>
<td>76</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>
<td>F0</td>
<td>AD</td>
<td>D4</td>
<td>A2</td>
<td>AF</td>
<td>9C</td>
<td>A4</td>
<td>72</td>
<td>C0</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>
<td>CC</td>
<td>34</td>
<td>A5</td>
<td>E5</td>
<td>F1</td>
<td>71</td>
<td>D8</td>
<td>31</td>
<td>15</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>
<td>9A</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>E2</td>
<td>EB</td>
<td>27</td>
<td>B2</td>
<td>75</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>
<td>A0</td>
<td>52</td>
<td>3B</td>
<td>D6</td>
<td>B3</td>
<td>29</td>
<td>E3</td>
<td>2F</td>
<td>84</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>
<td>5B</td>
<td>6A</td>
<td>CB</td>
<td>BE</td>
<td>39</td>
<td>4A</td>
<td>4C</td>
<td>58</td>
<td>CF</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>
<td>85</td>
<td>45</td>
<td>F9</td>
<td>02</td>
<td>7F</td>
<td>50</td>
<td>3C</td>
<td>9F</td>
<td>A8</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>
<td>F5</td>
<td>BC</td>
<td>B6</td>
<td>DA</td>
<td>21</td>
<td>10</td>
<td>FF</td>
<td>F3</td>
<td>D2</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>
<td>17</td>
<td>C4</td>
<td>A7</td>
<td>7E</td>
<td>3D</td>
<td>64</td>
<td>5D</td>
<td>19</td>
<td>73</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>
<td>88</td>
<td>46</td>
<td>EE</td>
<td>B8</td>
<td>14</td>
<td>DE</td>
<td>5E</td>
<td>0B</td>
<td>DB</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>
<td>5C</td>
<td>C2</td>
<td>D3</td>
<td>AC</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>E4</td>
<td>79</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>
<td>A9</td>
<td>6C</td>
<td>56</td>
<td>F4</td>
<td>EA</td>
<td>65</td>
<td>7A</td>
<td>AE</td>
<td>08</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>
<td>C6</td>
<td>E8</td>
<td>DD</td>
<td>74</td>
<td>1F</td>
<td>4B</td>
<td>BD</td>
<td>8B</td>
<td>8A</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>
<td>0E</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>B9</td>
<td>86</td>
<td>C1</td>
<td>1D</td>
<td>9E</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>
<td>94</td>
<td>9B</td>
<td>1E</td>
<td>87</td>
<td>E9</td>
<td>CE</td>
<td>55</td>
<td>28</td>
<td>DF</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>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2D</td>
<td>0F</td>
<td>B0</td>
<td>54</td>
<td>BB</td>
<td>16</td>
</tr>
</tbody>
</table>
1. ByteSub Transformation

- **Example**
  - \(11001011 = 0xC\B \rightarrow S\text{-box} \rightarrow 0x1F = 00011111\)
  - \(01011101 = 0x5D \rightarrow S\text{-box} \rightarrow 0x4C = 01001100\)

