Week 4: Symmetric Encryption II

Dr. Qublai Ali Mirza

University of Gloucestershire

qalimirza@glos.ac.uk

Overview

UNIVERSITY OF GLOUCESTERSHIRE

Advanced Encryption System (*AES*)

- Designed to replace the DES (Data Encryption System)
- NIST issued calls for new cipher proposals in 1997
- The selection criteria were:
  - Security
  - Cost
  - Characteristics of the algorithm
- 15 candidates accepted in 1998, with 5 shortlisted a year later

UNIVERSITY OF GLOUCESTERSHIRE

Shortlisted finalists

- MARS
  - Developed by IBM
- RC6
  - Developed by RSA Labs
- *Rijndael*
  - Developed by Joan Daemen (Proton World International) and Vincent Rijmen (Katholieke Universiteit Leuven)
- Serpent
  - Developed by Ross Anderson (University of Cambridge), Eli Biham (Technion) and Lars Knudsen (UCSD)
- Twofish
  - Developed by Bruce Schneier et.al

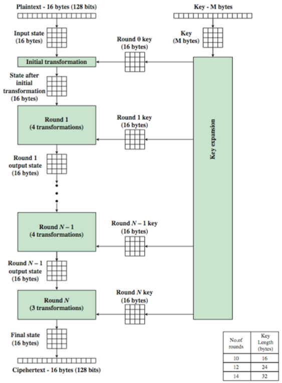
UNIVERSITY OF GLOUCESTERSHIRE

Characteristics

- Iterative in nature
    - 10 rounds for 128 bits
    - 12 rounds for 192 bits
    - 14 rounds for 256 bits
- Data processed as blocks of 4 columns of 4 bytes
- Uses the Vernam cipher to add the round key
- Easy stage of algorithm reversible
- Decryption can be done by simply reversing the encryption process

UNIVERSITY OF GLOUCESTERSHIRE

Inner workings


Plaintext - 16 bytes (128 bits)
Key - M bytes
Input state (16 bytes)
Round 0 key (16 bytes)
Key (M bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round $N-1$ (4 transformations)
Round $N-1$ key (16 bytes)
Round $N-1$ output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext - 16 bytes (128 bits)

No.of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure: AES Inner workings, from [5]

Steps in AES

The AES algorithm consists of *four* main stages, namely

1. Substitute Bytes
2. Shift Rows
3. Mix Columns
4. Add RoundKey

Algorithm overview

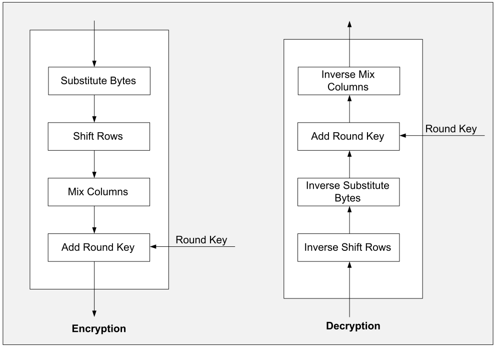
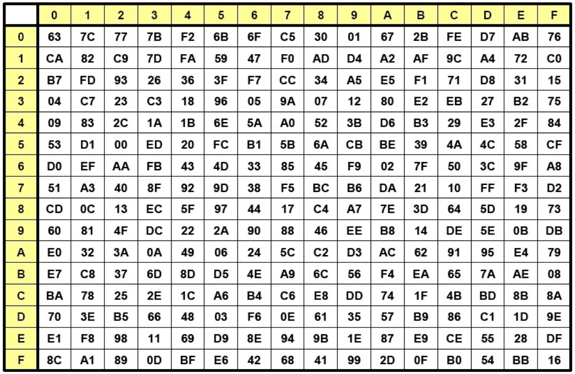


Figure: AES overview, reproduced from [2]

