Advanced Encryption Standard

密碼學與應用

海洋大學資訊工程系

丁培毅

Advanced Encryption Standard (AES)

- 1997 NIST call for candidate
  - larger key size (bits): 128, 192, 256
  - larger block size (bits): 128
  - different hardware implementations: 8 bit - 32 bit
- 1998 15 candidates, five finalists
  - MARS (IBM), RC6 (RSA), Rijndael (Daemen and Rijmen), Serpent (Anderson et al), Twofish (Schneier et al)
- 2000 AES standard: Rijndael (FIPS 197)
  replace DES in the following 30 years
  http://csrc.nist.gov/CryptoToolkit/aes/rijndael/

Rijndael

- Pronounced as ‘Reign Dahl’ or ‘Rain Doll’, ‘Rhine Dahl’
- block cipher, 128 bit data block, key lengths can be 128, 192, and 256 bits, 10 rounds, not Feistel structure
- four steps (layers) in each round
  - ByteSub Transformation: resist differential and linear attacks
  - ShiftRow Transformation: diffusion effect
  - MixColumn Transformation: diffusion effect
  - AddRoundKey: key XORed

Rijndael Encryption

- Encryption Algorithm

1. ARK, using the 0-th round key
2. Nine rounds of BS, SR, MC, ARK, using round keys 1 to 9
3. A final round: BS, SR, ARK, using the 10-th round key

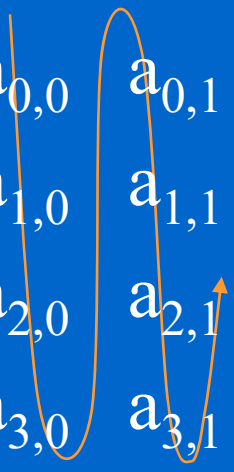
BS: ByteSub
SR: ShiftRow
MC: MixColumn
ARK: AddRoundKey

Input Data

- 128 bits (16 bytes)
- arranged as a 4×4 matrix

$$a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{0,1}, a_{1,1}, \ldots, a_{3,3}$$

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$



- each byte is an elements of $GF(2^8)$, can be added / multiplied mod $X^8+X^4+X^3+X+1$

ByteSub Transformation

- Ex. Input $a_{0,0}$ is 10001011

  1000 $\Rightarrow$ the 9-th row

  1011 $\Rightarrow$ the 12-th column

  Output $b_{0,0}$ is 61

- Each elements in $[a_{i,j}]$ matrix are transformed independently to matrix $[b_{i,j}]$

$$\begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

ByteSub Transformation

- S-box a nonlinear permutation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
1	202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
2	183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
3	4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
4	9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
5	83	209	0	137	32	252	177	91	106	203	190	57	74	76	88	207
6	208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
7	81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
8	205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
9	96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
10	224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
11	231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
12	186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
13	112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
14	225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
15	140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

ShiftRow Transformation

- The four rows of the matrix $[b_{i,j}]$ are shifted cyclically to the left by offsets of 0, 1, 2, and 3 to obtain

$$\begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix} = \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,1} & b_{1,2} & b_{1,3} & b_{1,0} \\ b_{2,2} & b_{2,3} & b_{2,0} & b_{2,1} \\ b_{3,3} & b_{3,0} & b_{3,1} & b_{3,2} \end{pmatrix}$$

MixColumn Transformation

- Perform the following matrix multiplication in $GF(2^8)$

$$\begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} =$$

$$\begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix}$$

RoundKey Addition

- The 128-bit round key matrix $[k_{ij}]$ is derived from the key, and XORed to the output of $[d_{ij}]$

$$\begin{pmatrix} e_{0,0} & e_{0,1} & e_{0,2} & e_{0,3} \\ e_{1,0} & e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,0} & e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,0} & e_{3,1} & e_{3,2} & e_{3,3} \end{pmatrix} = \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & d_{0,3} \\ d_{1,0} & d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,0} & d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,0} & d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \oplus \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}$$