<table>
<thead>
<tr>
<th>(x)</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>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>
<td>C5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2B</td>
<td>FE</td>
<td>D7</td>
<td>AB</td>
<td>76</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>
<td>F0</td>
<td>AD</td>
<td>D4</td>
<td>A2</td>
<td>AF</td>
<td>9C</td>
<td>A4</td>
<td>72</td>
<td>C0</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>
<td>CC</td>
<td>34</td>
<td>A5</td>
<td>E5</td>
<td>F1</td>
<td>71</td>
<td>D8</td>
<td>31</td>
<td>15</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>
<td>9A</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>E2</td>
<td>EB</td>
<td>27</td>
<td>B2</td>
<td>75</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>
<td>A0</td>
<td>52</td>
<td>3B</td>
<td>D6</td>
<td>B3</td>
<td>29</td>
<td>E3</td>
<td>2F</td>
<td>84</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>
<td>5B</td>
<td>6A</td>
<td>CB</td>
<td>BE</td>
<td>39</td>
<td>4A</td>
<td>4C</td>
<td>58</td>
<td>CF</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>
<td>85</td>
<td>45</td>
<td>F9</td>
<td>02</td>
<td>7F</td>
<td>50</td>
<td>3C</td>
<td>9F</td>
<td>A8</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>
<td>F5</td>
<td>BC</td>
<td>B6</td>
<td>DA</td>
<td>21</td>
<td>10</td>
<td>FF</td>
<td>F3</td>
<td>D2</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>
<td>17</td>
<td>C4</td>
<td>A7</td>
<td>7E</td>
<td>3D</td>
<td>64</td>
<td>5D</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>81</td>
<td>4F</td>
<td>DC</td>
<td>22</td>
<td>DA</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>EE</td>
<td>B8</td>
<td>14</td>
<td>DE</td>
<td>5E</td>
<td>0B</td>
<td>DB</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>
<td>5C</td>
<td>C2</td>
<td>D3</td>
<td>AC</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>E4</td>
<td>79</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>
<td>A9</td>
<td>6C</td>
<td>56</td>
<td>F4</td>
<td>EA</td>
<td>65</td>
<td>7A</td>
<td>AE</td>
<td>08</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>
<td>C6</td>
<td>E8</td>
<td>DD</td>
<td>74</td>
<td>1F</td>
<td>4B</td>
<td>BD</td>
<td>8B</td>
<td>8A</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>
<td>0E</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>B9</td>
<td>86</td>
<td>C1</td>
<td>1D</td>
<td>9E</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>
<td>94</td>
<td>9B</td>
<td>1E</td>
<td>87</td>
<td>E9</td>
<td>CE</td>
<td>55</td>
<td>28</td>
<td>DF</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>
<td>D8</td>
<td>41</td>
<td>99</td>
<td>2D</td>
<td>0F</td>
<td>B0</td>
<td>54</td>
<td>BB</td>
<td>16</td>
</tr>
</tbody>
</table>
1. ByteSub Transformation

- Example

<table>
<thead>
<tr>
<th>EA</th>
<th>04</th>
<th>65</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>45</td>
<td>5D</td>
<td>96</td>
</tr>
<tr>
<td>5C</td>
<td>33</td>
<td>98</td>
<td>B0</td>
</tr>
<tr>
<td>F0</td>
<td>2D</td>
<td>AD</td>
<td>C5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>6E</td>
<td>4C</td>
<td>90</td>
</tr>
<tr>
<td>4A</td>
<td>C3</td>
<td>46</td>
<td>E7</td>
</tr>
<tr>
<td>8C</td>
<td>D8</td>
<td>95</td>
<td>A6</td>
</tr>
</tbody>
</table>
2. ShiftRow Transformation

- a circular byte shift in each
  - 1st row is unchanged
  - 2nd row does 1 byte circular left shift
  - 3rd row does 2 byte circular left shift
  - 4th row does 3 byte circular left shift

- decrypt does shifts to right

- since state is processed by columns, this step permutes bytes between the columns
2. ShiftRow Transformation

- Simple permutation of bytes

![ShiftRow Transformation Diagram]

<table>
<thead>
<tr>
<th>$s_{0,0}$</th>
<th>$s_{0,1}$</th>
<th>$s_{0,2}$</th>
<th>$s_{0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{1,0}$</td>
<td>$s_{1,1}$</td>
<td>$s_{1,2}$</td>
<td>$s_{1,3}$</td>
</tr>
<tr>
<td>$s_{2,0}$</td>
<td>$s_{2,1}$</td>
<td>$s_{2,2}$</td>
<td>$s_{2,3}$</td>
</tr>
<tr>
<td>$s_{3,0}$</td>
<td>$s_{3,1}$</td>
<td>$s_{3,2}$</td>
<td>$s_{3,3}$</td>
</tr>
</tbody>
</table>

1 byte circular left shift

2 byte circular left shift

3 byte circular left shift
### 2. ShiftRow Transformation

- **Example**

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>6E</td>
<td>4C</td>
<td>90</td>
</tr>
<tr>
<td>4A</td>
<td>C3</td>
<td>46</td>
<td>E7</td>
</tr>
<tr>
<td>8C</td>
<td>D8</td>
<td>95</td>
<td>A6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>6E</td>
<td>4C</td>
<td>90</td>
<td>EC</td>
</tr>
<tr>
<td>46</td>
<td>E7</td>
<td>4A</td>
<td>C3</td>
</tr>
<tr>
<td>A6</td>
<td>8C</td>
<td>D8</td>
<td>95</td>
</tr>
</tbody>
</table>
3. MixColumn Transformation

