The main principles of the AES cryptographic algorithm

Martin Rupp
SCIENTIFIC AND COMPUTER DEVELOPMENT SCD LTD

Contents

1 Introduction

AES stands for "Advanced Encryption Standard". This is a cryptographic algorithm primarily developed for and by normative bodies in the United States of America such as NIST. It is also often referred to as the algorithm *Rijndael* although the AES is actually a variant of the latter.

Historically, this standard replaced its precursor, DES or 'Data Encryption Standard'[1], when the latter was declared insecure, following advances in the field of cryptanalysis and especially in the field of miniaturization of processors.

[1] The DES is also known by its other name, the DEA (Data Encryption Algorithm) which in fact refers to all the algorithmic part of the encryption contained in the standard, but which in practice coincides with the latter.

Although DES can no longer be used, the 3-DES algorithm, still considered safe, remains widely used, particularly in the banking industry. 3-DES consists of an iteration of three DES ciphers. Usually cipher-decipher-cipher ('ede' mode).

AES has a number of similarities with DES. It also uses substitution tables. However, unlike DES, AES works by using computation over a finite field.

AES has spread into applications using cryptography. It is typically found in banking, military or government applications and in many 'consumer' applications that need secure encryption.

Computer scientists or computer programmers routinely implement or use applications based on the AES encryption functions but do not necessarily know its principle (and do not need to know it) and simply use AES encryption software libraries. Which libraries can offer guarantees or not concerning their fidelity to the encryption standard. Theoretically, only software libraries certified as conforming to the AES standard (and using a digital signature) should be used. If not, IT professionals implementing AES should be able to understand the validity or otherwise of an AES software library by studying and analyzing its source code.

In what follows, we will define the main theoretical foundations underlying AES and we will detail why it is considered a valid encryption algorithm over a relatively large cryptographic period.

2 Reminders on Galois groups

The AES uses computation on finite groups and particularly on Galois fields.
The algebraic definition of a field is assumed to be known to the reader. One can consult it in [1].

A Galois field is simply in mathematics, a finite field, that is to say a field provided with a finite number of elements. An elementary algebra theorem states that the order (ie the number of elements) of such a group can only be a power of a prime number p.

A Galois field is generally denoted $GF(q)$ with $q = p^n$, p is a prime number and q obviously is the order of the group.

Another fundamental algebra theorem states that all finite fields of the same order are isomorphic. $GF(p)$ is therefore isomorphic to Z/pZ where Z is the ring of relative integers. For $n \geq 1$ the representation of $GF(q)$ is more complex.(see [1])

The splitting field of the polynomial $P : P(X) = X^q - X$ (or $X \in GP(p)$)is a finite field (traditionally called a Galois group) and has exactly q elements. It is therefore $GF(q)$ (except for an isomorphism). A proof of the fact that this group exists whatever $n \geq 1$ can be found in [2] (§3, Theorem 3.3).

Many other constructions of $GF(q)$ are possible. However, in the context of AES, the construction using polynomials is the only one that is important. We will see in what follows how to construct $GF(q)$ as a field of polynomials whose degrees are strictly less than n.

3 Use of Galois groups within the framework of digital calculus

In order to be able to use the AES algorithm, we must for that, consider the algebraic field $GF(p)[X]$ of the polynomials on $GF(p)$.

In addition, in the context of digital computation and data representation, we need to work with the binary basis. This is possible if we consider the particular case where $p = 2$.

Using finite field arithmetic, and using an algebraic basis for $GF(p)[X]$ (considered as a vector space over $GF(2)$), we can represent the elements of $GF(2^n)$ as binary numbers.

For example the polynomial $P(X) : P(X) = X^7 + X^3 + 1$ has for coordinates $(1, 0, 0, 1, 0, 0, 0, 0, 1)$. which gives the following binary representation:

10001001

That is to say the (natural) number *137*.

It is therefore possible to define a multiplication and an addition on a certain finite subset of natural numbers $N_q = \{0, 1, .., q - 1\}$, which will thus define an isomorphism between N_q and $GF(q)$. the basic idea is that, if $n = 8$, then N_q represents the set of hexadecimal numbers and therefore allows one byte of information to be represented.

To explicitly construct $GF(q)$ in this case, we must consider the ideal (P) generated by the polynomial $P : P(X) = X^q - X$.

