

Advanced Encryption Standard

História:

Em 1997, o *NIST* (*National Institute of Standards and Technology*) nos Estados Unidos anunciaram que estavam à procura de um novo algoritmo de encriptação para se tornar no *Advanced Encryption Standard* (*AES*). Criptógrafos de todo o mundo foram convidados para fazer propostas para o novo *AES*.

Quinze algoritmos foram submetidos no total e foram alvo de debates, pesquisas e análises. Alguns foram considerados pouco seguros ou pouco eficientes. No final de todo o processo, um algoritmo – *Rijndael* – ganhou a competição e foi eleito o conhecido *AES*.

Atualmente, o *AES* está em todo lado: Encripta discos rígidos, comunicações de internet e muitos processadores de computadores e telemóveis e muitos outros dispositivos, trazem instruções de fábrica para executar o *AES* de forma eficiente.

Como funciona? O que o faz um bom algoritmo de encriptação?

Encriptação

Quando se fala em criptografia, fala-se em transformar um *plaintext* (texto normal) e pegar numa *secret key* (chave secreta) e transformá-lo num *ciphertext* (texto cifrado).

Um texto deve ser cifrado de tal forma que seja necessário a chave para o transformar de volta em texto normal.

Em termos computacionais, um *plaintext*, *ciphertext* e *secret key* são vistos como uma sequência de *bits*: 0's e 1's

Um exemplo simples de encriptação é utilizar o *Exclusive Or (XOR)* da lógica Booleana.

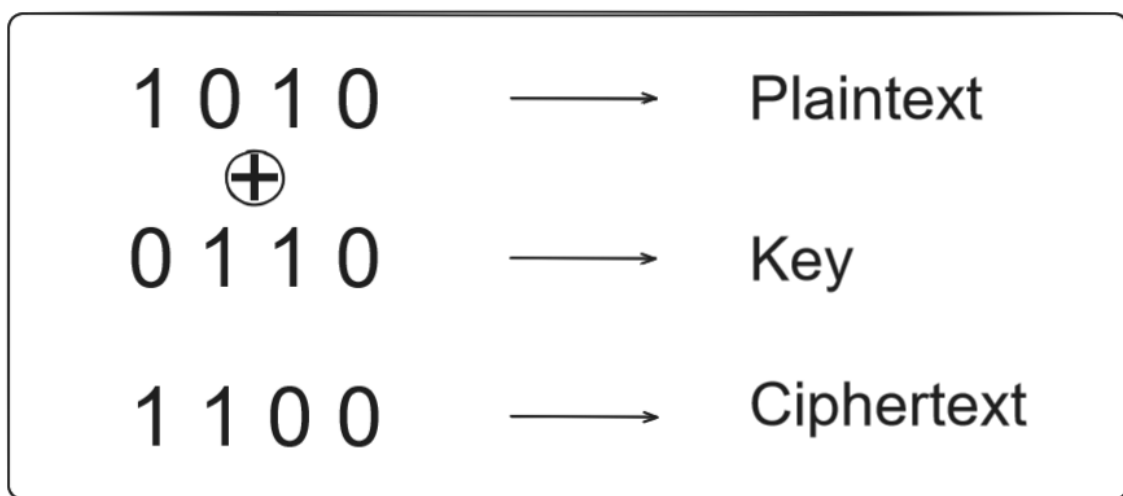


Figura 1 - Alteração de bits com XOR (Exclusive Or)

Este é um bom exemplo de encriptação segura. Se um adversário obtivesse o *ciphertext* não conseguiria ver nada relacionado com o conteúdo do *plaintext* original. Mas um problema maior pode surgir:

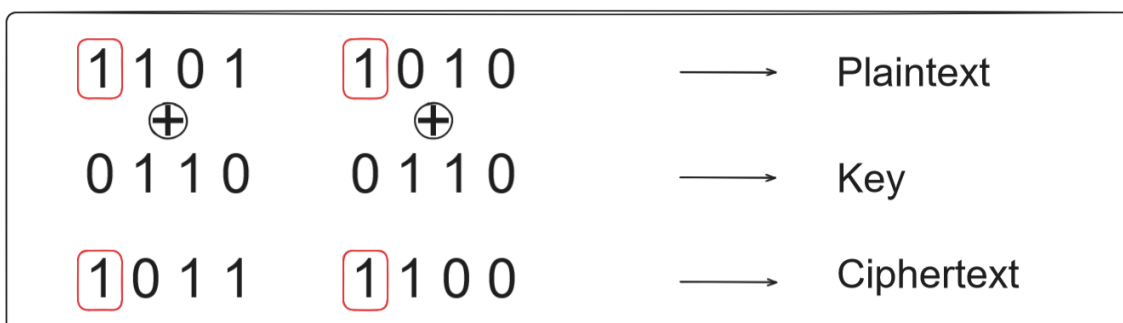


Figura 2 - Textos diferentes cifrados com a mesma chave podem originar resultados semelhantes