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF($2^8$) using minimal poly $m(x)$
  $= x^8 + x^4 + x^3 + x + 1$

$$
\begin{bmatrix}
0 & 2 & 0 & 3 \\
0 & 1 & 2 & 0 \\
0 & 1 & 0 & 2 \\
0 & 3 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
\bar{s}_{0,0} \\
\bar{s}_{1,0} \\
\bar{s}_{2,0} \\
\bar{s}_{3,0} \\
\end{bmatrix}
=
\begin{bmatrix}
\bar{s}_{0,0} \\
\bar{s}_{1,0} \\
\bar{s}_{2,0} \\
\bar{s}_{3,0} \\
\end{bmatrix}
$$

The MixColumns transformation on a single column $j (0 \leq j \leq 3)$ of State can be expressed as

$$s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})$$
$$s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})$$
3. MixColumn Transformation

- GF($2^8$) matrix multiplication
# 3. MixColumn Transformation

## Example

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>6E</td>
<td>4C</td>
<td>90</td>
<td>EC</td>
</tr>
<tr>
<td>46</td>
<td>E7</td>
<td>4A</td>
<td>C3</td>
</tr>
<tr>
<td>A6</td>
<td>8C</td>
<td>D8</td>
<td>95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>47</th>
<th>40</th>
<th>A3</th>
<th>4C</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>D4</td>
<td>70</td>
<td>9F</td>
</tr>
<tr>
<td>94</td>
<td>E4</td>
<td>3A</td>
<td>42</td>
</tr>
<tr>
<td>ED</td>
<td>A5</td>
<td>A6</td>
<td>BC</td>
</tr>
</tbody>
</table>

\[
\{02\} \times \{87\} \oplus \{03\} \times \{6E\} \oplus \{01\} \times \{46\} \oplus \{01\} \times \{A6\} = \{47\}
\]

\[
\{01\} \times \{87\} \oplus \{02\} \times \{6E\} \oplus \{03\} \times \{46\} \oplus \{01\} \times \{A6\} = \{37\}
\]

\[
\{01\} \times \{87\} \oplus \{01\} \times \{6E\} \oplus \{02\} \times \{46\} \oplus \{03\} \times \{A6\} = \{94\}
\]

\[
\{03\} \times \{87\} \oplus \{01\} \times \{6E\} \oplus \{01\} \times \{46\} \oplus \{02\} \times \{A6\} = \{ED\}
\]

\[
\{02\} \times \{6E\} \oplus \{01\} \times \{46\} \oplus \{02\} \times \{A6\} = \{00001110\}
\]

\[
\{01\} \times \{6E\} \oplus \{02\} \times \{A6\} = \{00011011\}
\]

\[
\{6E\} + \{A6\} = \{01101110\}
\]

\[
\{A6\} = \{01000110\}
\]

\[
\{6E\} = \{10100110\}
\]

\[
\{01\} \times \{x^8 + x^4 + x^3 + x + 1\}
\]

\[
\rightarrow \text{adding the min. polynomial (} m(x) = x^8 + x^4 + x^3 + x + 1 \text{)}
\]
4. RoundKey Addition

- XOR state with 128-bits of the round key
  - bitwise XOR of the 128-bit block and the 128-bit round key

- inverse for decryption is identical since XOR is own inverse, just with correct round key

- designed to be as simple as possible
Key Schedule

- Subkeys are derived recursively from the original 128/192/256-bit input key
- Each round has 1 subkey, plus 1 subkey at the beginning of AES

<table>
<thead>
<tr>
<th>Key length (bits)</th>
<th>Number of subkeys</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>11</td>
</tr>
<tr>
<td>192</td>
<td>13</td>
</tr>
<tr>
<td>256</td>
<td>15</td>
</tr>
</tbody>
</table>

- Key whitening: Subkey is used both at the input and output of AES
  $\Rightarrow$ # subkeys = # rounds + 1

- There are different key schedules for the different key sizes

Ref: Understanding Cryptography by C.Paar
Key Schedule

Example: Key schedule for 128-bit key AES

- Word-oriented: 1 word = 32 bits
- First subkey $W[0]...W[3]$ is the original AES key

Function $g$ rotates its four input bytes and performs a bytewise S-Box substitution $\rightarrow$ nonlinearity

- The round coefficient $RC$ is only added to the leftmost byte and varies from round to round:

  $RC[1] = x^0 = (00000001)_2$
  $RC[2] = x^1 = (00000010)_2$
  $RC[3] = x^2 = (00000100)_2$