$GF(q)$ is then defined as the quotient space: $GF(p)[X]/(P)$

P can be replaced by any other irreducible polynomial in $GF(p)[X]$ provided it is primitive and minimal.

Concretely, there is no “universal" method to construct $GF(q)$ and it is a difficult problem when n is important. A general construction method exists for the case $n = 2$ for example but not for any value of n.

However, as we indicated, AES only needs to know $GF(2^8)$. We will therefore explicit the construction of this field.

4 Construction of GF(2^8)

In this precise case, P is chosen as the polynomial defined by $P(X) = X^8 + X^4 + X^3 + X + 1$.

A construction method is to consider a "symbolic" root α of this polynomial.
This root is a generator of the group and thus, the elements of $GF(2^8)$ can be written as a power of this generator:

$GF(2^8) = \left\{1\alpha..., \alpha^{255}\right\}$

These elements are all polynomials with binary coefficients of degree at most equal to 7. It is therefore easy to see that their total number is indeed equal to 2^8. It is not necessarily obvious that the order of $GF(p)[X]/(P)$ is equal to p and that the elements describe the set of polynomials of $GF(p)[X]$ of degree strictly less than not. We can find a proof in ([2]).

There is a duality between the value of $\alpha^s, 1 \leq s \leq 255$ and a polynomial of $GF(p)[X]/(P)$.

	Polynomial	**value Binary**	**value Numeric value (base 10)**
α	X	10	2
α^2	X^2	100	4
α^3	X^3	1000	8
α^4	X^4	10000	16
α^5	X^5	100000	32
α^6	X^6	1000000	64
α^7	X^7	10000000	128
α^8	$X^4 + X^3 + X + 1$	00011011	27
α^9	$X^5 + X^4 + X^2 + X$	00110110	54
α^{10}	$X^6 + X^5 + X^3 + X^2$	01101100	108
α^{11}	$X^7 + X^6 + X^4 + X^3$	11011000	216
α^{12}	$X^7 + X^2 + X$	10110000 $+$ 00110110 $=$ 1000 0110	134
..		etc ...	...

The elements of $GF(2^8)$ are therefore calculated successively using the formula $\alpha^8 = \alpha^4 + \alpha^3 + \alpha + 1$. (we are on the base field $GF(2)$ so any coefficient which is not zero is equal to 1!)

It is obviously possible to calculate all the 256 elements of the Galois field without having to resort to a computer.

Once the elements of the field are known, we need to define the operations of addition and multiplication. Due to the binary nature of the base body, these are greatly simplified compared to other finite fields.

5 Calculation on GF(2^8)

The addition is terribly simple. Let be two polynomials represented by, respectively

$P = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ and $Q = (y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7)$.
Then we will have $P + Q = (x_0 \wedge y_0, x_1 \wedge y_1, x_2 \wedge y_2, x_3 \wedge y_3, x_4 \wedge y_4, x_5 \wedge y_5, x_6 \wedge y_6, x_7 \wedge y_7)$
Where $\wedge$ denote the operation *XOR (actually the addition over Z/2Z but it's a more natural way of looking at it in this context)* .

Multiplication is, on the other hand, more complicated.

$PQ = R$ with: $P(X)Q(X) = R(X)Mod(X^8 + X^4 + X^3 + X + 1)$

That is, R is the remainder of the Euclidean division of PQ by the polynomial $X^8 + X^4 + X^3 + X + 1$.

These operations now allow us to perform encryption (and therefore also decryption) on bytes by representing them by their associated polynomials.
Note that AES is only one of the many algorithms using that technique.

As an example consider the following two polynomials:

$P : P(X) = X^7 + X^3 + 1$ and $Q : Q(X) = X^5 + X^4 + X^3 + 1$

We have: $P + Q = S : S(X) = X^7 + X^5 + X^4$

$P(X)Q(X) = X^{12} + X^{11} + X^{10} + X^7 + X^8 + X^7 + X^6 + X^3 + X^5 + X^4 + X^3 + 1$
$P(X)Q(X) = X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + 1$
$P(X)Q(X) = (X^4 + X^3 + X^2)(X^8 + X^4 + X^3 + X + 1) + (X^6 + X^5 + X^4 + X^2 + 1)$

This means that $P.Q = R : R(X) = X^6 + X^5 + X^4 + X^2 + 1$

If we consider these polynomials as numbers, we get:

10001001+00111001 = 10110000
10001001**.**00111001 = 01110101

Which gives us, in the end:

137+57 = 176
137**.**57 = 117

As always in finite fields, calculating the inverse of a number is a particularly non-trivial and absolutely non-linear operation. It is this property which is used mainly in the AES algorithm to guarantee a higher security.

