MAS 433: Cryptography

Lecture 6

Block Cipher (Part 3, AES)

Wu Hongjun

Lecture Outline

- Classical ciphers
- Symmetric key encryption
  - One-time pad & information theory
  - Block cipher
    - DES, Double DES, Triple DES
    - **AES**
    - Modes of Operation
    - Attacks
  - Stream cipher
- Hash function and Message Authentication Code
- Public key encryption
- Digital signature
- Key establishment and management
- Introduction to other cryptographic topics

Recommended Reading

- CTP Section 3.6
- FIPS 197 (complete AES specifications)
  http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- Wikipedia:
  - AES
    http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Advanced Encryption Standard (AES)

- AES
  - Block cipher
  - 128-bit block size
  - Substitution-permutation network
  - Three different key sizes & round numbers
    - AES-128: 128-bit key + 10 rounds
    - AES-192: 192-bit key + 12 rounds
    - AES-256: 256-bit key + 14 rounds

AES: History

- 1997: NIST called for algorithm to replace DES
- 1998: 15 ciphers submitted for competition
- 2001: Rijndael was approved as AES (FIPS 197)
  - Designers: Joan Daemen, Vincent Rijmen

AES: Applications

- AES
  - Free
  - Simple & decent design
  - High security
  - High performance (hardware & software)
- USA government standard
  - AES-128 for SECRET information
  - AES-192 and AES-256 for TOP SECRET information
- Commercial applications
  - Too many …

Mathematical Preliminaries

1. Euclidean Algorithm and Extended Euclidean Algorithm

Euclidean Algorithm

- used to find the GCD efficiently
- uses the following property repeatedly:

  GCD(a,b) = GCD(b, a mod b)

- Example:

  GCD(12,5) = ?

Mathematical Preliminaries

Extended Euclidean Algorithm

- is used to find x and y satisfying $ax + by = \mathrm{GCD}(a, b)$
- when $\mathrm{GCD}(a, b) = 1$, x is the multiplicative inverse of a modulo b, and y is the multiplicative inverse of b modulo a.
- Basic idea of the algorithm: at each step i in the Euclidean algorithm, find

$$r_i = ax_i + by_i$$

Mathematical Preliminaries

Extended Euclidean Algorithm

- Example $a = 12$, $b = 5$, find GCD(a,b)

GCD(a, b)

$r_1 = a \bmod b = 2$ GCD(b, r_1)

$r_2 = b \bmod r_1 = 1$

$r_1 = a - 2b$

$r_2 = b - 2r_1 = b - 2(a-2b) = 5b-2a$

Euclidean Algorithm: GCD(a,b) = 1

Extended Euclidean Algorithm: 1 = 5b-2a

Mathematical Preliminaries

2. Group, Ring, Field

Group

- A group is **a set**, *G*, together with **an operation** `•' that combines any two elements *a* and *b* to form another element, denoted *a* • *b* or *ab*. To qualify as a group, the set and operation, (*G*, •), must satisfy:
  - Closure: For all *a*, *b* in *G*, the result of the operation, *a* • *b*, is also in *G*.
  - Associativity: (*a* • *b*) • *c* = *a* • (*b* • *c*).
  - Identity element: There exists an element *e* in *G*, such that for every element *a* in *G*, the equation *e* • *a* = *a* • *e* = *a* holds.
  - Inverse element: For each *a* in *G*, there exists an element *b* in *G* such that *a* • *b* = *b* • *a* = *e*
- Example:
  - The set of integers Z together with integer addition

Mathematical Preliminaries

Ring

- A **ring** is **a set** *R* equipped with **two operations**

  $+ : R \times R \rightarrow R$ and $\cdot : R \times R \rightarrow R$,

  called *addition* and *multiplication*. To qualify as a ring, the set and two operations, $(R, +, \cdot)$, must satisfy:
  - $(R, +)$ is required to be an *abelian group* under addition
    - Abelian group has additional property: Commutativity: $a \bullet b = b \bullet a$
  - $(R, \cdot)$ is required to satisfy
    - Closure of multiplication
    - Associativity of multiplication
    - Existence of multiplicative identify
  - The distributive laws:
    - $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
    - $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
- Example:
  - The set of integers Z together with integer addition and multiplication

Mathematical Preliminaries

Field

- A field is a commutative ring whose nonzero elements form a group under multiplication
- Example
  - the field of rational numbers

Mathematical Preliminaries

Finite field (or Galois Field)

– A field that contains a finite number of elements

– For every prime number p and positive integer n, there exists a finite field with p^n elements.

