Criptografia Simétrica Moderna

Introdução às Redes de Substituição e Permutação (ou Cifras de Feistel)

Enno Nagel

Minicurso no IFAL – Maceió, 1 Dezembro 2018

Criptografia Outrora

A criptografia estuda a transformação de um

texto inteligível

$$\Downarrow$$

texto *in*inteligível

tal que só uma informação adicional secreta, a *chave*, permite desfazê-la.

Substituição (do Alfabeto por traslado das letras)

Este método foi usado por César (63 - 14 a.C.).

- Escolhemos uma *chave*

  δ = uma distância entre letras em ordem alfabética

  = um número entre 0 e 25

- trasladamos por δ cada letra do alfabeto (latino e cujas letras são arranjadas em um anel).

Por exemplo, se $\delta = 3$, então

$$A \mapsto D, B \mapsto E, C \mapsto F, ..., W \mapsto Z, X \mapsto A, ..., Z \mapsto C.$$

e, por exemplo, para $\delta = 3$,

$$CARA \mapsto FDUD$$

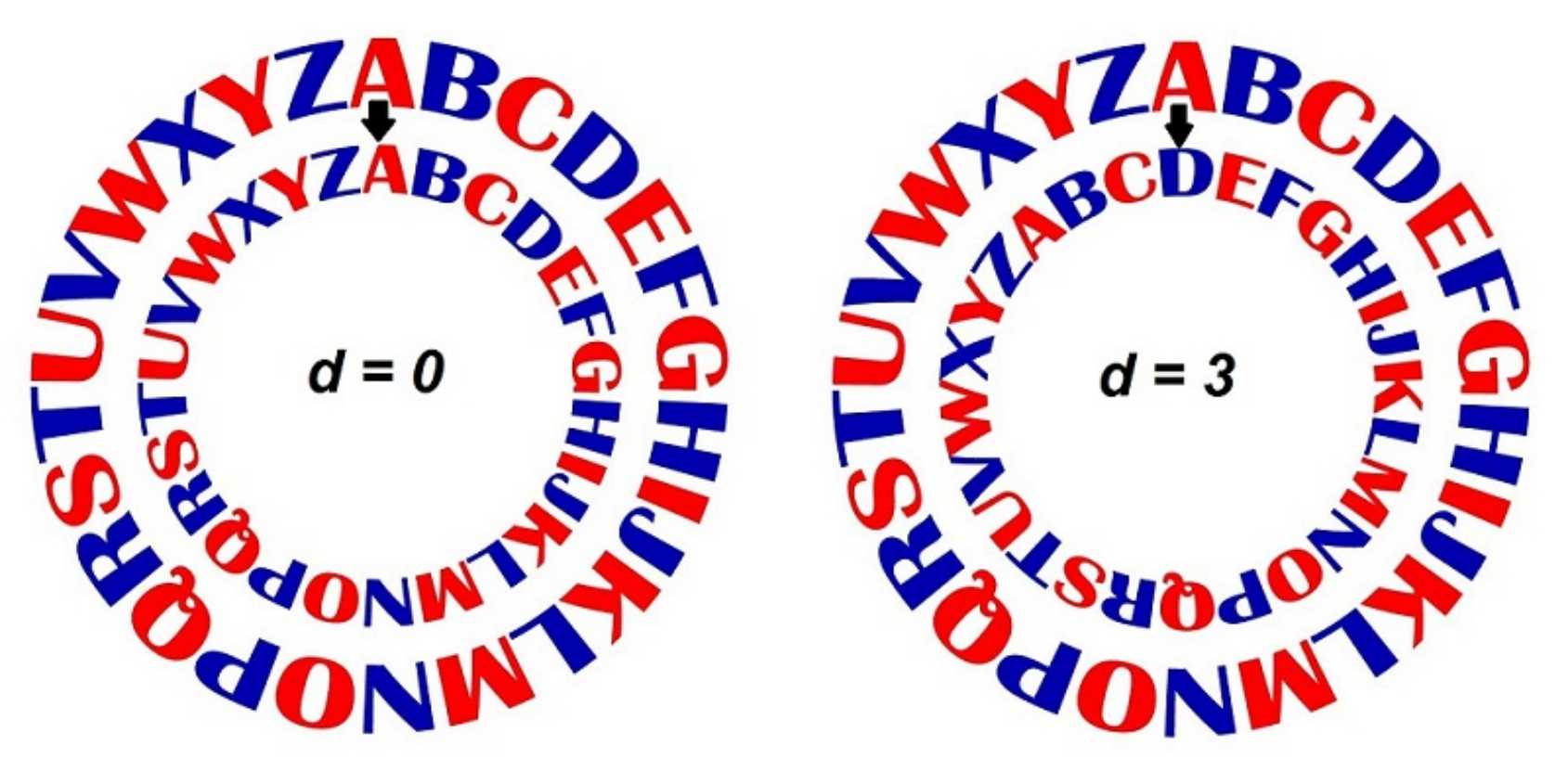


Figura 1: César como duas rodas deslocadas

Substituição (do Alfabeto por permutação das letras)

Em vez de

- substituirmos cada letra por um traslado pela mesma distância δ,
- substituímos cada letra por uma letra arbitrária; isto é, permutamos as letras. Por exemplo,

A	B	...	Y	Z
↓	↓	...	↓	↓
E	Z	...	G	A

Por exemplo,

$$\mathrm{ABA} \mapsto \mathrm{EZE}.$$

Há $26 \cdot 25 \cdots 1 = 26! > 10^{26}$ chaves = permutações.

Transposição (das letras do Texto Claro)

A *cítala* ou o *bastão de Licurgo* (= legislador de Esparta) é um bastão com o qual os espartanos cifravam como segue:

- enrolar o bastão com um pano estreito,
- escrever neste pano no sentido da borda maior, e
- desenrolar o pano do bastão.

As letras assim transpostas no pano unicamente podiam ser decifradas por um bastão com

- a mesma circunferência (e o mesmo comprimento),

pela mesma maneira como o texto foi cifrado:

- enrolar o bastão com um pano,
- ler este pano no sentido da borda maior.

A chave é o número dado pelo **número das letras que cabem nesta circunferência**.

Figura 2: A cítala enrolada por um cinto de couro

Exemplo da Cítala

Por exemplo, se o bastão tem uma circunferência de 2 letras (e um comprimento de 3 letras), as duas linhas

