MAS 433: Cryptography

Lecture 7

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
  - Simplicity
  - 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) The finite field $GF(2^8)$

Mathematical Preliminaries

- The finite field GF(2^8)
  - 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

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.)

- 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:

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

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

$$
= x^7+x^6+1.
$$

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

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.)

2) 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 $\otimes$

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

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

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

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

$$c_0 = a_0 \bullet b_0$$

$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$$

$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$$

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

$$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$c_6 = a_3 \bullet b_3$$

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

$$d(x) = d_3 x^3 + d_2 x^2 + d_1 x + d_0$$

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}

→

State array

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

→

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)

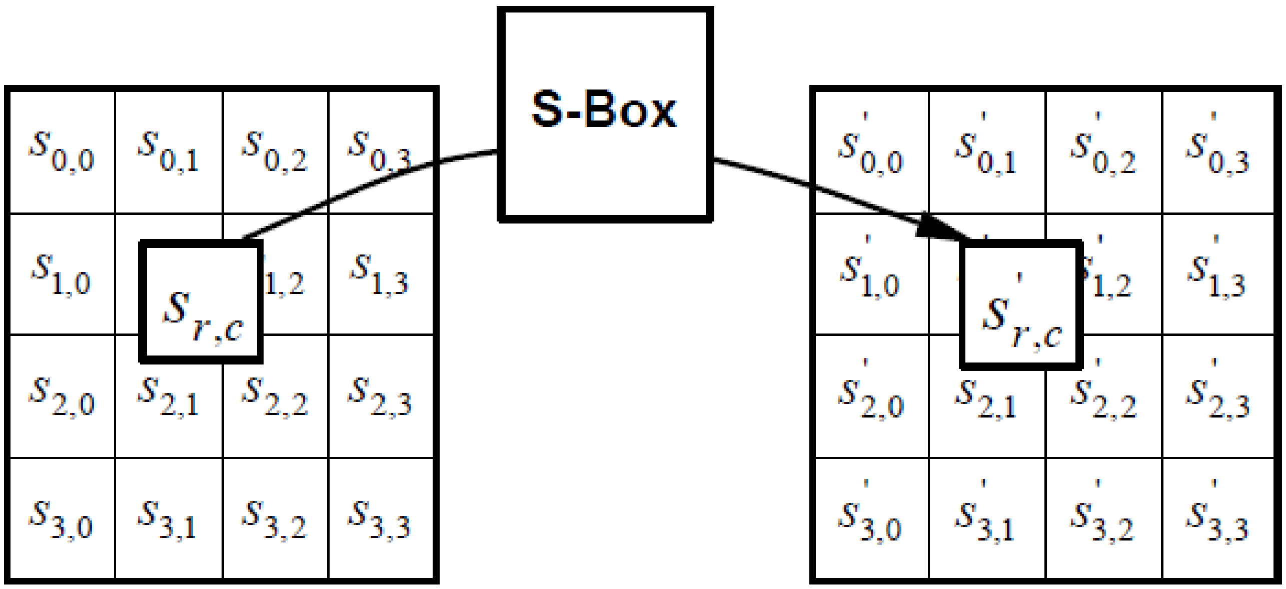
~~MixColumns(State)~~

AddRoundKey(State, RoundKey$_r$)

Ciphertext = state

AES: SubByte (contd.)

Apply Sbox to each byte in the state



AES: SubByte (contd.)

- S-Box
  - 8-bit input; 8-bit output, Invertible
  - Two steps to compute $b' = S(x)$
    - $b = x^{-1}$ 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;

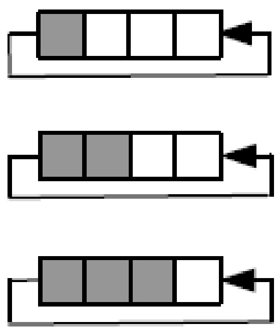
SubBytes(State)
ShiftRows(State)
~~MixColumns(State)~~
AddRoundKey(State, RoundKey$_r$)

Ciphertext = state

AES: ShiftRows (contd.)

s

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



s'