– Example

- GF(7)
  0,1,2,3,4,5,6
  addition and multiplication modulo 7

Évariste Galois (1811-1832)

Mathematical Preliminaries

3. The finite field GF(2^8)

Mathematical Preliminaries

- An element in the finite field GF(2^8) can be denoted in
  - Binary notation: byte ***b***, consisting of eight bits

$$b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$$

  - Polynomial notation: ***b*** is considered as a polynomial with binary coefficients (either 0 or 1):

$$b_7\, x^7 + b_6\, x^6 + b_5\, x^5 + b_4\, x^4 + b_3\, x^3 + b_2\, x^2 + b_1\, x + b_0$$

  - Example: the byte with hexadecimal value {57} (denoted as 0x57, binary 01010111) corresponds with polynomial:

$$x^6 + x^4 + x^2 + x + 1$$

Bit Pattern	Character
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	a
1011	b
1100	c
1101	d
1110	e
1111	f

Mathematical Preliminaries (contd.)

- Addition in finite field GF(2^8):

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (polynomial notation);

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (binary notation);

$\{57\} \oplus \{83\} = \{d4\}$ (hexadecimal notation).

Bit Pattern	Character
0000	0
0001	1
0010	2
0011	3

Bit Pattern	Character
0100	4
0101	5
0110	6
0111	7

Bit Pattern	Character
1000	8
1001	9
1010	a
1011	b

Bit Pattern	Character
1100	c
1101	d
1110	e
1111	f

Mathematical Preliminaries (contd.)

- Multiplication in finite field GF(2^8)
  - denoted by $\bullet$
  - defined as multiplication of binary polynomials modulo an irreducible binary polynomial of degree 8
  - irreducible polynomial: indivisible by any polynomial other than 1 and itself
    - In AES, the following irreducible polynomial is used:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Mathematical Preliminaries (contd.)

- Multiplication in finite field GF(2^8) (contd.)
  - Example:

$$\begin{aligned}
\underline{(x^6+x^4+x^2+x+1)}\ \underline{(x^7+x+1)} &= x^{13}+x^{11}+x^9+x^8+x^7+ \\
&\ x^7+x^5+x^3+x^2+x+ \\
&\ x^6+x^4+x^2+x+1 \\
&= x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1
\end{aligned}$$

$$x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \ \texttt{modulo}\ (x^8+x^4+x^3+x+1)$$

$$= \underline{x^7+x^6+1}.$$

$$\{57\} \bullet \{83\} = \{\text{c1}\}$$

Mathematical Preliminaries (contd.)

- Multiplicative inverse in finite field GF(2^8)
  - Extended Euclidean Algorithm is used to find the inverse
    - For given a and b, find x and y satisfying

$$ax + by = \gcd(a, b)$$

    - If $\gcd(a, b) = 1$, then $ax \bmod b = 1$, i.e.,
      x is the modular multiplicative inverse of a modulo b

Mathematical Preliminaries (contd.)

4. Polynomials with coefficients in GF(2^8)

Mathematical Preliminaries (contd.)

- Polynomials with coefficients in GF(2^8)
  - Let [a_0, a_1, a_2, a_3] denote four bytes

    $$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$
  - The above notation of polynomial is different from the polynomial notation of GF(2^8)
    - The same indeterminate x is used
    - But coefficients here are elements of GF(2^8)
    - And the multiplication of four-term polynomials uses a different reduction polynomial:

      $$x^4 + 1$$
      - $x^4 + 1$ is not an irreducible polynomial over GF(2^8)!
      - Multiplication by a fixed polynomial is not necessarily invertible
      - In AES, a particular fixed polynomial with an inverse is used.