Key Schedule

- 128 bit key K is arranged to 4x4 matrix $[w_{ij}]$ of bytes, let the four column be W(0), W(1), W(2), W(3)
- expanded in the following recursive way
  - $i \not\equiv 0 \bmod 4$, $W(i) = W(i-4) \oplus W(i-1)$
  - $i \equiv 0 \bmod 4$, $W(i) = W(i-4) \oplus T(W(i-1))$
    - where $T(\cdot)$ is defined as

$$T\left(\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}\right) = \begin{pmatrix} S(b) \oplus 00000010^{\frac{i-4}{i}} \\ S(c) \\ S(d) \\ S(a) \end{pmatrix}$$

and $S(\cdot)$ is the S-box

- the i-th round key is (W(4i), W(4i+1), W(4i+2), W(4i+3))

Construction of the S-Box

- There is a simple mathematical formula to calculate each elements in the S-Box
- ex. consider row 12=$(1100)_2$ and column 11=$(1011)_2$, this entry is

$$31 = (00011111)_2$$

  - starting from the byte $(11001011)_2$
  - its inverse in GF(2^8) w.r.t. $X^8+X^4+X^3+X+1$ is $(00000100)_2$
  - multiply by a matrix and add the column vector $(1,1,0,0,0,1,1,0)^T$ in GF(2^8), we obtain the entry $(00011111)_2$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Construction of the S-Box

- The inverse mapping in GF(2^8) was used to achieve non-linearity.
- This simple mapping could possibly allow certain attacks, so it was combined with multiplication by the matrix and adding the vector.
- The matrix was chosen mostly because of its simple form.
- The vector was chosen so that no input ever equals its S-box output or the complement of its S-box output.

Rijndael Decryption

- Each of the steps ByteSub, ShiftRow, MixColumn, and AddRoundKey are invertible
  - The inverse of ByteSub is another lookup table, called InvByteSub
  - The inverse of ShiftRow is obtained by shifting the rows to the right instead of to the left, yielding InvShiftRow
  - The inverse of MixColumn exists because the 4×4 matrix used in MixColumn is invertible. The transformation InvMixColumn is given by multiplication of the matrix

$$\begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix}$$

  - AddRoundKey is its own inverse

Rijndael Decryption(cont'd)

- Rijndael **Encryption**
  - **ARK**
  - **BS, SR, MC, ARK**
  - **…**
  - **BS, SR, MC, ARK**
  - **BS, SR, ARK**
- **Decryption** with all steps reversed (key schedule also reversed)
  - **ARK, ISR, IBS**
  - **ARK, IMC, ISR, IBS**
  - **…**
  - **ARK, IMC, ISR, IBS**
  - **ARK**
- Note: the step sequence of encryption is very different from that of decryption, we want to make it look more alike.

Rijndael Decryption (cont'd)

- Note: BS then SR is the same as SR then BS, since BS acts one byte at a time and SR permutes the bytes. Therefore, the order of ISR and IBS can be reversed.
- The order of ARK and IMC need to be reversed.

  applying ARK then IMC to $[c_{ij}]$:

$$\left[e_{i,j}\right] = \left[m_{i,j}\right]^{-1} \cdot \left(\left[c_{i,j}\right] \oplus \left[k_{i,j}\right]\right) = \left[m_{i,j}\right]^{-1} \cdot \left[c_{i,j}\right] \oplus \left[m_{i,j}\right]^{-1} \cdot \left[k_{i,j}\right]$$

$$= \underbrace{\left(\left[m_{i,j}\right]^{-1} \cdot \left[c_{i,j}\right]\right)}_{\text{IMC}} \underbrace{\oplus \left[k'_{i,j}\right]}_{\text{InvAddRoundKey (IARK)}}$$

IMC then IARK

Rijndael Decryption (cont'd)

- Start from the direct decryption step sequence

  ARK, ISR, IBS
  ARK, IMC, ISR, IBS
  …
  ARK, IMC, ISR, IBS
  ARK

- Modify the above sequence with ISR, IBS reversed and ARK, IMC replaced by IMC, IARK

  ARK,
  IBS, ISR, IMC, IARK,
  IBS, ISR, IMC, IARK,
  …
  IBS, ISR, ARK

Rijndael Decryption (cont'd)

- Decryption Algorithm

1. ARK, using the 10-th round key
2. Nine rounds of IBS, ISR, IMC, IARK, using round keys 9 to 1
3. A final round: IBS, ISR, ARK, using the 0-th round key

Note: 1. Decryption and encryption has essentially the same structure, yet not identical.