Se a mesma *key* for utilizada várias vezes em *plaintexts* diferentes, pode originar resultados semelhantes de *ciphertexts*. No exemplo acima mostra que os dois *ciphertexts* têm o primeiro *bit* igual, como usam a mesma *key* é possível verificar que o primeiro *bit*

do *plaintext* também é igual. Essa informação pode ser o suficiente para fazer uma análise estatística para adivinhar o conteúdo original do *plaintext*.

Para este algoritmo ser seguro a chave só pode ser utilizada uma vez, e é por isso que esta abordagem é conhecida como uma cifra *one-time pad* (cifra de uso único).

Neste exemplo trabalhamos com um texto e uma chave com o mesmo tamanho. Se o nosso texto for mais longo do que a nossa chave ou precisarmos da chave para encriptar vários pedaços de texto, então precisámos de uma abordagem diferente.

Então que propriedades deve ter uma boa cifra?

Confusão e Difusão

Duas propriedades importantes para a criptografia foram originalmente pensadas por um cientista da computação - Claude Shannon - nos anos 40.

Confusão trata de complicar a relação entre a *key* e o *ciphertext*. No exemplo do *Exclusive Or* a relação entre a chave e o resultado é óbvia. Uma alteração de um *bit* da chave resulta na alteração de um *bit* no texto cifrado na mesma posição.



Figura 3 - Um bit do Ciphertext depende de um bit da Key

O ideal é garantir que cada *bit* do *ciphertext* dependa de vários *bits* da chave para que a sua relação seja complicada de analisar.

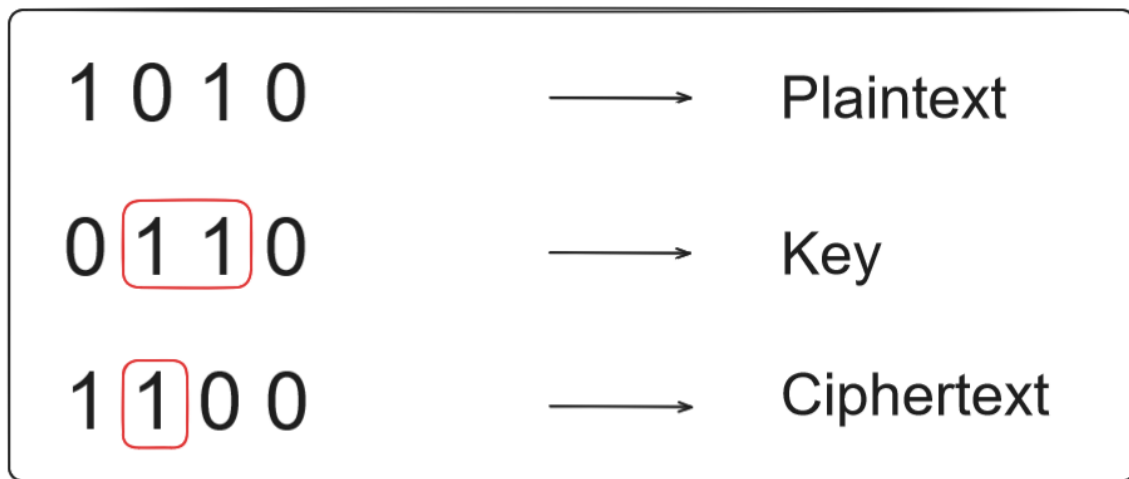


Figura 4 - 1 bit do Ciphertext depende de vários bits da Key

Difusão trata de separar ou afastar os padrões estatísticos da estrutura do *plaintext*. Os textos têm padrões estatísticos, em inglês por exemplo:

“THE PATTERN IN THE MESSAGE” → Letras variam na frequência com que aparecem.

“THE PATTERN IN THE MESSAGE” → Algumas palavras são mais populares do que outras.

Se estes padrões aparecerem numa cifra, então um adversário seria capaz de os analisar também e possivelmente chegar ao texto original. Então, para criar uma cifra segura, a informação do *plaintext* deve ser distribuída igualmente pelo *ciphertext*. Na prática, uma cifra com difusão deve funcionar de tal forma que a alteração de um *bit* do *plaintext* altere cerca de metade dos *bits* do *ciphertext*.