Substitute Bytes

- Involves the substitution of each byte *b* in input state matrix *S*
- Features the use of a 16 $\times$ 16 S-box table consisting of 256 entries
- Look-up achieved by using the first byte as the *row* index and the second byte as the *column* index

AES S-box lookup table



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure: AES S-box, from [1]

UNIVERSITY OF GLOUCESTERSHIRE

Shift Rows

- Involves the *left* circular shift of each row in S
- Shifts each row by a fixed amount in that:
  - Row 0 of S does not shift
  - Row 1 of S is shifted by 1 byte
  - Row 2 of S is shifted by 2 bytes
  - Row 3 of S is shifted by 3 bytes

Mix Columns

- 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 prime poly

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

- m(x) then becomes: `100011011` or `11b` in hex

UNIVERSITY OF GLOUCESTERSHIRE

Example: Finite field multiplication in GF(2^8)

- Calculate the result of $(02 \bullet ad)\ mod(11b)$

Demonstration on the board!

Add Round Key

- XOR state with 128-bits of the round key
- Processed by column (though effectively a series of byte operations)
- inverse for decryption identical

Example

- Suppose that we have plaintext $P =$ `ATTACK AT NIGHT!` and key $K =$ `YELLOW SUBMARINE`
- Step 0: Convert P and K in `hex` values, so that
  - $P =$ 41 54 54 41 43 4b 20 41 54 20 4e 49 47 48 54 21
  - $K =$ 59 45 4c 4c 4f 57 20 53 55 42 4d 41 52 49 4e 45

Matrix representation

- P is converted into a state matrix S like so:

$$S = \begin{bmatrix} 41 & 43 & 54 & 47 \\ 54 & 4b & 20 & 48 \\ 54 & 20 & 4e & 54 \\ 41 & 41 & 49 & 21 \end{bmatrix}$$

- K is also converted into a matrix like so:

$$K = \begin{bmatrix} 59 & 4f & 55 & 52 \\ 45 & 57 & 42 & 49 \\ 4c & 20 & 4d & 4e \\ 4c & 53 & 41 & 45 \end{bmatrix}$$

SubBytes: Key K

- First let $k_0 = [59, 45, 4c, 4c]$, $k_1 = [4f, 57, 20, 53]$, $k_2 = [55, 42, 4d, 41]$, and $k_3 = [52, 49, 4e, 45]$
- Then take the last column k_3 of $K$and apply function $f(k_3)$ which involves:
  - Circular *left* shift of $k_3 = [49, 4e, 45, 52]$
  - Byte Substitution using S-Box, to get (3B, 2F, 6E, 00)
  - Adding a round constant (01, 00, 00, 00) gives us $\gamma = f(k_3) =$ (3C, 2F, 6E, 00)

UNIVERSITY OF GLOUCESTERSHIRE

SubBytes: Key K (Cont.)

- Once we've got γ, we then get the first round key K_1' by performing cumulative `XOR` operation on each column k_0, k_1, k_2, k_3 like so:
  - $k_0' = k_0 \oplus \gamma = [59, 45, 4c, 4c] \oplus [3C, 2F, 6E, 00] = [65, 6A, 22, 4C]$
  - $k_1' = k_0' \oplus k_1$
  - $k_2' = k_1' \oplus k_2$
  - $k_3' = k_2' \oplus k_2$

SubBytes: Key K (Cont.)

- $k'_0 = [65, 6A, 22, 4C]$
- $k'_1 = [2A, 3D, 02, 1F]$
- $k'_2 = [7F, 7F, 4F, 5E]$
- $k'_3 = [2D, 36, 01, 1B]$
- The key K'_1 for the first round then becomes:

  $65, 6A, 22, 4C, 2A, 3D, 02, 1F, 7F, 7F, 4F, 5E, 2D, 36, 01, 1B$

UNIVERSITY OF GLOUCESTERSHIRE

Step 0: Add RoundKey

- At this stage, our state S and key K matrices are respectively

$$S = \begin{bmatrix} 41 & 43 & 54 & 47 \\ 54 & 4b & 20 & 48 \\ 54 & 20 & 4e & 54 \\ 41 & 41 & 49 & 21 \end{bmatrix}, K = \begin{bmatrix} 59 & 4f & 55 & 52 \\ 45 & 57 & 42 & 49 \\ 4c & 20 & 4d & 4e \\ 4c & 53 & 41 & 45 \end{bmatrix}$$