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

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

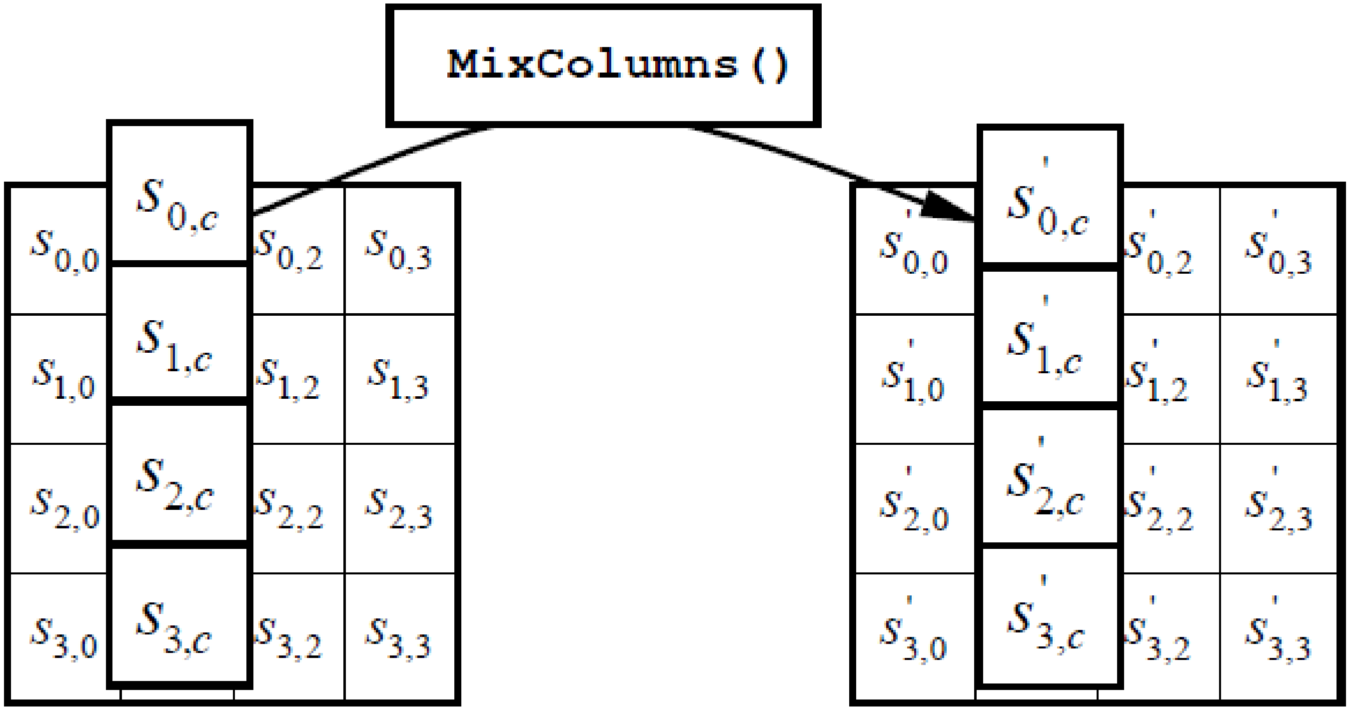
AddRoundKey(State, Key0)

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)

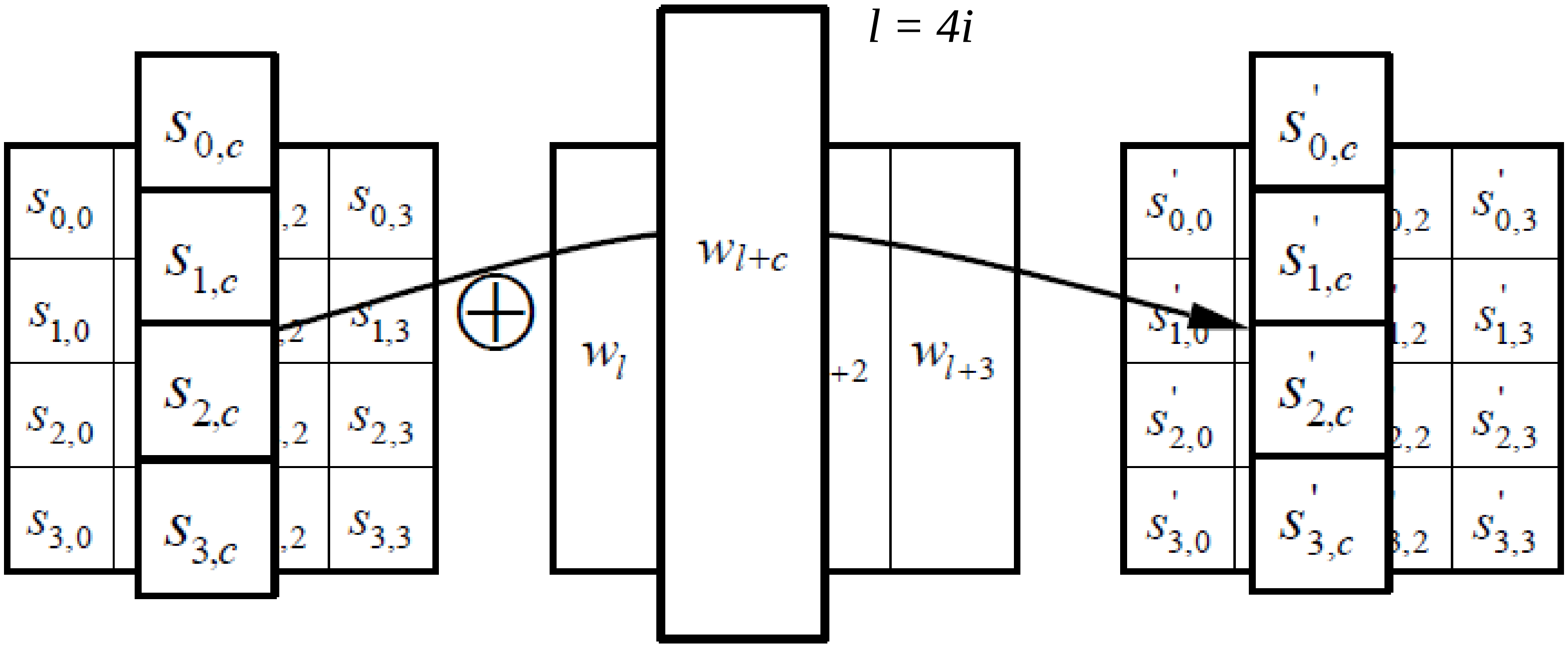
ShiftRows(State)