Uma boa cifra deve conter ambos, **confusão** e **difusão**. O AES não é exceção.

O Advanced Encryption Standard é conhecido por ser uma cifra de blocos, isto é, pega num bloco de *plaintext* de um certo tamanho e retorna um bloco de *ciphertext* do mesmo tamanho. Trabalha com blocos de 128 *bits* (16 *bytes*).

Então começa-se com 16 *bytes* de informação que queremos encriptar:



Figura 5 - 16 Bytes de informação a encriptar

O AES pega nos 16 *bytes* e agrupa numa tabela de 4x4 conhecida como o “*state*” do algoritmo:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Figura 6 - State do AES

Ao longo do algoritmo, são feitas alterações até obtermos o nosso bloco encriptado.

O que é feito para encriptar este bloco?

Chave (*Adding the key*)

Voltando atrás, o quê que a nossa cifra com o *Exclusive Or* necessitaria aqui? Uma chave, e neste caso uma chave com um tamanho de 16 *bytes*. Realiza a operação *Exclusive Or* e a esse processo chama-se de “*Adding the key*” (*adicionar a chave*) ao *state*.

Plaintext		Key
1		1
5		5
9		9
13		13
2		2
6		6
10		10
14		14
3		3
7		7
11		11
15		15
4		4
8		8
12		12
16		16

Figura 7 - Adding the key ao state

Agora é preciso adicionar confusão ao processo. Relembrando que confusão é fazer com que o *ciphertext* dependa de várias partes da chave em vez de apenas uma. Uma forma de adicionar a confusão é pegar na chave e expandi-la em várias chaves. Há várias formas de o fazer, mas o AES especifica um processo particular que começa com a chave original e vai determinado os *bytes* da *expanded key* (*chave expandida*) ao combinar os valores dos *bytes* da chave original.

Pode ser adicionar valores constantes, trocar os *bytes* de sítio, puxar *bytes* para trás ou para a frente,

O objetivo é obter um conjunto de chaves expandidas em que os *bits* das chaves expandidas dependem dos *bits* da chave original.

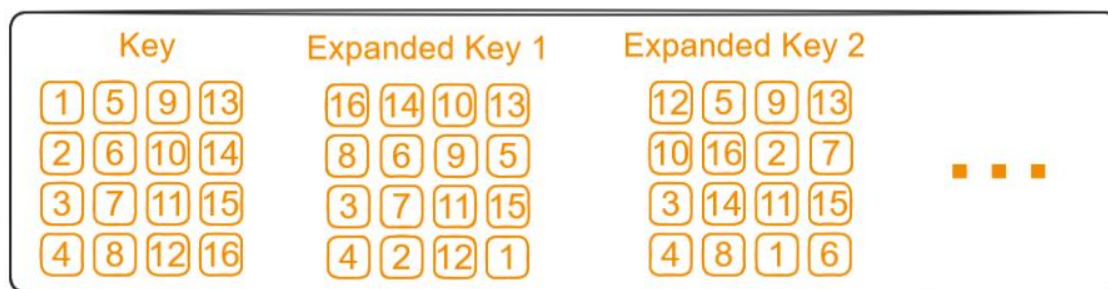


Figura 8 - Key Expansion

Com isto, em vez de aplicarmos apenas uma chave, vamos aplicar várias chaves em que em cada ronda vai-se aplicar uma *round key*:

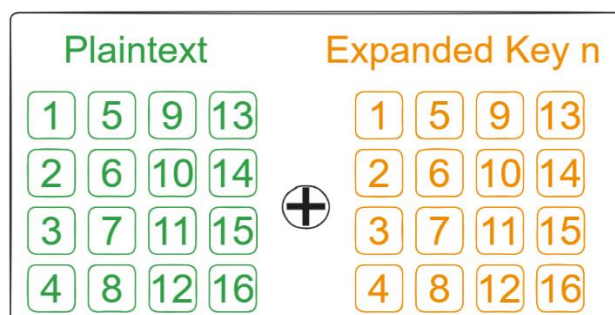


Figura 9 - 1ª Round key

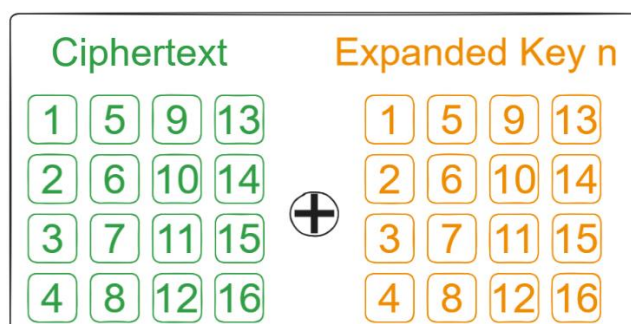


Figura 10 - nª Round Key

Se pensarmos bem, adicionar múltiplas chaves não aumenta nada na segurança, pelo que adicionar múltiplas chaves seria o mesmo que adicionar apenas uma chave que seria o resultado de adicionar todas as chaves juntas. Mas a ideia de adicionar várias *round keys* pode dar jeito quando começarmos a adicionar outros passos no processo.