6 The AES

6.1 Architecture

The encryption / decryption algorithm mainly uses the principles of confusion and diffusion (substitution and permutation). It is a symmetric encryption algorithm, which means that the same key is used for encryption and for decryption. The algorithm works only on bytes and not on bits like the DES.

The heart of the security of the algorithm, and what makes it safe, consists in the use of substitution tables, the "S-Box". As with DES, these tables are designed to maximize the difficulty of a cryptanalysis. However, and this is the big difference with DES, these substitution tables use the inversion in the Galois group $G(2^8)$.

The AES encryption key is used to generate a number of subkeys by deriving from the initial key; these subkeys will encrypt or decrypt data at each round.

AES uses a number of successive rounds over a block of operations.
This basic block of operations is itself made up of 3 operations:

- Substitution;
- Shifting of lines;
- Mixing Columns ;

For encryption.

- Inverse substitution;
- Inverse Shifting lines;
- Inverse Mixing Columns;

For decryption.

Depending on whether one is in encryption or in decryption , these operations will not be done in the same order. In addition, the derived keys will be introduced during the course of the rounds.

The introduction of the derived keys is in itself a fourth operation.

The data is encrypted in blocks of 128 bits, i.e. 16 bytes. Each block of data is formatted as a square of bytes of size 4x4.

This square is called the state. The state is transformed as and when the transformations it undergoes: substitution, shifting of rows or mixing of columns as well as during the introduction of derived keys. This is obviously the principle of all the encryption and all of the decryption of the AES: transforming the state so that the block of data is encrypted or decrypted. If we name $f_i, i = 1...N$ the successive encryption operations, we have the successive states $C = C_0, C_1 = f_1(C_0)...., C_N = f_N(C_{N-1})$. $C_N = E$ where E is the ciphering of C.

If the starting block to be encrypted or decrypted is represented by bytes $a_1...a_{16}$, the state $C = C_0$ will be defined by:

a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}
a_4	a_8	a_{12}	a_{16}

After the N successive operations of the AES the state $E = C_N$ will have a certain value

a_1'	a_5'	a_9'	a_{13}'
a'_2	a_6'	a_{10}'	a_{14}'
a_3'	a_7'	a_{11}'	a_{15}'
a_4'	a_8'	a_{12}'	a_{16}'

And the encryption / decryption of the starting block of bytes is simply defined by:

$a_1...a_{16} \to a'_1...a'_{16}$

6.2 Substitution

The substitution is based on a substitution table named S-BOX.

A substitution table has the goal to “break" any possible linearity.

The construction of substitution tables is often done using Boolean functions called *bent functions*. These functions have the particularity of being at a maximum distance from the set of linear Boolean functions.

6.2.1 Reminder on the bent functions

If d is a distance for the numerical functions on the vector space $GF(2)^n$: $f : GF(2)^n \to N$, we define the numerical function $r : f \to \sum_{g \in L} d(f, g)$ where L is the subspace of the digital linear functions of $GF(2)^n$.

We define the set of curves C functions $f \in C \Rightarrow r(f) < r(f) f \notin$C.

Usually d is defined as the Hamming distance.

$d : d(f, g) = card\,\{x \in GF(2)^n, f(x) \neq g(x)\}$.

There are equivalent definitions of bent functions. For example, f is a bent function if the Fourier transform $F(\lambda)$ of $x \to -1^{f(x)}$is such that: $|F(\lambda)| = 1, \lambda \in GF(2^n)$ or if the Walsh-Hadamard transform of f is such that: $|w_f(\omega)| = 2^{n/2}$. This transform makes it possible to measure the correlation between f and the linear functions.

Bent functions are often the "heart" of the complexity of symmetric cipher algorithms. Indeed, they are designed to resist linear cryptanalysis since they cannot be approximated by linear functions and they are therefore used to construct the S-boxes (substitution tables) which will carry out the diffusion of the bytes in a chaotic manner.

Given their fundamental character in cryptography, they are often the subject of in-depth studies. There are several techniques for constructing such functions. The most classic is the Maiorana MCFarland technique (see [3] for a complete overview on the use of bent functions in cryptography).