~~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])

$$\left.\begin{array}{l} RCon[1] \leftarrow \texttt{01000000} \\ RCon[2] \leftarrow \texttt{02000000} \\ RCon[3] \leftarrow \texttt{04000000} \\ RCon[4] \leftarrow \texttt{08000000} \\ RCon[5] \leftarrow \texttt{10000000} \\ RCon[6] \leftarrow \texttt{20000000} \\ RCon[7] \leftarrow \texttt{40000000} \\ RCon[8] \leftarrow \texttt{80000000} \\ RCon[9] \leftarrow \texttt{1B000000} \\ RCon[10] \leftarrow \texttt{36000000} \end{array}\right\} \text{Round constants}$$

$$\left.\begin{array}{l} \textbf{for } i \leftarrow 0 \textbf{ to } 3 \\ \quad \textbf{do } w[i] \leftarrow (key[4i], key[4i+1], key[4i+2], key[4i+3]) \end{array}\right\} \text{Load the key into w[ ]}$$

$$\begin{array}{l} \textbf{for } i \leftarrow 4 \textbf{ to } 43 \\ \quad \textbf{do } \begin{cases} temp \leftarrow w[i-1] \\ \textbf{if } i \equiv 0 \pmod 4 \\ \quad \textbf{then } temp \leftarrow \textsc{SubWord}(\textsc{RotWord}(temp)) \oplus RCon[i/4] \\ w[i] \leftarrow w[i-4] \oplus temp \end{cases} \\ \textbf{return } (w[0], \ldots, w[43]) \end{array}$$

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)

$\left.\begin{array}{l}\text{for } i = 1 \text{ to } r\text{-1}, \\ \quad \text{InvShiftRows(State)} \\ \quad \text{InvSubBytes(State)} \\ \quad \text{AddRoundKey(State, RoundKey}_{r-i}\text{)} \\ \quad \text{InvMixColumns(State)} \\ \text{end for;}\end{array}\right\}$ r-1 rounds

$\left.\begin{array}{l}\text{InvShiftRows(State)} \\ \text{InvSubBytes(State)} \\ \text{AddRoundKey(State, RoundKey}_0\text{)} \\ \cancel{\text{InvMixColumns(State)}}\end{array}\right\}$ The last round

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: Equivalent Decryption Algorithm

- The order of InvSubBytes() and InvShiftRows() can be reversed
- The order of AddRoundKey() and InvMixColumns() can also be reversed, if the columns of decryption round keys are modified using the InvMixColumns() transformation

AES: Equivalent Decryption Algorithm

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

State = ciphertext

AddRoundKey(State, RoundKey$_r$)

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

end for;

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

plaintext = State

RoundKey'$_{r-i}$ = InvMixColumns(RoundKey$_{r-i}$) for i = 1 to r-1

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

Summary

- Mathematical preliminaries
  - $GF(2^8)$
  - 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
  - Two equivalent decryption algorithms
    - One is straight forward inverse
    - Another with modified key schedule