- To get a new state matrix S_0, we perform `XOR` between each entry $s_{ij} \in S$ and $k_{ij} \in K$ giving us

$$S_0 = \begin{bmatrix} 18 & 0c & 01 & 15 \\ 11 & 1c & 62 & 01 \\ 18 & 00 & 03 & 1a \\ 0d & 12 & 08 & 64 \end{bmatrix}$$

Round 1: Substitute Bytes

- Current state matrix S_0 is

$$S_0 = \begin{bmatrix} 18 & 0c & 01 & 15 \\ 11 & 1c & 62 & 01 \\ 18 & 00 & 03 & 1a \\ 0d & 12 & 08 & 64 \end{bmatrix}$$

- First we substitute each entry $s_{ij} \in S_0$ with corresponding entry in the AES S-box
- The new state matrix S_1 then becomes

$$S_1 = \begin{bmatrix} ad & fe & 7c & 59 \\ 82 & 9c & aa & 7c \\ ad & 63 & 7b & a2 \\ d7 & c9 & 30 & 43 \end{bmatrix}$$

UNIVERSITY OF GLOUCESTERSHIRE

Round 1: Shift Rows

- Once we have got our S_1, each row is shifted by 0, 1, 2 and 3
- So after shifting, our S_1 then becomes

$$S_1 = \begin{bmatrix} ad & fe & 7c & 59 \\ 9c & aa & 7c & 82 \\ 7b & a2 & ad & 63 \\ 43 & d7 & c9 & 30 \end{bmatrix}$$

Round 1: Mix Column

- Once we have got our S_1 matrix with the rows shifted, we then perform column mixing
- This is done by multiplying a fixed matrix with S_1 like this:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \oplus \begin{bmatrix} ad & fe & 7c & 59 \\ 9c & aa & 7c & 82 \\ 7b & a2 & ad & 63 \\ 43 & d7 & c9 & 30 \end{bmatrix}$$

- This gives us a new matrix S_1' which is

$$S_1' = \begin{bmatrix} b6 & 77 & 1e8 & 167 \\ a7 & 9b & 24a & 44 \\ 0e & 395 & 301 & 8d \\ 266 & 35c & 037 & 18a \end{bmatrix}$$

UNIVERSITY OF GLOUCESTERSHIRE

Round 1: Add RoundKey

- Given S_1' and K_1 matrices respectively

$$S_1' = \begin{bmatrix} b6 & 77 & 1e8 & 167 \\ a7 & 9b & 24a & 44 \\ 0e & 395 & 301 & 8d \\ 266 & 35c & 037 & 18a \end{bmatrix}, K_1 = \begin{bmatrix} 65 & 2a & 7f & 2d \\ 6a & 3d & 7f & 36 \\ 22 & 02 & 4f & 01 \\ 4c & 1f & 5e & 1b \end{bmatrix}$$

- Finally we perform a pairwise XOR operation, which gives us the final Round 1 output:

$$S_2 = \begin{bmatrix} d3 & 5d & 197 & 14A \\ cd & a6 & 235 & 72 \\ 2c & 397 & 34e & 8c \\ 22a & 343 & 69 & 191 \end{bmatrix}$$

UNIVERSITY OF GLOUCESTERSHIRE

Final output

- For a 128-bit encryption, we perform 9 more rounds of the same operations as in round 1
- The final encrypted ciphertext then becomes