Mathematical Preliminaries (contd.)

- Addition of polynomials with coefficients in GF(2^8)

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$$

Mathematical Preliminaries (contd.)

- Multiplication of polynomials with coefficients in GF(2^8)
  - Denoted by $\otimes$

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

$$c(x) = a(x) \bullet b(x)$$

$$= c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

$$c_0 = a_0 \bullet b_0 \qquad c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \qquad c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \qquad c_6 = a_3 \bullet b_3$$

$$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

Mathematical Preliminaries (contd.)

- Multiplication of polynomials with coefficients in GF(2^8)   (contd.)

$$d(x) = a(x) \otimes b(x)$$

$$= (a(\mathrm{x}) \bullet b(x)) \bmod (x^4 + 1)$$

Since

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4}$$

we have

$$d(x) = c_3\, x_3 + (c_6 + c_2)x^2 + (c_5 + c_1)x + (c_4 + c_0)$$

Mathematical Preliminaries (contd.)

- Multiplication of polynomials with coefficients in GF(2^8) (contd.)

  Let

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$$

  We have

$$d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3)$$

$$d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3)$$

$$d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3)$$

$$d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)$$

Mathematical Preliminaries (contd.)

- Multiplication of polynomials with coefficients in GF(2^8) (contd.)

  For a fixed polynomial $a(x)$, $d(x) = a(x) \otimes b(x)$ can be written in a matrix form:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Mathematical Preliminaries (contd.)

- Multiplication of polynomials with coefficients in GF(2^8) (contd.)

  In AES, an invertible $a(x)$ is used:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

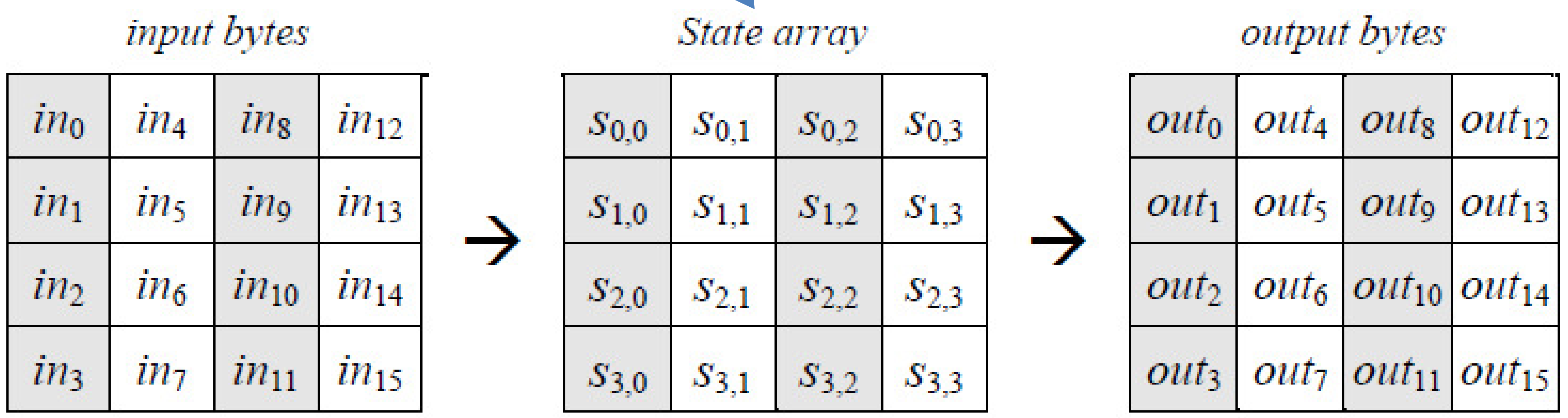
$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

AES: state

16 bytes (the same as the plaintext size)

