CRYPTOGRAPHY CASE STUDY #2 - AES Operation

José Donato - 2016225043

I. Introduction

A. AES Explanation

The Advanced Encryption Standard (AES) algorithm operates on blocks of 128 bits (16 Bytes) and it can have 128, 192 or 256 bits-length keys. The size of the key will differ on the number of rounds/stages (it can be 10, 12, 14, respectively). It uses substitution-permutation network. Each stage operates on a state matrix 4*4 (block of 128 bits, 16 bytes) and each column of the matrix is considered a word of 4 bytes. In every stage, except the last one, there are four operations in this exact sequence: SubBytes, ShiftRows, MixColumns and AddRoundKey (explained under). In the last round the MixColumns operation doesn't happen. Because every round needs a key, there is one another operation called Key Schedule, it produces the keys (called round keys) from the original provided private key.

B. AES Transformations and Operations

All the following transformations operate on the state matrix:

1) SubBytes: every byte of the state matrix is exchanged by another (the first 4 bits show us the line in the S-BOX and last 4 show the column)
2) ShiftRows: the lines are mixed (the first line isn't changed; on the second line, the bytes shift circularly one position to the left; on the third line, two positions to the left; and on the last one, three positions)
3) MixColumns: the columns are mixed, which is the most complex operation
4) AddRoundKey: simple XOR of the resulting text with the round key

II. Computing Transformations

1) SubBytes:

TABLE I: Example of a state matrix 4x4

01100100	01010001	10100110	10000010
11010001	11111000	10100101	11000011
01100101	01011000	00001001	00111011
10101010	01010100	10000011	00011101

Looking at our state matrix provided in table I we see that the first four bits of the first byte are equal to 6 on decimal, so we see on line 6 and last four bytes are equal to 4 on decimal so we search on column 4. The correspondent value in the S-BOX table provided below ([1]) is 43_{16}. Now doing the same for every cell we will get what we can see on table II.

TABLE II: State Matrix after SubBytes transformation

43	D1	24	2C
3E	8C	06	2E
4D	6A	01	E2
AC	20	EC	A4

2) ShiftRows (result in table III):

TABLE III: State Matrix after ShiftRows transformation

43	D1	24	2C
8C	06	2E	3E
01	E2	4D	6A
A4	AC	20	EC

3) MixColumns:
   a) The operations must be done according to the following matrix:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

   Which translates to:

s'00 = 2•s00 ⊕3•s10 ⊕1•s20 ⊕1•s30
s'01 = 2•s01 ⊕3•s11 ⊕1•s21 ⊕1•s31
...
s'10 = 1•s00 ⊕2•x10 ⊕3•s20 ⊕1•s30
...
s'20 = 1•s00 ⊕1•s10 ⊕2•s20 ⊕3•s30
...
s'30 = 3•s00 ⊕1•s10 ⊕1•s20 ⊕2•s30
...

   b) multiply by 2:
      i) first shift 1 bit to the left. Using the first cell of our state matrix (43_{16}) we got:

43_{16} equals to 01000011_2. shift 1 bit: 10000110_2

ii) most significant bit of the original byte is 0 (**0**1000011$_2$) so nothing needs to be done. if it was 1, the XOR operation of the result of the last operation with the value 00011011$_2$ must be done

$2 \bullet s_{00} = 10000110_2$

c) multiply by 3: results of the multiplication by 2 as explained above and then XOR with the original value:

$3 \bullet s_{10} = (2 \bullet s_{10}) \oplus s_{10}$

Example for first row and first column (s_{00}):

$s'_{00} = 2 \bullet s_{00} \oplus 3 \bullet s_{10} \oplus 1 \bullet s_{20} \oplus 1 \bullet s_{30}$

So we need to calculate $2 \bullet s_{00}$ (which we already have from above, 43_{16}) and $3 \bullet s_{10}$ then XOR all the bytes. $s_{10}=(8C)_{16}= (10001100)_2$

• shift 1 to the left: $(00011000)_2$
• first bit was 1 so:
$(00011000)_2 \oplus (00011011)_2$
$= (00000011)_2$
• XOR with s_{10}: $(10001111)_2 = (8F)_{16}$

So:

$s'_{10} = 3 \bullet s_{10} = (2 \bullet s_{10}) \oplus s_{10} = (8F)_{16}$

d) Now that we have the 2 values we needed to calculate, we just need to XOR the four numbers:

$$s'_{00} = 2 \bullet s_{00} \oplus 3 \bullet s_{10} \oplus 1 \bullet s_{20} \oplus 1 \bullet s_{30}$$
$$= 10000110_2 \oplus 10001111_2 \oplus 00000001_2 \oplus 10100100_2$$
$$= 00001001_2 \oplus 10100101_2$$
$$= 10101100_2 = (AC)_{16}$$

This was an example done only for the first cell of the matrix. To complete this step the 16 cells of the state matrix must be calculated.

