January 26, 2020

Abstract

1 Advanced Encryption Standard

1.1 Evaluation Criteria for AES

Essentially the cipher's submitted NIST for AES were judged on three broad categories

1. Security
   (a) Actual security compared with other submitted algorithms
   (b) Randomness
   (c) Soundness (mathematical)
   (d) Other security factors
2. Cost
   (a) Licensing
   (b) Computational Efficency
   (c) Memory Requirements
3. Algorithm and implementation characteristics
   (a) Flexibility in key and block size, wide variety of plateforms and applications, and USE: implemented as a stream cipher, message authentication code, random number generator.
   (b) Hareware and software suitable
   (c) Simplicity

1.2 The AES Cipher

Figure 1 shows a diagram of the AES cipher. Look through the review questions below for a good explanation of each step.

1.3 Review Questions

What was the original set of criteria used by NIST to evaluate candidate AES ciphers?

In general they said:

1. Security strength equal to or greater than 3DES
2. Significantly improved efficiency
3. Symmetric block cipher with a block length of 128 bits.
4. Support key lengths of 128, 192, and 256 bits.

The specific evaluation criteria:

1. Security: This referes to the effort required to cryptanalyze an algorithm.
2. Cost: Practical, Efficient enough to use on high bandwidth links and high speed applications.
3. Algorithm and Implementation Characteristics: flexibility, suitability for a variety of hardware and software implementations, simplicity

What was the final set?

1. General Security:
   (a) Software Implementations: Speed
   (b) Hardware Implementations: small hardware size to keep cost down.
   (c) Attacks on Implementations: timing attacks and power attacks.
   (d) Encryption versus decryption: Are they the same...
   (e) Key agility: ability to change keys quickly and efficiently
2. Other versatility and fexibility: Parameter flexibility (other key and block sizes, change in the number of rounds),
3. Implementation Flexibility (optimizing cipher elements for particular environments).
4. Potential for instruction-level parallelism:The ability to exploit ILP in processors.

What is the power analysis? Observing the power used to detect a multiply or add operation or to see if ones or zeros are being written.

What is the difference between Rijndael and AES? Rijndael took different blocks sizes of 128, 192, 256. AES only takes 128.

What is the purpose of the *state* array? The state array holds the input block that is massaged through each round.

How is the S-Box constructed?

1. Initialize the $S - Box$ with the byte values in ascending sequence row by row
   $$\begin{bmatrix} 00 & 01 & ... & 0F \\ 10 & 11 & ... & 1F \\ \vdots & & \ddots & \\ F0 & F1 & & FF \end{bmatrix}$$
   Thus any element value in row A element B is 0xAB

2. Map each byte in the S-Box to its multiplicative inverse in $GF(2^8)$ where $00 \rightarrow 00$.

3. Each byte in the S-Box consists of 8 bits labeled $(b_7, b_6, ..., b_0)$. Apply the following transformation to each bit of each byte:

   $$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

   where c_i is the i^{th} bit of byte c with the value $\{63\}$. That is $(c_7c_6c_5c_4c_3c_2c_1c_0) = (01100011)$.

Briefly describe Sub Bytes.

SubBytes: Uses the S-box described above to perform a byte-by-byte substitution of the state (or input) block as show in Figure 2

In the decryption algorithm an Inverse-S-Box is used. $S : EA \rightarrow 87$ and $S^{-1} : 87 \rightarrow EA$.

Briefly describe ShiftRow Transformation.

To perform the ShiftRow transformation, we take the state and "left circular shift" row 0 by 0 byts, 1 by 1 byte, row 2 by 2 bytes, and row 3 by 3 bytes. To perform the inverse we use right shifts instead of left shifts.

How many bytes in "State" are affected by Shift Rows? 12 Bytes

Briefly describe MixColumns.