Represented as a 2D array

– Four rows, four columns



input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

$\rightarrow$

State array

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$\rightarrow$

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

AES: overall

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

```
State = Plaintext

AddRoundKey(State, RoundKey$_0$)

for $i$ = 1 to $r$-1,
    SubBytes(State)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, RoundKey$_i$)
end for;
```

r-1 rounds

```
SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)
```

The last round

```
Ciphertext = state
```

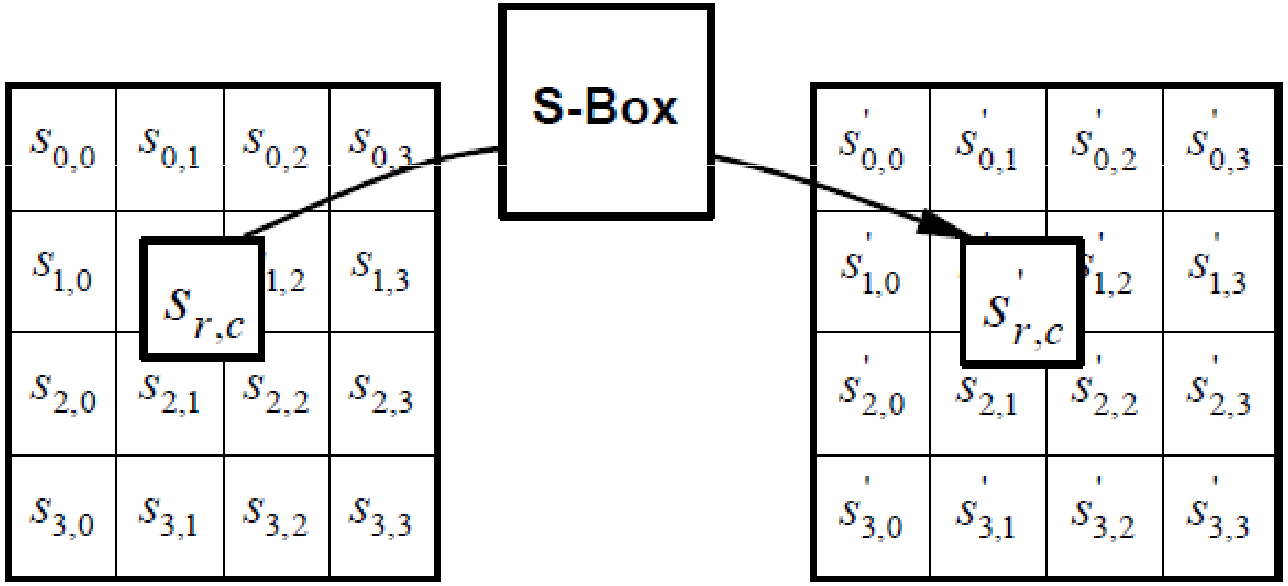
r+1 round keys are used

AES: SubByte

```
State = Plaintext

AddRoundKey(State, Key0)

for $i$ = 1 to $r$-1,
    SubBytes(State)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, RoundKey$_i$)
end for;

SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)

Ciphertext = state
```

AES: SubByte (contd.)

Apply Sbox to each byte in the state



S-Box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{r,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{r,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

AES: SubByte (contd.)