4) AddRoundKey: each column XOR with each word of the round key (four words and columns in total).
For example, if the first column of the matrix after the three last transformations is the table IV and the first word of the round key is the table V:

43
8C
01
A4

TABLE IV: First column of the state matrix

AC
32
10
23

TABLE V: First word of the round key

$$43_{16} \oplus AC_{16} = 2F_{16}$$
$$8C_{16} \oplus 32_{16} = BE_{16}$$
$$01_{16} \oplus 10_{16} = 11_{16}$$
$$A4_{16} \oplus 23_{16} = 81_{16}$$

So, the final column after AddRoundKey transformation would be the table VI.

Note that, to complete this stage, all columns needed to be XOR with the rest of the words in the round key. For sake of simplicity only the first was done.

TABLE VI: State Matrix' first column after AddRoundKey

2F
BE
11
81

5) Key Schedule: There is still one more operation in AES, which is called Key Schedule. The purpose of this operation is to extend the initial key so that all rounds can have one key with the same size. In this step, the initial key is expanded in (N+1, where N equals to round number) bytes. For AES-128 the 128-bit key is expanded to 11 round keys, each one with 4 bytes (total 176 bytes). It has N+1 iterations (in AES-128, 11 iterations).
Steps to complete this operation:

• the first 4 words of four bytes are copied directly from the key
• then, for the words that are not first in each iteration: $w_n = w_{n-1} \oplus w_{n-4}$
• for the first word of each iteration: $w_n = g(w_{n-1}) \oplus w_{n-4}$

The output of function g can be calculated using the following steps:

• circular shift one position to the left
• S-BOX [1] operation as in SubBytes transformation to all the bytes
• XOR operation of the result of the last steps with a 4-byte word (first byte is a constant and the last three are zero, the constant depends of the iteration [2])

I'll only show the function g operation on one word. So, if we have, for example, the first word (table VII) at the first iteration (RC constant will be 01 and all other bytes of the word 0):

AC
32
10
23

TABLE VII: First word of key

32
10
23
AC

TABLE VIII: Key after shift operation

7D
CA
26
DE

TABLE IX: Key after S-BOX substitution

7C
CA
26
DE

TABLE X: Final key after XOR with word 01 0 0 0

III. Experimental Results

To make some tests i made a script in python (`https://repl.it/repls/QuixoticIvoryCopyleft`) that with always the same key will make the encryption for multiple plaintexts and see how many bits of the output were different. To calculate this i did the XOR of one first cyphertext with other and count the number of 1's in the result, that way we can know how many bits are different between those too. For example, the test1 cyphertext has 69 bits different to test0 cyphertext. Note that only 1 bit changed between test0 and test1 and it produced a difference of 69 bits between them.

Plaintext	**Different bits in the output**
test0	0
test1	69
test2	70
test3	64
test4	61
test5	65
cn4b2KTYLd	0
ldx1uw9pjq	65
xFmkEGojlmmAE549ptuo	138

As expected and explained before, if we change only one bit a lot of bits (60 to 70 bits) are changed in the cyphertext, this is called the avalanche effect. We can also see in the last three rows that as we increase the number of altered bits of plaintext (the changes increase each row) and the number of altered bits of cyphertext also increases more and more.

IV. Conclusions

We can now take some conclusions. The first one is that this algorithm achieves what is expected in a strong encryption algorithm. It produces complete random output from some input and it guarantees the avalanche effect (small changes in input lead to big changes in output). Since AES is based in a substitution-permutation network, it's important to see that the bytes are mixed as much as possible. An interesting property about this is that only after three rounds, because of the ShiftRows and MixColumns transformations, every byte of the state matrix depends on all 16 plaintext bytes [6].

With the increase of cyber-attacks, the need of a good encryption algorithm increases as well. Special AES instructions was introduced in Intel CPUs and it is now used widely, which shows how strong and reliable this algorithm AES is.

References

[1] https://captanu.files.wordpress.com/2015/04/aes_sbox.jpg

[2] https://en.wikipedia.org/wiki/AES_key_schedule

[3] https://www.devglan.com/online-tools/aes-encryption-decryption

[4] https://www.rapidtables.com/convert/number/binary-to-hex.html

[5] Fernando Boavida and Mário Bernardes, FCA, Introdução a Criptografia, 2019.

[6] Christof Paar and Jan Pelzl, Springer, Understanding Cryptography, 2010.