2. This explains why MC is omitted in the last round in the encryption algorithm.
3. On 8-bit processors, decryption takes 30% longer than encryption because entries of $[m_{ij}]^{-1}$ are more complex than $[m_{ij}]$ (some modes, like CFB, do not need decryption)

Design Considerations

- Not Feistel system (half the bits are not changed in each round)
  All bits are treated uniformly.
  Diffusing the input bits faster, actually each output bits of a Rijndael round depends on each of the 128 input bits.
- No mystery in the design of S-Box (explicit, simple algebraic way to construct the S-Box)
- The S-box is highly nonlinear, based on $x \rightarrow x^{-1}$ in $GF(2^8)$, excellently at resisting differential and linear attacks and interpolation attacks.

Design Considerations

- The ShiftRow step was added to resist truncated differentials and the Square attack.
- The MixColumn step caused diffusion among the bytes. A change in one input byte results in four output bytes changing.
- The Key Schedule involves nonlinear mixing of the key bits by using S-box. This can resist attacks on the key when partial bits of the key are known. This also ensure that two distinct keys do not have many round keys in common.

Design Considerations

- The number of rounds was chosen to be 10 because there are attacks that are better than brute force up to six rounds. The number of rounds could easily be increased if needed.

Weak Keys

- Symmetry properties and DES-type weak keys
  - Round constants are different in each round to eliminate symmetry in the cipher.
  - The cipher and its inverse use different components to eliminates the possibility for weak and semi-weak keys, as existing for DES.
  - The non-linearity of the key expansion eliminates the possibility of equivalent keys.

Differential Cryptanalysis

- Biham and Shamir, "Differential cryptanalysis of DES-like cryptosystems," Crypto90
- A differential propagation is composed of differential trails(DT), where its propagation ratio(PR) is the sum of the PRs of all DTs that have the specified initial and final difference patterns.
- Necessary condition to resist differential cryptanalysis: No DT with predicated PR > 2^{1-n}, n the block length.
- For Rijndael: No 4-round DT with predicated PR above 2^{-150} (no 8-round trails with PR above 2^{-300}).

Linear Cryptanalysis

- M. Matsui, "Linear cryptanalysis method for DES cipher," Eurocrypt'93
- An input-output correlation is composed of linear trails (LT) that have the specified initial and final selection patterns.
- Necessary condition to be resistant against LC: No LTs with correlation coefficient $> 2^{-n/2}$
- For Rijndael: No 4-round LTs with correlation above 2^{-75} (no 8-round LTs with correlation above 2^{-150}).

Interpolation Attacks

- Jakobsen and Knudsen,1997.
- The attacker constructs polynomials using cipher input/output pairs. If the polynomials have a small degree, only a few pairs are necessary to solve for the coefficients of the polynomial.
- The expression for the S-box is given by

$$63+8f\,X^{127}+b5\,X^{191}+01\,X^{223}+f4\,X^{239}+25\,X^{247}+f9\,X^{251}+09\,X^{253}+05\,X^{25}$$

Advantages

- Implementation aspects
  - Rijndael can be implemented to run at speeds unusually fast on a Pentium (Pro). Trade-off between table size and performance.
  - Rijndael can be implemented on a smart card in a small code, using a small amount of RAM and a small number of cycles.
  - The round transformation is parallel by design.
  - As the cipher makes no use of arithmetic operations, it has no bias towards processor architectures.

Advantages

- Simplicity of design
  - The cipher is fully “self-supporting”.
  - The cipher does not base its security on obscurity and not well understood arithmetic operations.
  - The tight cipher design does not leave enough room to hide a trapdoor.
- Variable block length and extensions
  - Block lengths and key lengths both range from 128 to 256 in steps of 32 bits.
  - Round number can be also modified as a parameter.

Limitations

- The inverse cipher is less suited to be implemented on a smart card than the cipher itself. It takes more code and cycles.
- In software, the cipher and its inverse cipher make use of different code and/or tables.
- In hardware, the inverse cipher can only partially re-use the circuitry that implements the cipher.