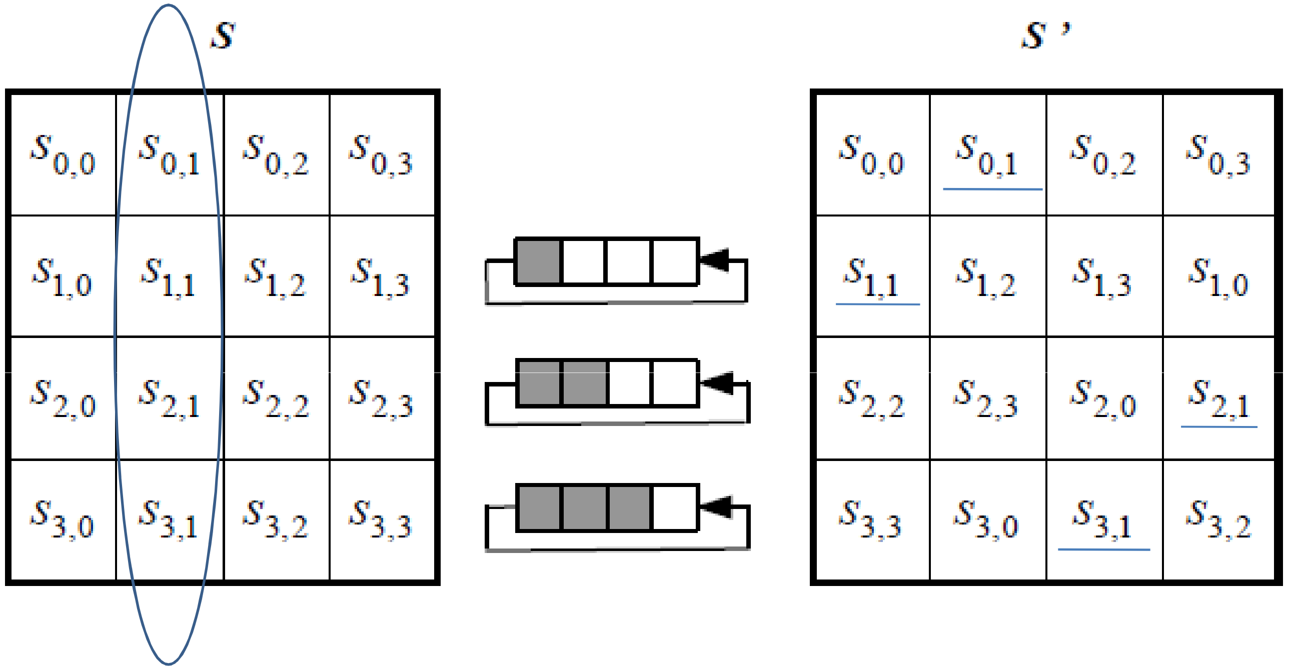
- S-Box
  - 8-bit input; 8-bit output, Invertible
  - Two steps to compute $b' = \mathrm{S}(x)$
    - $b = x^{-1}$ in GF(2^8)
    - Apply the following transformation to b

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

AES: ShiftRows

```
State = Plaintext

AddRoundKey(State, Key0)

for $i$ = 1 to $r$-1,
    SubBytes(State)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, RoundKey$_i$)
end for;

SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)

Ciphertext = state
```

AES: ShiftRows (contd.)



$$S = \begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \qquad S' = \begin{array}{|c|c|c|c|} \hline s_{0,0} & \underline{s_{0,1}} & s_{0,2} & s_{0,3} \\ \hline \underline{s_{1,1}} & s_{1,2} & s_{1,3} & s_{1,0} \\ \hline s_{2,2} & s_{2,3} & s_{2,0} & \underline{s_{2,1}} \\ \hline s_{3,3} & s_{3,0} & \underline{s_{3,1}} & s_{3,2} \\ \hline \end{array}$$