  ...
  $RC[10] = x^9 = (00110110)_2$

Ref: Understanding Cryptography by C.Paar
AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - every 4th has S-box + rotate + XOR constant of previous before XOR together
- designed to resist known attacks
AES Key Expansion

KeyExpansion (byte key[16], word w[44]) {
    word temp;
    for (i=0; i<4; i++){
        w[i]=(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
    }
    for (i=4; i<44; i++){
        temp = w[i-1];
        if (i mod 4 = 0)
            temp = SubWord (RotWord (temp) ) ⊕ Rcon[i/4];
        w[i] = w[i-4] ⊕ temp;
    }
}


AES Key Expansion

RotWord([byte0, byte1, byte2, byte3])
    = [byte1, byte2, byte3, byte0]
SubWord([byte0, byte1, byte2, byte3])
    = [Sbox[byte0], Sbox[byte1], Sbox[byte2], Sbox[byte3]]
Rcon[j] = (RC[j], 0, 0, 0)

<table>
<thead>
<tr>
<th>j</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>
</tr>
</thead>
<tbody>
<tr>
<td>RC[j]</td>
<td>01</td>
<td>02</td>
<td>04</td>
<td>08</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>80</td>
<td>1B</td>
<td>36</td>
</tr>
</tbody>
</table>
AES Key Expansion

1. Rotate Bytes of $W_3$.
   $abcd \rightarrow bcda$

2. Apply the $S$-box to each of these four bytes, and obtain $efgh$.

3. Compute $r(i) = 00000010^{(i-4)/4}$ in $GF(2^8)$, that is, if $(i \mod 4 = 0)$ then do exp. computation. Where $rcon(4) = 00000001$.

4. Output $(e \ xor \ r(i), f, g, h)$.

- $r(4) = 00000001$
- $r(8) = 00000010$
- $r(12) = 00000100$
- $r(16) = 00001000$
- $r(20) = 00010000$
- $r(24) = 00100000$
- $r(28) = 01000000$
- $r(32) = 10000000$
- $r(36) = 00011011$
- $r(40) = 00110110$
1. Rotate Bytes of $W_3$.  
   $abcd \rightarrow bcda$

2. Apply the $S$-box to each of these four bytes, and obtain $efgh$.

3. Compute $r(i) = 00000010^{(i-4)/4}$ in GF($2^8$), that is, if $(i \mod 4 = 0)$ then do exp. computation. Where $rcon(4) = 00000001$.

4. Output $w' = (e \text{ xor } r(i), f, g, h)$. 

Construction of the S-box

- Start with a byte \(x_7x_6x_5x_4x_3x_2x_1x_0\), where each \(x_i\) is a bit.

1. Compute its inverse in \(GF(2^8)\) using the irreducible polynomial \(x^8 + x^4 + x^3 + x + 1\).

- If the byte is 00000000, there is no inverse, so we use 00000000 in place of its inverse.

- The resulting byte is \(y_7y_6y_5y_4y_3y_2y_1y_0\).

2. And then, apply the following Affine transformation (shown in next page)
Construction of the S-box

- Apply the following transformation to each bit.  
  
  \[ z_i = y_i \oplus y_{(i+4) \mod 8} \oplus y_{(i+5) \mod 8} \oplus y_{(i+6) \mod 8} \oplus y_{(i+7) \mod 8} \oplus c_i \]
  
  where \( c_7 = 0, c_6 = 1, c_5 = 1, c_4 = 0, c_3 = 0, c_2 = 0, c_1 = 1, c_0 = 1 \)

- It is equivalent to the following matrix transform

  \[
  \begin{pmatrix}
  1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
  1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
  1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
  1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
  1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
  0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
  0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
  \end{pmatrix} \begin{pmatrix}
  y_0 \\
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
  y_5 \\
  y_6 \\
  y_7 \\
  \end{pmatrix} + \begin{pmatrix}
  1 \\
  1 \\
  0 \\
  0 \\
  0 \\
  1 \\
  1 \\
  0 \\
  \end{pmatrix} = \begin{pmatrix}
  z_0 \\
  z_1 \\
  z_2 \\
  z_3 \\
  z_4 \\
  z_5 \\
  z_6 \\
  z_7 \\
  \end{pmatrix}
  \]

- The byte \( z_7z_6z_5z_4z_3z_2z_1z_0 \) is the entry in the S-box.