Mathematical Backgrounds

- The field $GF(2^8)$
  - Example: $(57)_{16} = (01010111)_2 \Rightarrow x^6+x^4+x^2+x+1$
  - Addition
  - Multiplication
  - Multiplication by x
- Polynomials with coefficients in $GF(2^8)$
  - Multiplication by x

Addition

- The sum of two elements can be obtained by adding corresponding polynomials with modulo 2 addition.

- Example: $(57)_{16} + (83)_{16} = (D4)_{16}$

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

Multiplication

- Multiplication in GF(2^8) can be obtained by multiplying both polynomials modulo an irreducible binary polynomial of degree 8. For Rijndael, this polynomial is called m(x) and given by: $m(x)=x^8+x^4+x^3+x+1$ or $(11B)_{16}$.
- Example: $(57)_{16} \bullet (83)_{16} = (C1)_{16}$

$$(x^6+x^4+x^2+x+1) \bullet (x^7+x+1)$$
$$= x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \pmod{x^8+x^4+x^3+x+1}$$
$$= x^7+x^6+1$$

Extended Euclidean Algorithm

- The multiplication defined above is associative and there is an identity element $(01)_{16}$. For any binary polynomial b(x) of degree below 8, the extended Euclidean algorithm can be used to compute polynomials a(x), c(x) such that

$$b(x)\,a(x) + m(x)\,c(x) = 1.$$

- Hence, $a(x)\cdot b(x) \equiv 1 \pmod{m(x)}$ or
  $b^{-1}(x) \equiv a(x) \pmod{m(x)}$.
- Also, $a(x)\cdot(b(x)+c(x)) \equiv a(x)\cdot b(x)+a(x)\cdot c(x) \pmod{m(x)}$.
- It follows that the set of 256 possible byte values, with the XOR as addition and the multiplication defined as above has the structure of the finite field $GF(2^8)$.

Multiplication by x

- Multiply b(x) by the polynomial x, we have
  $b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x$
- $x \bullet b(x)$ is obtained by reducing the above result (mod m(x)). If $b_7=0$, the reduction is identity operation; if $b_7=1$, m(x) must be subtracted (i.e. XORed).
- That is, multiplication by x or $(02)_{16}$ can be implemented by a left shift and a conditional XOR with $(1B)_{16}$.

Example

- $57 \bullet 13 = FE$

$57 \bullet 02 = x \bullet 57 = AE$

$57 \bullet 04 = x \bullet AE = 47$

$57 \bullet 08 = x \bullet 47 = 8E$

$57 \bullet 10 = x \bullet 8E = 07$

$57 \bullet 13 = 57 \bullet (01 \oplus 02 \oplus 10)$

$= 57 \oplus AE \oplus 07$

$= FE$

Polynomials with coefficients in GF(2^8)

- Consider two polynomials over GF(2^8):

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

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

- The product $c(x) = c_6x^6+c_5x^5+c_4x^4+c_3x^3+c_2x^2+c_1x+c_0$

$$c_0 = a_0 \bullet b_0$$

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

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

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

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

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

$$c_6 = a_3 \bullet b_3$$

Polynomials with coefficients in GF(2^8)

- By reducing c(x) on previous slide modulo a polynomial of degree 4, the result can be reduced to a polynomial of degree below 4. In Rijndael, the polynomial $m(x) = x^4+1$.

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

Polynomials with coefficients in GF(2^8)

- The modular product of a(x) and b(x), denoted by $d(x) = a(x) \otimes b(x)$ is given by

$$d(x) = d_3x^3+d_2x^2+d_1x+d_0 \text{ with}$$

$$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$
$$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$$
$$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$$
$$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

Polynomials with coefficients in GF(2^8)

- The operation consisting of multiplication by a fixed polynomial a(x) can be written as matrix multiplication where the matrix is a circular matrix.

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

Multiplication by x

- If we multiply b(x) by the polynomial x, we have: $b_3x^4+b_2x^3+b_1x^2+b_0x$
- $x \otimes b(x)$ is obtained by reducing the above result modulo $1+x^4$. This gives $b_2x^3+b_1x^2+b_0x+b_3$
- The multiplication by x is equivalent to multiplication by a matrix as above with all a_i =‘00’ except a_1 =‘01’. Let $c(x) = x \otimes b(x)$. We have:

Multiplication by x

- Hence, multiplication by x, or power of x, corresponds to a cyclic shift of the bytes inside the vector

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$