  3F 1E 41 74 C7 75 44 06 96 0F 95 61 9B 2A BE 4A 78 CE EB 21 CD 64 9A 1E BE A0 A8 0E 0C 2D B4 83

Overview

- Developed and proposed by Bruce Schneier in 1993 [4]
- Block cipher
- Uses the *Fiestel* cipher
- Block size: 64-bit
- Key size: Variable, from 32 bits to 448 bits

UNIVERSITY OF GLOUCESTERSHIRE

Blowfish phases

- The Blowfish encryption algorithm consists of three main phases, namely
  - Key Expansion
  - Data Encryption
  - Data Decryption

UNIVERSITY OF GLOUCESTERSHIRE

Key expansion

- P-arrays
  - Consists of 18 subkeys
  - Key length per subkey:32 bits
- S-boxes
  - Number of S-boxes used: 4
  - No.of entries per S-box: 256, i.e.
    - $S_1 = S_{1,0}, S_{1,2}, .., S_{1,255}$
    - $S_2 = S_{2,0}, S_{2,2}, .., S_{2,255}$
    - $S_3 = S_{3,0}, S_{3,2}, .., S_{3,255}$
    - $S_4 = S_{4,0}, S_{4,2}, .., S_{4,255}$
- Both are initialised *sequentially* using the hexadecimal digits of π (minus the 3)

Data Encryption: Key Scheduling

- Initialise the P-arrays and the 4 S-boxes with the hexadecimal digits of π. That means: $P_1 = 0x243f6a88$, $P_2 = 0x85a308d3$, $P_3 = 0x13198a2e$, $P4 = 0x03707344$, and so on.
- `XOR` P_1 with the first 32 bits of K, with the second 32 bits `XOR`ed with P_2 for all possible bits of K for the rest of the P arrays
- Encrypt an all-zero string using the subkeys obtained in (1) and (2)
- Substitute the entries in P_1 and P_2 with the results

Data Encryption: Key Scheduling (cont.)

- Use the modified keys to encrypt the output of the all-zero string encryption
- Substitute the entries in P_3 and P_4 with the resulting output
- Repeat the process until all the P arrays are replaced, and then update the S-boxes with the resulting outputs

N.B: This approach allows for a total of 521 iterations.

Algorithm 1 Blowfish encryption algorithm, from [4]

```
 1: Input: Plaintext $X$ and Key $K$
 2: Divide $X$ into two halves: $X_L$ and $X_R$
 3: for i from 1 to 16 do
 4:     $X_L = X_L \oplus P_i$
 5:     $X_R = F(X_L) \oplus X_R$
 6:     Swap $X_L$ and $X_R$
 7: end for
 8: Swap $X_L$ and $X_R$
 9: $X_R = X_R \oplus P_{17}$
10: $X_L = X_L \oplus P_{18}$
11: Combine $X_L$ and $X_R$
```

Bringing it together

- Today we looked at AES and TwoFish
- We also looked at the inner workings of AES as well as finite field arithmetic
- Next week: *Hashing*

Post-sessional work

- Using the journal article by [3] (available on *Moodle*) as a starting point, write a critical review on the effectiveness as well as computational efficiency between *DES* (inc. its different variants) and *AES*
- Upload your completed work to *Moodle* before next *Monday*.

References I

AES-Advanced Encryption Standard. https://captanu.wordpress.com/tag/aes/. Accessed: 2018-01-17.

Avi Kak. "Lecture 8: AES: The advanced encryption standard". In: *Lecture Notes on'Computer and Network Security', Purdue University, URL: https://engineering. purdue. edu/kak/compsec/NewLectures/Lecture8. pdf* (2016).

Priyadarshini Patil et al. "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish". In: *Procedia Computer Science* 78 (2016), pp. 617–624.

References II

Bruce Schneier. "Description of a new variable-length key, 64-bit block cipher (Blowfish)". In: *International Workshop on Fast Software Encryption*. Springer. 1993, pp. 191–204.

William Stallings. *Cryptography and network security: principles and practices*. Pearson Education India, 2006.

UNIVERSITY OF GLOUCESTERSHIRE

Q & A