Que mais pode ser aplicado aqui?

Substituição

Um tipo simples de cifra é a cifra de substituição. Numa cifra de substituição segue-se um padrão de substituição, isto é, substitui-se bocados de texto por outros bocados de texto. Como por exemplo a “Cifra de César” ou “ROT13”.

Ambas as “Cifra de César” e “ROT13” baseiam-se em substituir umas letras por outras de forma que um adversário não consiga ler a mensagem original.

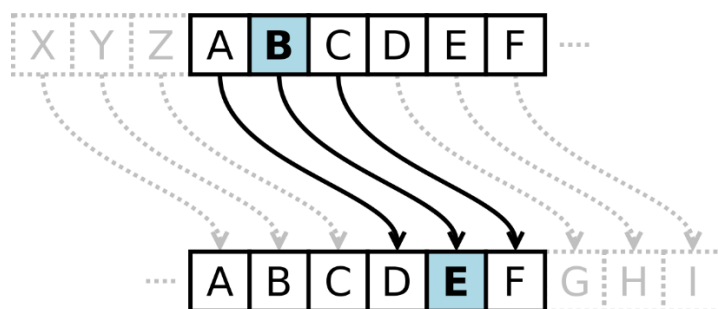


Figura 11 - Cifra de César

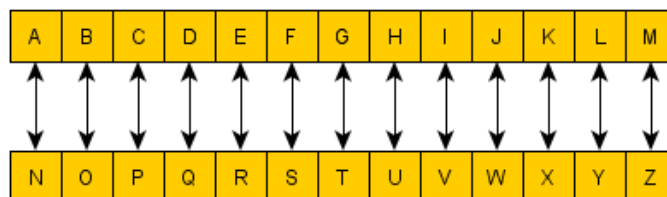


Figura 12 - ROT13 (Rotate by 13)

Exemplo:

Plaintext: "O inimigo está perto"

ROT13: "b vavzvtb rfgn cregb"

Como é possível verificar, estas cifras são fáceis de analisar estatisticamente e de as quebrar (descobrir a mensagem original).

Para evitar esta relação tão simples o AES utiliza uma substituição chamada de *SubBytes* - substituição de *bytes* – ou seja, em vez de substituir letras por outras, cada *byte* é substituído por um *byte* diferente.

(row index) (column index) S-Box GF (2')

hex	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	e5	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	4d	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	e7	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	fe	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	60	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	0e	13	ee	5f	97	44	17	e4	a7	7e	3d	64	5d	19	73
9	60	81	4f	6e	22	2a	90	88	46	ee	b8	14	6e	5e	05	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	6e	56	f4	ea	65	7a	be	08
c	ba	78	25	2e	1e	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	cf	b0	54	bb	16

16 times

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34
32	c4	c7	e1
1a	be	c7	9a
42	04	46	c5
c2	5d	3a	18

Figura 13 - Tabela de substituição de bytes

Então, quando o AES faz a substituição de *bytes*, percorre cada *byte* no *state* e troca pelo seu correspondente como mostra na tabela da Figura 13. Tal como nas letras, onde queríamos evitar escolher uma relação simples para a substituição, a substituição do AES foi pensada para ser uma expressão algébrica complexa e não apenas uma substituição linear de *bytes*.

Apesar de a substituição ser complexa, é também de conhecimento público, então o AES não poderia depositar a sua confiança apenas nestes passos visto que qualquer um poderia simplesmente reverter a substituição de *bytes* e obter o texto original.

Em vez disso, é combinado a substituição de *bytes* com adição de uma *round key* um certo número de vezes.

