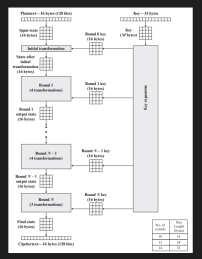
22/02/2021

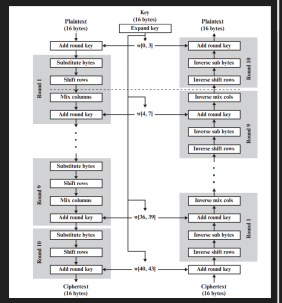
Lecture 14

Advanced Encryption Standard



ADVANCED ENCRYPTION STANDARD (SUMMARY)

1. The cipher takes a plaintext block size of 128 bits, or 16 bytes.
2. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits).
3. The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.
4. The input to the encryption and decryption algorithms is a single 128-bit block.
5. This block is depicted as a 4 * 4 square matrix of bytes.
6. This block is copied into the State array, which is modified at each stage of encryption or decryption.
7. After the final stage, State is copied to an output matrix.
8. Similarly, the key is depicted as a square matrix of bytes.
9. This key is then expanded into an array of key schedule words.
10. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key.
11. The cipher consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key.
12. The first N - 1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described subsequently.
13. The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round, which can be considered Round 0.



1 Word ⇒ 4 Bytes ⇒ 32 bits

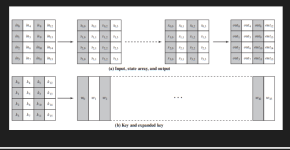
4 Words ⇒ 16 Bytes ⇒ 128 bits

Plaintext Size = 128 bits

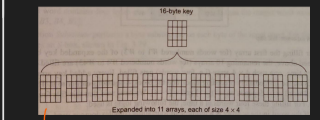
Hence 4 words

OR

16 Bytes



Key Transformation

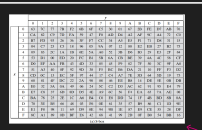


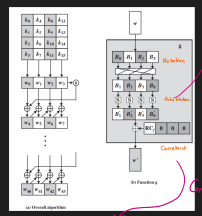
The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).

This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

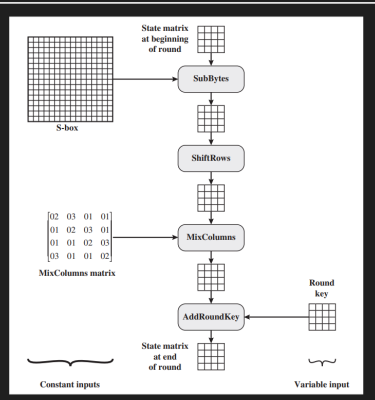
The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added word w[i] depends on the immediately preceding word, w[i - 1], and the word four positions back, w[i - 4].

In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used.





j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36



Overall operations in a Single AES Round

Detailed AES Stages / Transformations

Four different stages are used, one of permutation and three of substitution:

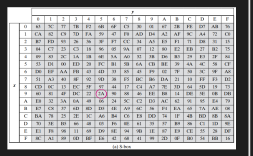
- Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
- ShiftRows: A simple permutation
- MixColumns: A substitution that makes use of arithmetic over GF(2^8)
- AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key

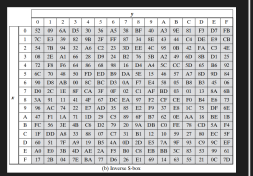
The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.

Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm. For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block, using the result that A ⊕ B ⊕ B = A.

As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order

In mathematics, a finite field or Galois field (so-named in honor of Evariste Galois) is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.



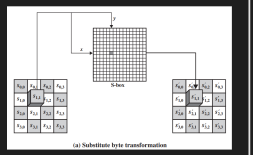


Substitute Bytes Transformation

The forward substitute byte transformation, called SubBytes, is a simple table lookup . AES defines a 16 * 16 matrix of byte values, called an S-box , that contains a permutation of all possible 256 8-bit values.

Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value.

These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.



ShiftRows Transformation

The first row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. The following is an example of ShiftRows.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

The inverse shift row transformation, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.

Mix Column Transformation

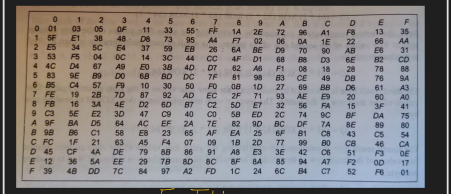
The forward mix column transformation, called MixColumns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column. The transformation can be defined by the following matrix multiplication on State

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

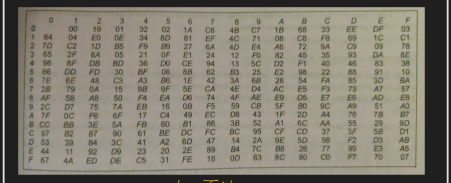
Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications are performed in GF(2^8). The MixColumns transformation on a single column of State can be expressed as

$$\begin{aligned} s'_{1,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

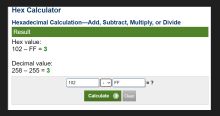
The multiplication mentioned above is performed over a Galois Field. Let us see how the multiplication is performed with the help of lookup tables E-Table and L- Table.



E - Table



L- Table



For example, if the two hex values being multiplied are AF * B , we first look up L(AF) index which returns B7 then lookup L(0B) which returns 4B.

We then simply add these 2 numbers, B7 and 4B to obtain 102.

Hex(102) =258 which is greater than FF i.e 255 so we subtract 255 from it and get the result as 03 in hex.

Now lookup 03 in the E table . i.e E(03) gives 0F. Therefore the result of multiplying AF*B over a Galois Field 2^8 is 0F.

The Mix Column transformation stage happens till the (N-1)th Round. It is not performed in the Nth Round.

Add Round Key

In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key

47	40	A3	4C	⊕	AC	19	28	57	=	EB	59	8B	1B
37	D4	70	9F		77	FA	D1	5C		40	2E	A1	C3
94	E4	3A	42		66	DC	29	00		F2	38	13	42
ED	A5	A6	BC		F3	21	41	6A		1E	84	E7	D6

State Round key Result

The first matrix is State, and the second matrix is the round key. The inverse add round key transformation is identical to the forward add round key transformation, because the XOR operation is its own inverse.