  **Affine Transformation**

  **Inverse over GF(2^8)** with \( m(x) = x^8 + x^4 + x^3 + x + 1 \).

  **Constant:** 0x63

Nonlinear combination of these two mathematical operations! But inverse exists.
Construction of the S-box: Example

- **Method 1 – Using S-box:** Recall the encryption example.
  - $11001011 = \{\text{CB}\} \Rightarrow \text{S-box} \Rightarrow \{1F\} = 00011111$.

- **Method 2 – Inverse & Affine Transform**
  - $11001011 \Rightarrow \text{GF}(2^8) \Rightarrow x^7 + x^6 + x^3 + x + 1$
  1. Its inverse: $x^2 \Rightarrow 00000100$
  2. $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0 = 00000100$
  2. matrix transform $\Rightarrow z_7z_6z_5z_4z_3z_2z_1z_0 = 00011111$. 
Construction of the S-box: Example

- Inversion operation with $m(x)$: $x^8 + x^4 + x^3 + x + 1$
  - Extended Euclid Alg.

- Affine transformation

- Example:
  - Inverse of "19, $x^4 + x^3 + 1$"
  - By Extended Euclid Alg. We get $x^5 + x^4 + x^3 + x^2 + x + 1$
    - That is, "00111111"
  - Apply this value to Affine transformation matrix.

\[
\begin{bmatrix}
0 \\
0 \\
1 \\
0 \\
1 \\
1
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
0 \\
0 \\
0
\end{bmatrix}
+ \begin{bmatrix}
1 \\
1 \\
0 \\
0 \\
0 \\
1 \\
1 \\
0
\end{bmatrix}
\]
Agenda

- Review
- AES Basics
- AES Encryption
  - ByteSub, ShiftRow
  - MixColumn, KeyExpansion, S-Box
- AES Decryption
  - Inverse ByteSub, InverseShiftRow
  - InverseMixColumn
AES Decryption

- AES decryption is not identical to encryption since steps done in reverse

- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule

- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key
Encryption

Plaintext

Add round key

ByteSub

Shift rows

Mix columns

Add round key

Expand key

w[0, 3]

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Inverse ByteSub

Inverse shift rows

Inverse mix cols

Add round key

Ciphertext

Decryption

Plaintext
1. Inverse ByteSub Transformation

- **S-box**: $11001011 = \{CB\} \rightarrow S-box \rightarrow \{1F\} = 00011111$
- **Inverse**: $00011111 = \{1F\} \rightarrow Inverse S-box \rightarrow \{CB\} = 11001011$