MixColumns operates on each column individually and is defined by the following matrix multiplication on state:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{matrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{matrix}$$

In the matrix multiplication we must remember that we are doing multiplication in $G\left(2^8\right)$. We do multiplication as follows:

$$01 * S_{i,j} = S_{i,j}$$

$$02 * S_{i,j} = \begin{cases} (b_6b_5b_4b_3b_2b_1b_00) & if \;\; b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_00) \oplus (00011011) & if \;\; b_7 = 1 \end{cases}$$

$$03 * S_{i,j} = \begin{cases} (b_6b_5b_4b_3b_2b_1b_00) \oplus (b_7b_6b_5b_4b_3b_2b_1b_0) & if \;\; b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_00) \oplus (00011011) \oplus (b_7b_6b_5b_4b_3b_2b_1b_0) & if \;\; b_7 = 1 \end{cases}$$

The inverse matrix is even uglier because it contains elements such as $0x$ where $x \geq 9$.

Briefly describe Add Round Key.

Recall that the key is $4 - 32$ bit words. and that the key block is arranged

$$k = \left[\begin{array}{cccc} w_0 & w_1 & w_2 & w_3 \end{array} \right]$$

where each word is a column of 32 bits. to write this as a square we just break the 32 bits into 8 bit bytes per row. Then we can just $\oplus$ the state with the key to get the next state:

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \oplus \begin{bmatrix} w_{0,0} & w_{1,0} & w_{2,0} & w_{3,0} \\ w_{0,1} & w_{1,1} & w_{2,1} & w_{3,1} \\ w_{0,2} & w_{1,2} & w_{2,2} & w_{3,2} \\ w_{0,3} & w_{1,3} & w_{2,3} & w_{3,3} \end{bmatrix}$$

Breifly describe the key expansion algorithm.

We start with a 16 byte (128 bit) key and perform the following: KeyExpansion(byte key[16], word w[44])

What is the difference between SubBytes and SubWord?

SubBytes performs takes a byte and performs the substitution using the S-Box. SubWord takes a word (4 bytes) and performs SubBytes on each byte in place.

What is the difference between ShiftRows and RotWord?

Nothing really except that Shift Rows really does shift a row, and the words are stored in a column which for RotWord we can view as a row.

What is the difference between the AES decryption algorithm and the equivalent inverse cipher?

Because Round 10 is different than the other rounds you can not just reverse the process. Plus you must use inverse S-box which is not the same as the original S-Box. similarly the SubBytes and MixCols are not there own inverse, thus the decryption can not be the same as the encryption.