1st row: remain unchanged

2nd row: left rotate one-byte position

3rd row: left rotate two-byte position

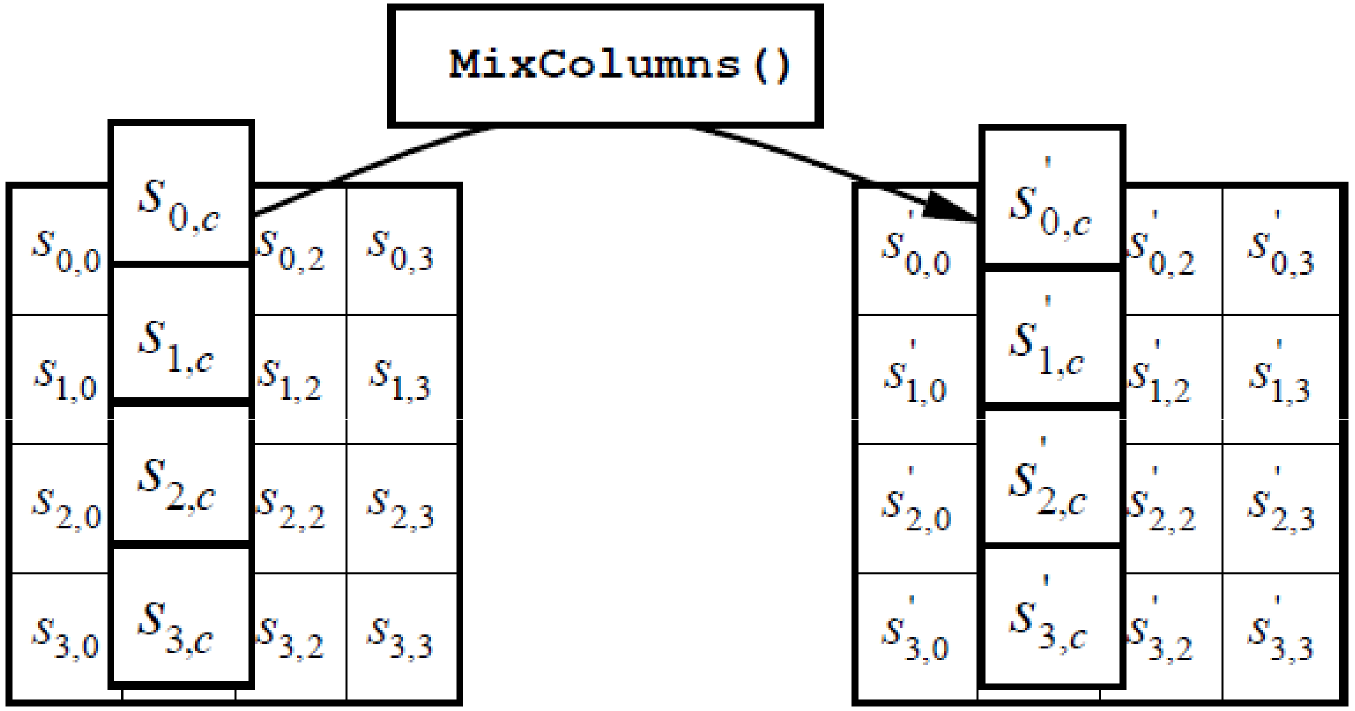
4th row: left rotate three-byte position

Reason for ShiftRows: four elements in one column relocated to 4 different columns after the ShiftRows

AES: MixColumns

```
State = Plaintext

AddRoundKey(State, Key0)

for $i$ = 1 to $r$-1,
    SubBytes(State)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, RoundKey$_i$)
end for;

SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)

Ciphertext = state
```

AES: MixColumns (contd.)



$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$s'(x) = a(x) \otimes s(x)$$

AES: MixColumns (contd.)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

AES: MixColumns (contd.)

- Important properties of MixColumn
  - One input byte affects all four output bytes
  - If there are α non-zero bytes in the input; β non-zero bytes in the output; then $(\alpha + \beta) \geq 5$