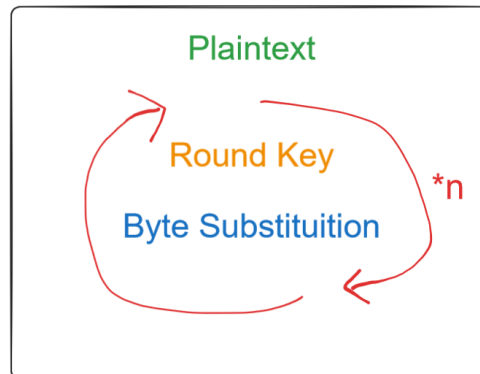


Figura 14 - Round Key + SubByte

Adicionar apenas *round keys* não faz grande coisa para a segurança. Adicionar apenas a substituição de *bytes* também não aprimora em nada o algoritmo. Mas combinando os dois juntos resulta numa **confusão** maior. Cada *bit* do *ciphertext* sofre várias alterações a partir da *round key* e ainda sofrem uma substituição, e a reversão não é óbvia visto que não têm acesso às chaves que dependem todas da chave original. Mas ainda não está completo.

Então que mais falta no algoritmo?

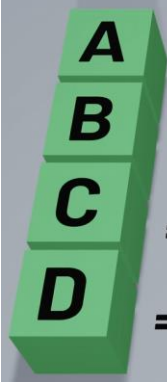
Até agora temos a criação de Expanded Keys, a aplicação da confusão (Byte Substitution e Add Round Key) mas falta ainda a difusão, ou seja, espalhar a informação do *plaintext* no *ciphertext*.

O ideal seria que quando um *bit* é alterado, um conjunto de *bits* também seja alterado (de preferência cerca de metade), mas até agora o nosso algoritmo ainda não faz isso. O que está a acontecer é a alteração de *bits* que resulta na versão encriptada dos *bytes*, mas não fazem qualquer impacto nos outros *bytes* vizinhos.

Desta forma, o AES precisava de arranjar uma forma de espalhar a informação de uns *bytes* para outros *bytes* com queda para aumentar ainda mais a confusão. Para atingir o seu objetivo o AES utiliza duas novas operações: *Shift rows* (*mudar linhas*) e *Mix Columns* (*misturar colunas*).

O papel do *MixColumns* é misturar os valores dos *bytes* em cada coluna, ou seja, difundir informação pelos valores.

Opera independentemente em cada coluna e em cada coluna há uma expressão particular para combinar os 4 valores da coluna.



$$\begin{aligned}
 A &= 2 \times A + 3 \times B + 1 \times C + 1 \times D \\
 B &= 1 \times A + 2 \times B + 3 \times C + 1 \times D \\
 C &= 1 \times A + 1 \times B + 2 \times C + 3 \times D \\
 D &= 3 \times A + 1 \times B + 1 \times C + 2 \times D
 \end{aligned}$$

Figura 15 – MixColumns

Para já o algoritmo mistura a informação de cada coluna com cada coluna independentemente. O que queremos também é misturar a informação entre as colunas (de umas para as outras) e é então que antes de fazer o *MixColumns* é feito o *ShiftRows*.

O *ShiftRows* altera a forma como as linhas são apresentadas, ou seja, muda as posições dos *bytes*, por exemplo:

a 1ª linha é deixada intacta;

a 2ª linha pega no 1º *byte* e passa-o para último lugar;

a 3ª linha pega nos dois primeiros *bytes* e troca com os dois últimos *bytes*;

a 4ª linha pega nos três primeiros *bytes* e passa-os para último lugar.

ShiftRows

1	5	9	13
6	10	14	2
11	15	3	7
16	4	8	12

Figura 16 – ShiftRows

Estes dois passos juntos, *ShiftRows* e *MixColumns*, ajudam a misturar os valores do *state* difundindo assim a informação. Os valores ficam ainda mais misturados quando este processo acontece durante várias rondas.

O Algoritmo completo

Agora que temos todos os passos falta apenas juntá-los.

O AES consiste em:

1. *Key Expansion*
2. *Add Round Key*
3. x N Rounds
 - a. *Substitute Bytes*
 - b. *Shift Rows*
 - c. *Mix Columns* (Não é feito na última ronda)
 - d. *Add Round Key*

Quantas rondas de transformação é que o algoritmo utiliza?

O número de rondas do algoritmo depende do tamanho da chave original. O AES foi desenhado para trabalhar com vários tamanhos de chaves:

- *Chaves de 128 bits → 11 Round Keys → 10 Rounds → AES-128*
- *Chaves de 192 bits → 13 Round Keys → 12 Rounds → AES-192*
- *Chaves de 256 bits → 15 Round Keys → 14 Rounds → AES-256*

Quanto maior a chave mais difícil é de quebrar o algoritmo por *brute force*.

Bibliografia

https://www.youtube.com/watch?v=C4ATDMLz5wc&ab_channel=SpanningTree