6.2.2 The AES substitution table

The AES does not use bent functions for substitution operations, it mainly uses the inversion on $GF(2^8)$ (which is of course different from a Boolean function which would be defined using a formula on $GF(2)^8$!)

This inversion[2] is combined with an affine operation as follows:

1. $x \to x^{-1} (= x^{255}), x \in GF(256)^*$ (and $0^{-1} = 0$)
2. $x \to A.x + b$

Where A is the following matrix of GL (GF (256)):

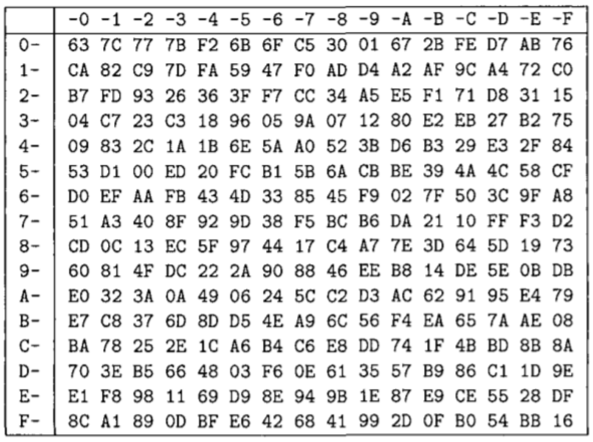
$$\begin{matrix}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{matrix}$$

[2]The inversion is understood with the convention that the inverse of 0 is 0.

And b is the number 63 in GF (256), that is to say (0,1,1,0,0,0,1,1)

The substitution table of the AES is therefore relatively simple, which has often been criticized, but this simplicity ensures, finally, a great robustness.

The substitution table can therefore be represented as follows:



	-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

Each input byte is broken down into two nibbles: respectively nibble of the 4 most significant bits and 4 least significant bits. The output value is read using the two nibbles as, respectively, column and row.

For example the byte 0x43 is transformed into the byte 0x18, which corresponds to the value of the table for row 3 and column 4.

6.3 Rows shifting

Substitution (confusion) is generally not sufficient. This is because it does not distribute the data evenly. For that, it is necessary to have recourse to operations of diffusion. This diffusion can be carried out by permutation operations, which can use permutation tables or P-BOX. These tables allow compression and expansion. For example, they can broadcast the result of an S-BOX output to r inputs of another S-BOX, making r copies each time, where r is any number.

The formula for the row offset is very simple, it is:

$S_{i,j} \rightarrow S_{i,ji}$ where $S_{i,j}, 0 \leq (i,j) \leq 3$ is an element of the report. ji is of course to be taken in the context of modulo 4 operations.

This operation is represented graphically as follows:

a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}
a_4	a_8	a_{12}	a_{16}

$\rightarrow$

a_1	a_5	a_9	a_{13}
a_6	a_{10}	a_{14}	a_2
a_{11}	a_{15}	a_3	a_7
a_{16}	a_4	a_8	a_{12}

6.4 Column mixing

The column mixing operation is a little more complex than that of row shifting.

Each column of the state is seen as a vector $C_j, j = 0, .., 3$ of $GF(256)^4$. The linear transformation defined by the following matrix is applied to it:

$$\begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix}$$

This linear transformation will linearly mix each of the columns of the report using only its own elements.

In fact, this linear transformation can also be written:

$S_{i,j} = 2S_{i,j} + S_{i+1,j} + S_{i+2,j} + 3S_{i+3,j}$

Attention: The calculation is to be understood in GF (256).

6.5 Generation and insertion of derived keys

6.5.1 Generation of derived keys

Until now we have not considered the operations implementing the encryption key. It is obviously necessary that the encryption key be introduced during the operations described above.