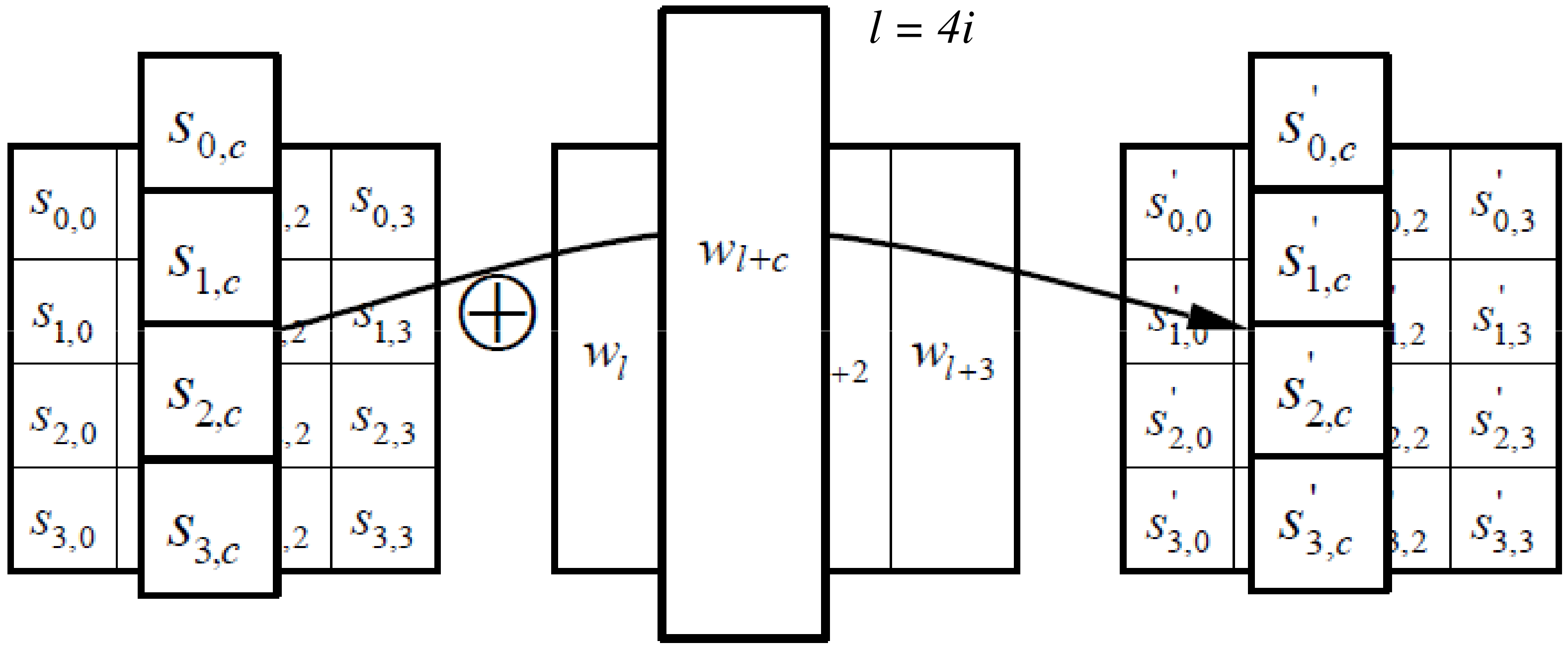
$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

The AES MixColumn generates a maximum distance separable (MDS) code. For a 4-byte *x*, the distance between any (x, Mixcolumn(*x*)) is at least 5 over GF(2^8).

AES: AddRoundKey

```
State = Plaintext

AddRoundKey(State, $Key_0$)                                  $l = 0$

for $i = 1$ to $r$-1,
        SubBytes(State)
        ShiftRows(State)
        MixColumns(State)
        AddRoundKey(State, $RoundKey_i$)                     $l = 4i$
end for;

SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, $RoundKey_r$)

Ciphertext = state
```

AES: AddRoundKey



AES: Key Schedule

- Each round key is 128-bit;
- Round keys are represented as an array of 32-bit words: *w*[*i*]
  - The first round key is w[0], w[1], w[2], w[3]
  - The second round key is w[4], w[5], w[6], w[7]
  - …..
- Secret key is represented as an array of bytes: key[0],key[1],key[2], ….