<table>
<thead>
<tr>
<th>$x$</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>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>52</td>
<td>09</td>
<td>6A</td>
<td>D5</td>
<td>30</td>
<td>36</td>
<td>A5</td>
<td>38</td>
<td>BF</td>
<td>40</td>
<td>A3</td>
<td>9E</td>
<td>81</td>
<td>F3</td>
<td>D7</td>
<td>FB</td>
</tr>
<tr>
<td>1</td>
<td>7C</td>
<td>E3</td>
<td>39</td>
<td>82</td>
<td>9B</td>
<td>2F</td>
<td>FF</td>
<td>87</td>
<td>34</td>
<td>8E</td>
<td>43</td>
<td>44</td>
<td>C4</td>
<td>DE</td>
<td>E9</td>
<td>CB</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
<td>7B</td>
<td>94</td>
<td>32</td>
<td>A6</td>
<td>C2</td>
<td>23</td>
<td>3D</td>
<td>EE</td>
<td>4C</td>
<td>95</td>
<td>0B</td>
<td>42</td>
<td>FA</td>
<td>C3</td>
<td>4E</td>
</tr>
<tr>
<td>3</td>
<td>08</td>
<td>2E</td>
<td>A1</td>
<td>66</td>
<td>28</td>
<td>D9</td>
<td>24</td>
<td>B2</td>
<td>76</td>
<td>5B</td>
<td>A2</td>
<td>49</td>
<td>6D</td>
<td>8B</td>
<td>D1</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
<td>F8</td>
<td>F6</td>
<td>84</td>
<td>68</td>
<td>98</td>
<td>16</td>
<td>D4</td>
<td>A4</td>
<td>5C</td>
<td>CC</td>
<td>5D</td>
<td>65</td>
<td>B6</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6C</td>
<td>70</td>
<td>48</td>
<td>50</td>
<td>FD</td>
<td>ED</td>
<td>B9</td>
<td>DA</td>
<td>5E</td>
<td>15</td>
<td>46</td>
<td>57</td>
<td>A7</td>
<td>8D</td>
<td>9D</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>D8</td>
<td>AB</td>
<td>00</td>
<td>8C</td>
<td>BC</td>
<td>D3</td>
<td>0A</td>
<td>F7</td>
<td>E4</td>
<td>58</td>
<td>05</td>
<td>B8</td>
<td>45</td>
<td>B3</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>D0</td>
<td>2C</td>
<td>1E</td>
<td>8F</td>
<td>CA</td>
<td>3F</td>
<td>0F</td>
<td>02</td>
<td>C1</td>
<td>AF</td>
<td>BD</td>
<td>03</td>
<td>01</td>
<td>13</td>
<td>8A</td>
<td>6B</td>
</tr>
<tr>
<td>8</td>
<td>3A</td>
<td>91</td>
<td>11</td>
<td>41</td>
<td>4F</td>
<td>67</td>
<td>DC</td>
<td>EA</td>
<td>97</td>
<td>F2</td>
<td>CF</td>
<td>CE</td>
<td>F0</td>
<td>B4</td>
<td>E6</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>96</td>
<td>AC</td>
<td>74</td>
<td>22</td>
<td>E7</td>
<td>AD</td>
<td>35</td>
<td>85</td>
<td>E2</td>
<td>F9</td>
<td>37</td>
<td>E8</td>
<td>1C</td>
<td>75</td>
<td>DF</td>
<td>6E</td>
</tr>
<tr>
<td>A</td>
<td>47</td>
<td>F1</td>
<td>1A</td>
<td>71</td>
<td>1D</td>
<td>29</td>
<td>C5</td>
<td>89</td>
<td>6F</td>
<td>B7</td>
<td>62</td>
<td>0E</td>
<td>AA</td>
<td>18</td>
<td>BE</td>
<td>1B</td>
</tr>
<tr>
<td>B</td>
<td>FC</td>
<td>56</td>
<td>3E</td>
<td>4B</td>
<td>C6</td>
<td>D2</td>
<td>79</td>
<td>20</td>
<td>9A</td>
<td>DB</td>
<td>C0</td>
<td>FE</td>
<td>78</td>
<td>CD</td>
<td>5A</td>
<td>F4</td>
</tr>
<tr>
<td>C</td>
<td>1F</td>
<td>DD</td>
<td>A8</td>
<td>33</td>
<td>88</td>
<td>07</td>
<td>C7</td>
<td>31</td>
<td>B1</td>
<td>12</td>
<td>10</td>
<td>59</td>
<td>27</td>
<td>80</td>
<td>EC</td>
<td>5F</td>
</tr>
<tr>
<td>D</td>
<td>60</td>
<td>51</td>
<td>7F</td>
<td>A9</td>
<td>19</td>
<td>B5</td>
<td>4A</td>
<td>0D</td>
<td>2D</td>
<td>E5</td>
<td>7A</td>
<td>9F</td>
<td>93</td>
<td>C9</td>
<td>9C</td>
<td>EF</td>
</tr>
<tr>
<td>E</td>
<td>A0</td>
<td>E0</td>
<td>3B</td>
<td>4D</td>
<td>AE</td>
<td>2A</td>
<td>F5</td>
<td>B0</td>
<td>C8</td>
<td>EB</td>
<td>BB</td>
<td>3C</td>
<td>83</td>
<td>53</td>
<td>99</td>
<td>61</td>
</tr>
<tr>
<td>F</td>
<td>17</td>
<td>2B</td>
<td>04</td>
<td>7E</td>
<td>BA</td>
<td>77</td>
<td>D6</td>
<td>26</td>
<td>E1</td>
<td>69</td>
<td>14</td>
<td>63</td>
<td>55</td>
<td>21</td>
<td>0C</td>
<td>7D</td>
</tr>
</tbody>
</table>
2. Inverse ShiftRow Transformation

- Example (right rotate, not left rotate)