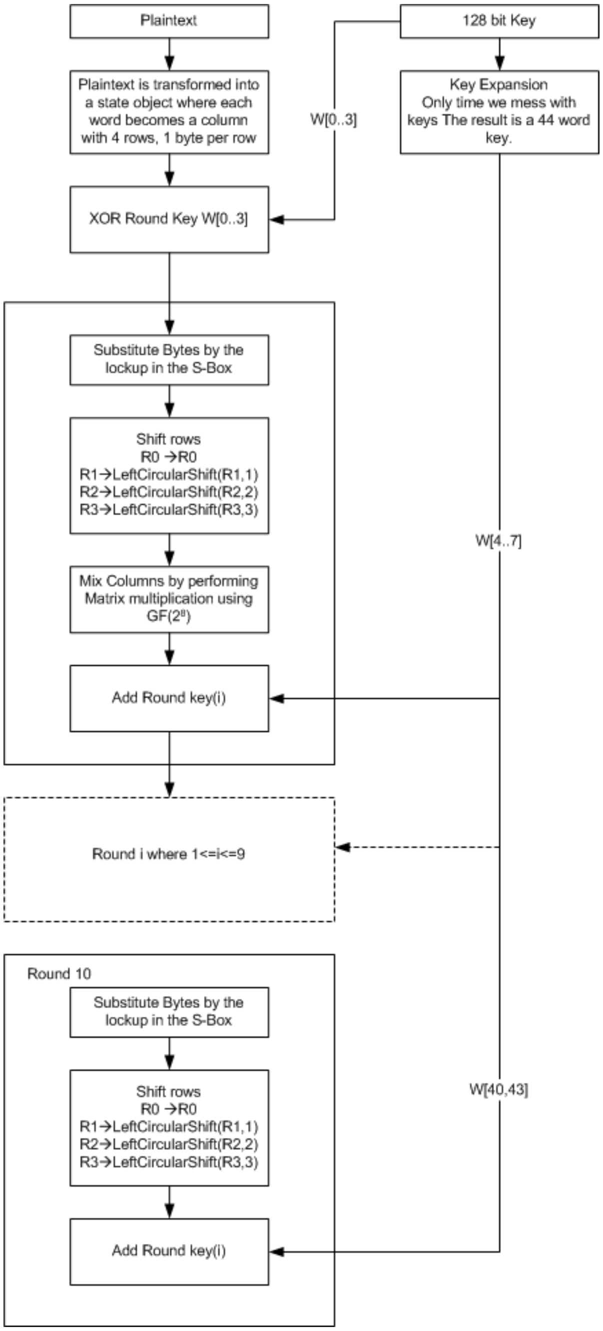
Plaintext
128 bit Key
Plaintext is transformed into a state object where each word becomes a column with 4 rows, 1 byte per row
W[0..3]
Key Expansion Only time we mess with keys The result is a 44 word key.
XOR Round Key W[0..3]
Substitute Bytes by the lockup in the S-Box
Shift rows
R0 →R0
R1→LeftCircularShift(R1,1)
R2→LeftCircularShift(R2,2)
R3→LeftCircularShift(R3,3)
W[4..7]
Mix Columns by performing Matrix multiplication using GF(2^8)
Add Round key(i)
Round i where 1<=i<=9
Round 10
Substitute Bytes by the lockup in the S-Box
Shift rows
R0 →R0
R1→LeftCircularShift(R1,1)
R2→LeftCircularShift(R2,2)
R3→LeftCircularShift(R3,3)
W[40,43]
Add Round key(i)

Figure 1: AES cipher diagram.

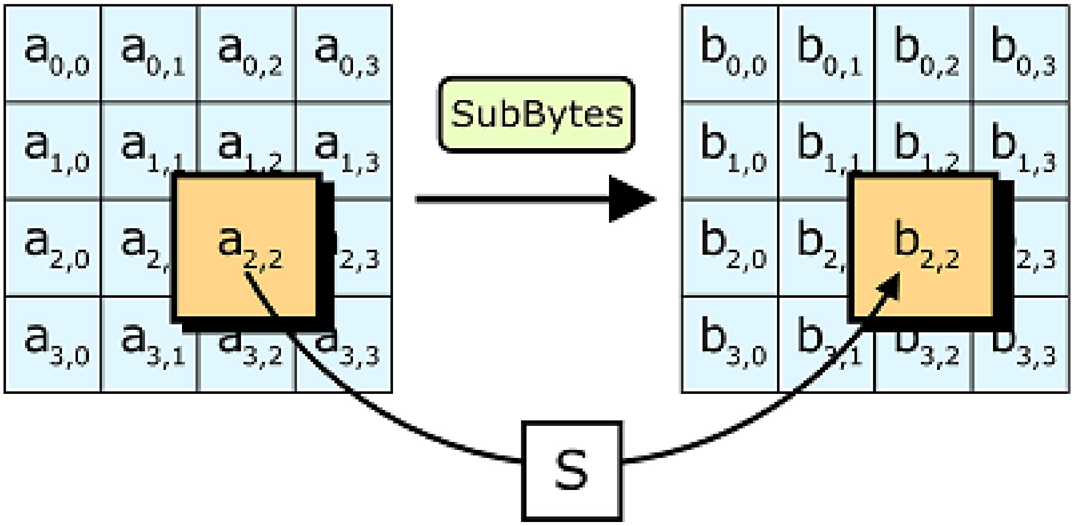
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S

Figure 2: S-Box substitution