To do this, the AES uses an algorithm for deriving the keys which are introduced as the rounds are carried out. It is about a “planning" of use of the keys named the *AES key schedule.*

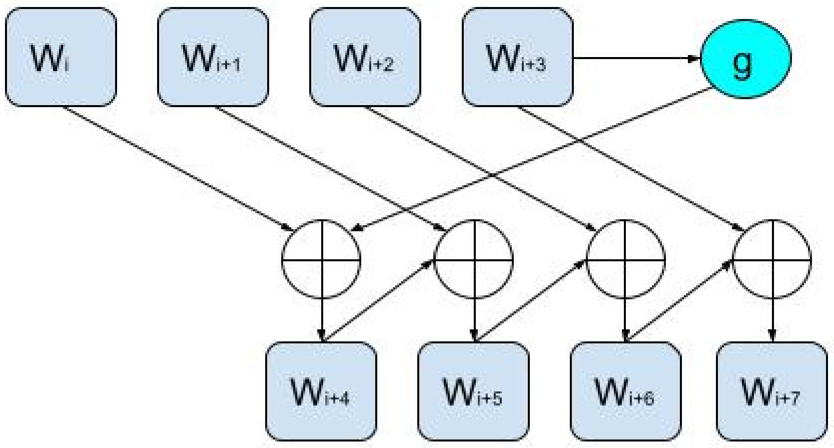
It is useful to remember that the AES only allows three key lengths: respectively 128, 196 and 256 bits.

Let us take the case of a key of length 128 bits. The 16 bytes of the key$K_1, ..., K_{16}$ are distributed in a box in the same way as for the report.

K_1	K_5	K_9	K_{13}
K_2	K_6	K_{10}	K_{14}
K_3	K_7	K_{11}	K_{15}
K_4	K_8	K_{12}	K_{16}

Each of the columns defines a word and thus the key can be represented by a series of 4 words: $w_0, w_1, w_2.w_3$.

The 4 words are extended to a series of 44 words $w_0, w_1, w_2.w_3, w_4, ..., w_{43}$ as follows:



The diagram above recursively calculates the 44 words from the initial sequence $w_0, w_1, w_2.w_3$.

The XOR operation is represented in the diagram by the usual symbol $\oplus$.
The value $g = g(w_{i+3})$ is calculated as follows:

1. A one-byte rotation to the left.
   $w_{i+3} = (a_0, a_1, a_2, a_3) \rightarrow (a_1, a_2, a_3, a_0)$
2. The substitution of the 4 bytes via the S-BOX substitution table of the AES which has been described previously
3. Finally, an XOR operation of the leftmost byte with a certain constant, called the round constant. This constant is determined by the following formula:

$RC_i = 2^{i-1}$ in GF (256). We give the table of its values below.

i	1	2	3	4	5	6	7	8	9	10
RC_i	01	02	04	08	10	20	40	80	1B	36

6.5.2 Insertion of the derived keys

Once the 44 words have been generated (in the case of a 128-bit key), you must insert them into the cryptographic stream so that, of course, without knowing this key it is impossible to correctly encrypt / decrypt a message.

The integration of the keys is very simple: the 44 words form a series of 11 groups of 4 words, and each time a key is inserted (represented by 4 words in a row), the state is transformed by an operation of or exclusive (XOR) with the 4 words[3].

[3]This is a one-time pad since both parts have an equal length of 16 bytes

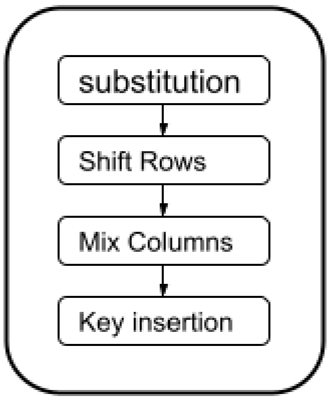
$C_{i-1} \oplus (w_{4i}, w_{4i+1}, w_{4i+2}, w_{4i+3}) \rightarrow C_{i-1}$ where C_i is the state in the i$^{\text{th}}$ round.

If we consider different key lengths, the principle remains the same, except that there are more words generated and therefore more rounds.