- Two functions are used in the key schedule
  - SubWord()
    - 4-byte input
    - Apply Sbox to each input byte
  - RotWord()
    - Input: 4-byte [a0, a1, a2, a3]
    - Output: [a1, a2, a3, a0]

- A 32-bit round constant is used for generating each round key
  - Rcon(i): [x^{i-1}, 0, 0, 0] , where $x = 2$, x^{i-1} is the power of x in the field GF(2^8)
  - Different constants for different rounds to prevent slide-attack

AES: Key Schedule

- Example: AES-128

W[0] = (K[0],K[1],K[2],K[3])
W[1] = (K[4],K[5],K[6],K[7])
W[2] = (K[8],K[9],K[10],K[11])
W[3] = (K[12],K[13],K[14],K[15])

```
$RCon[1] \leftarrow$ 01000000
$RCon[2] \leftarrow$ 02000000
$RCon[3] \leftarrow$ 04000000
$RCon[4] \leftarrow$ 08000000
$RCon[5] \leftarrow$ 10000000
$RCon[6] \leftarrow$ 20000000
$RCon[7] \leftarrow$ 40000000
$RCon[8] \leftarrow$ 80000000
$RCon[9] \leftarrow$ 1B000000
$RCon[10] \leftarrow$ 36000000
for $i \leftarrow 0$ to $3$
  do $w[i] \leftarrow (key[4i], key[4i+1], key[4i+2], key[4i+3])$
for $i \leftarrow 4$ to $43$
  do
    $temp \leftarrow w[i-1]$
    if $i \equiv 0 \pmod 4$
      then $temp \leftarrow \text{SubWord}(\text{RotWord}(temp)) \oplus RCon[i/4]$
    $w[i] \leftarrow w[i-4] \oplus temp$
return $(w[0], \ldots, w[43])$
```

Round constants

Load the key into w[ ]

$\boxed{\text{11 round keys}}$

AES: key schedule

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

	Key Length (*Nk words*)	Block Size (*Nb words*)	Number of Rounds (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES: Decryption

(functions are different from that in encryption)

```
State = ciphertext

AddRoundKey(State, RoundKey$_r$)

for $i$ = 1 to $r$-1,                                  ⎫
    InvShiftRows(State)                                ⎪
    InvSubBytes(State)                                 ⎬  $r$-1 rounds
    AddRoundKey(State, RoundKey$_{r-i}$)               ⎪
    InvMixColumns(State)                               ⎪
end for;                                               ⎭

InvShiftRows(State)                                    ⎫
InvSubBytes(State)                                     ⎬  The last round
AddRoundKey(State, RoundKey$_0$)                       ⎪
~~InvMixColumns(State)~~                               ⎭

plaintext = state
```

$\boxed{r\text{+1 round keys are used}}$

AES: Encryption & Decryption

```
State = Plaintext

AddRoundKey(State, RoundKey$_0$)

for $i$ = 1 to $r$-1,
    SubBytes(State)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, RoundKey$_i$)
end for;

SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)

ciphertext = state
```

```
State = ciphertext

AddRoundKey(State, RoundKey$_r$)

for $i$ = 1 to $r$-1,
    InvShiftRows(State)
    InvSubBytes(State)
    AddRoundKey(State, RoundKey$_{r-i}$)
    InvMixColumns(State)
end for;

InvShiftRows(State)
InvSubBytes(State)
AddRoundKey(State, RoundKey$_0$)
~~InvMixColumns(State)~~

plaintext = State
```

AES Implementation

- In practice, the implementation of the cipher must be correct
  - How to check the correctness of the implementation?
    - Using the test vectors provided in the standard

      http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- Efficient in
  - Hardware
  - Embedded system (normally 8-bit processors)
  - Software (32-bit processor)
    - Around 20 clock cycles/byte

AES Implementation

- Starting from 2009, Intel implemented the AES round function and step function of the key schedule in CPUs (part of AVX instruction set)
  - Mainly to address the cache-timing attack on AES
    - For software implementation on CPUs with memory cache, the table lookup timing may vary depending on whether the element is in the cache. Such information can be used to recover the key
  - Results in extremely fast AES
    - around 4 clock cycles/byte
  - AMD follows Intel

Summary

- Mathematical preliminaries
  - Extended Euclidean Algorithm
  - GF(2^8)
  - Polynomials with coefficients in GF(2^8)
- AES
  - Encryption
    - Substitution-Permutation Network
    - Round function
      - different round numbers for different key sizes
    - Key schedule
      - different for different key sizes