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>6E</td>
<td>4C</td>
<td>90</td>
<td>EC</td>
</tr>
<tr>
<td>46</td>
<td>E7</td>
<td>4A</td>
<td>C3</td>
</tr>
<tr>
<td>A6</td>
<td>8C</td>
<td>D8</td>
<td>95</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>87</th>
<th>F2</th>
<th>4D</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>6E</td>
<td>4C</td>
<td>90</td>
</tr>
<tr>
<td>4A</td>
<td>C3</td>
<td>46</td>
<td>E7</td>
</tr>
<tr>
<td>8C</td>
<td>D8</td>
<td>95</td>
<td>A6</td>
</tr>
</tbody>
</table>
3. Inverse MixColumn Transformation

- The MixColumn transformation is a matrix multiplication by a matrix.
- The Inverse MixColumn transformation is a multiplication by its inverse matrix

\[
\begin{pmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{pmatrix}^{-1} = \begin{pmatrix}
0E & 0B & 0D & 09 \\
09 & 0E & 0B & 0D \\
0D & 09 & 0E & 0B \\
0B & 0D & 09 & 0E
\end{pmatrix}
\]
4. Inverse of RoundKey Addition

- AddRoundKey is its own inverse (XOR operation)
Equivalent Inverse Cipher

- The AES decryption structure is not identical to the encryption structure.

<table>
<thead>
<tr>
<th>Encryption</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARK</td>
<td>ARK</td>
</tr>
<tr>
<td>Round 1: BS, SR, MC, ARK</td>
<td>Round 1: ISR, IBS, ARK, IMC</td>
</tr>
<tr>
<td>Round 2: BS, SR, MC, ARK</td>
<td>Round 2: ISR, IBS, ARK, IMC</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Round 9: BS, SR, MC, ARK</td>
<td>Round 9: ISR, IBS, ARK, IMC</td>
</tr>
<tr>
<td>Round 10: BS, SR, ARK</td>
<td>Round 10: ISR, IBS, ARK</td>
</tr>
</tbody>
</table>

- Disadvantage
  - Two separate software or firmware modules are needed for encryption and decryption.
Equivalent Inverse Cipher

- An equivalent version of the decryption algorithm that has the same structure as the encryption algorithm is possible.

- For this, two separate changes are required.
  - An encryption round is BS, SR, MC, ARK.
  - A decryption round is ISR, IBS, ARK, IMC.

- Are these changes possible?
  - First, note that BS and SR are interchangeable.
  - Thus, ISR and IBS can be interchanged.
  - Then a decryption round becomes IBS, ISR, ARK, IMC.
Equivalent Inverse Cipher

How about ARK and IMC?
- These are not interchangeable.
- That is, ‘ARK then IMC’ is different from ‘IMC then ARK’.

Solution
- Write ‘MC then ARK’ as
  \[ E = MC \cdot S \oplus K_i, \]
  where \( S, MC, E, K_i \) are 4 by 4 matrices.
- The inverse is obtained as
  \[ S = MC^{-1} \cdot (E \oplus K_i) = MC^{-1} \cdot E \oplus MC^{-1} \cdot K_i. \]
- Let an addition with \( MC^{-1} \cdot K_i \) be ‘IARK transform’.
- Then ‘ARK then IMC’ is equivalent to ‘IMC then IARK’.
Equivalent Inverse Cipher

Encryption
ARK
Round 1: BS, SR, MC, ARK
Round 2: BS, SR, MC, ARK
... 
Round 9: BS, SR, MC, ARK
Round 10: BS, SR, ARK

Decryption
ARK
Round 1: ISR, IBS, ARK, IMC
Round 2: ISR, IBS, ARK, IMC
... 
Round 9: ISR, IBS, ARK, IMC
Round 10: ISR, IBS, ARK

Decryption
ARK
Round 1: IBS, ISR, IMC, IARK
Round 2: IBS, ISR, IMC, IARK
... 
Round 9: IBS, ISR, IMC, IARK
Round 10: IBS, ISR, ARK
Implementation Aspects

- can efficiently be implemented on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shifting
  - add round key works on byte XORs
  - mix columns requires matrix multiply in $\text{GF}(2^8)$ which works on byte values, can be simplified to use a table lookup
Implementation Aspects

- can efficiently be implemented on 32-bit CPU
  - redefine steps to use 32-bit words
  - can precompute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 16Kb to store tables

- designers believe this very efficient implementation was a key factor in its selection as the AES cipher
Next...

- We will study more on Symmetric Ciphers...
Q&A