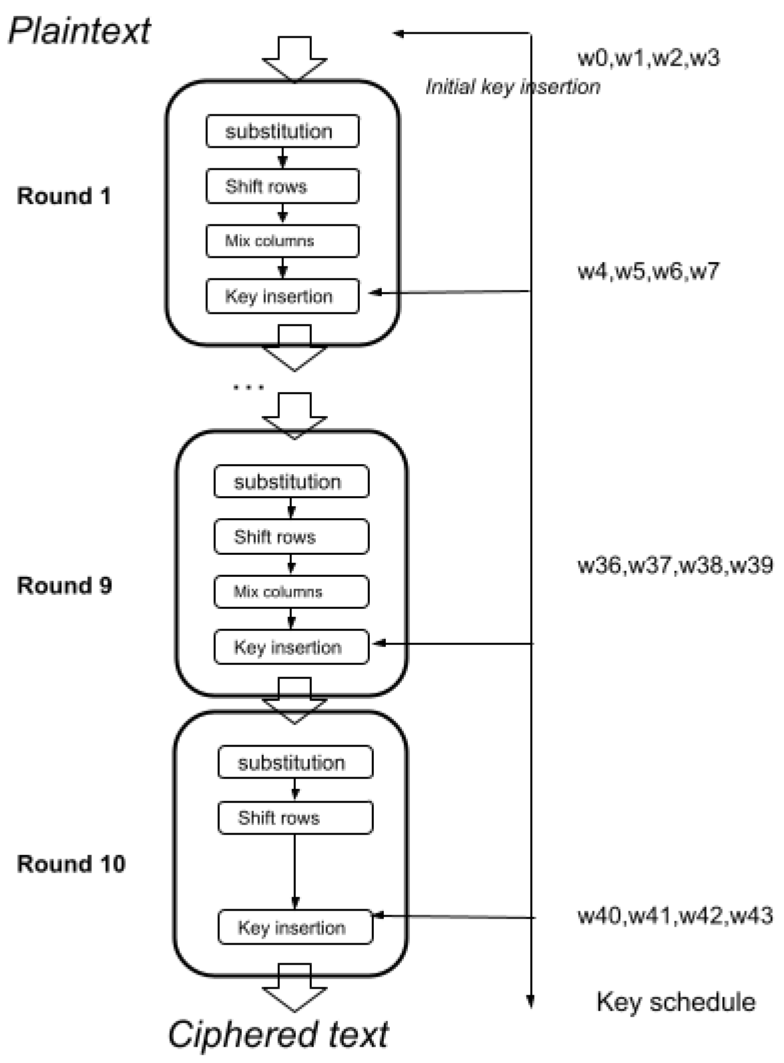
7 Complete Structure of AES

7.1 Encryption

We can now represent the complete diagram of operation of AES in the case of a 128-bits encryption key. For the case of 196-bits and 256-bits keys, the pattern remains similar, with the difference in the number of rounds which is greater.



Each round of the AES is done with the same “box" which contains the operations we have seen above, except for the last round where there is no column shift.
In the case of a 128-bit key, there are 10 rounds and 11 key inserts.



7.2 Decryption

All the operations that we have seen are invertible:

- All the matrices considered are invertible (they have a non-zero determinant)
- The inversion on a body is obviously an invertible operation (it is even involutive)
- The operations of or exclusive (XOR) are also invertibles (involutive)

It is therefore possible to follow the opposite path to find, from an encrypted text, the original text. The operation of mixing columns is omitted in the last round, which is justified by the desire to give the decryption a design similar to the encryption. Not only would this not affect the security of the AES, but it would make it more resistant to certain types of attacks (see [5]).

The decryption scheme therefore consists in using the encryption scheme but 'backwards'.

8 Security of AES

Unlike DES, AES uses finite group computation which means that the mathematical properties intrinsic to modular computation will be added to the diffusion and confusion properties to reinforce the encryption security. This makes this encryption algorithm a somewhat “hybrid" mechanism.

The key planning system also prevents the appearance of weak keys, which is an advantage over DES.

Ultimately, AES appears as a “reinforced" DES and which corrects the flaws of the latter, for example, with better substitution tables.

At this moment in time, AES is considered safe. Its design and the length of the encryption keys give it better security than DES. For the moment, it has withstood all the attacks proposed by cryptanalysts and in particular attacks using statistical analysis. However, with advances in cryptanalysis, advancements in quantum computers, and the use of artificial intelligence and neural networks for cracking ciphers, it will eventually become obsolete one day or another.

But this is another story.

9 Bibliography

[1] Andre Weil (Author), Basic Number Theory (Grundlehren der mathematischen Wissenschaften) 3rd ed. 1973 Edition

[2] Serge Lang, Algebra, 2002, Springer-Verlag
[3] Natalia Tokareva, Bent Functions. Results and Applications to Cryptography, Elsevier
[4] Algebraic Aspects of the Advanced Encryption Standard, Carlos Cid, Sean Murphy, Matthew Robshaw, Springer
[5] The Effects of the Omission of Last Round’s MixColumns on AES, Orr Dunkelman and Nathan Keller