l u a
m e l

tornam-se as três linhas

l m
u e
a l

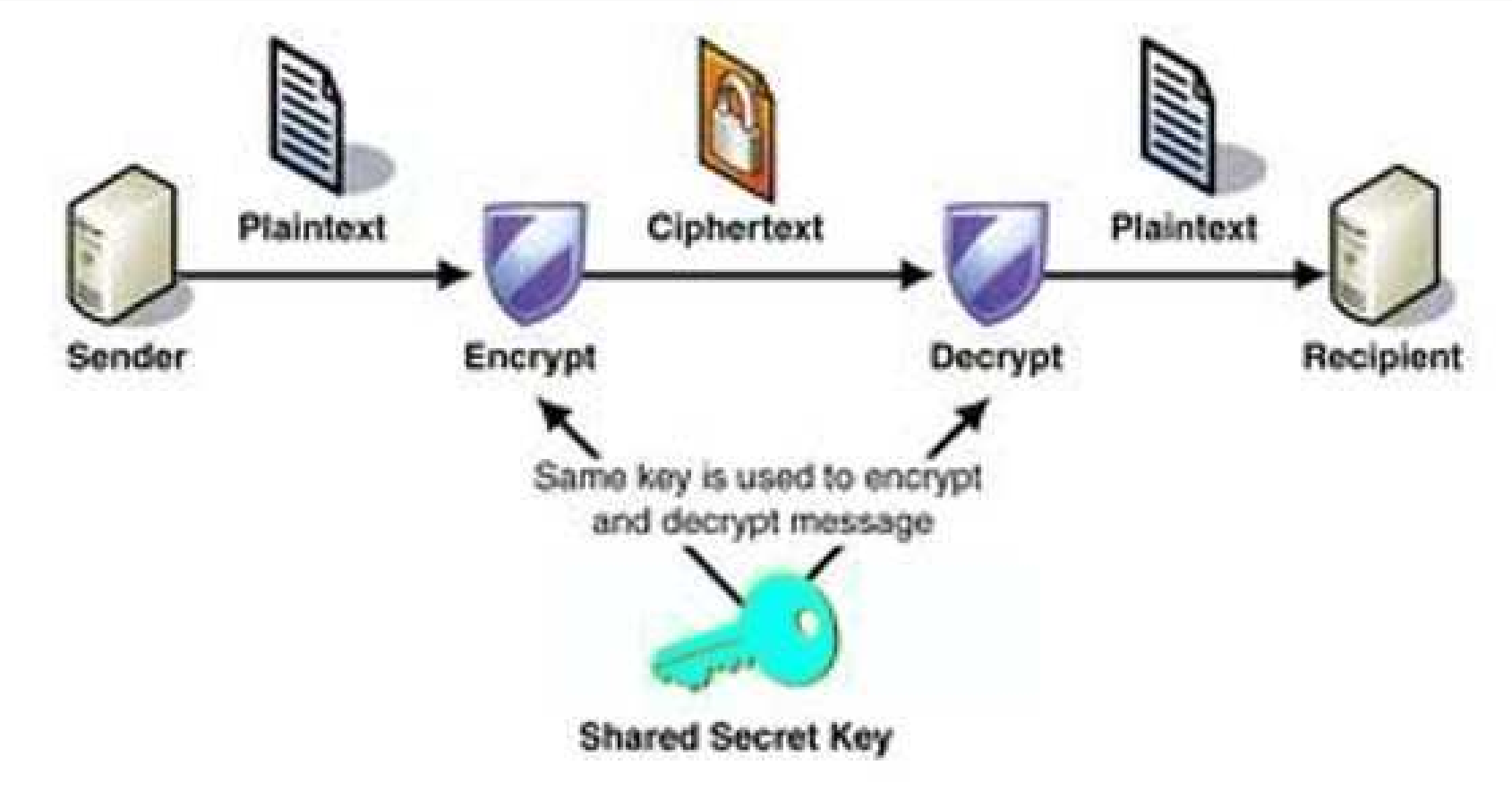
que são concatenadas (para não revelarem nem a circunferência, nem o comprimento) à linha

L M U E A L

Criptografia Simétrica

A criptografia é **simétrica** se

chave para cifrar = chave para decifrar.



A criptografia simétrica já era usada pelos egípcios 2000 anos antes de Cristo, e todos os exemplos históricos são simétricos.

Qualidades Desejáveis de um Algoritmo Criptográfico

O *princípio de Kerckhoff* postula que o

- algoritmo seja **público**.

Enquanto o conhecimento da chave compromete uma única cifração, o conhecimento do algoritmo compromete todas as;

$\Longrightarrow$

Um algoritmo público garante a dificuldade da decifração depender só da segurança da *chave*, mas não da do *algoritmo*. Quanto mais usado, tanto mais provável que o algoritmo será conhecido. Para ele ser útil, necessita ser seguro sendo público.

As *metas de Shannon* da

- *Confusão* respectivamente
- *Difusão*

desejam **minimizar** a **relação** entre o texto cifrado e

- a **chave** respectivamente
- o **texto claro**

Idealmente, quando uma letra do texto claro respectivamente da chave muda, cada letra do texto cifrado muda com uma probabilidade de *50%*.

$\Longrightarrow$

Uma boa confusão ou difusão evita ataques estatísticas sobre

- ou muitas *chaves* cifrando o mesmo texto claro,
- ou muitos *textos claros* cifrados pela mesma chave.

Falhas da Substituição do Alfabeto

A *Substituição do Alfabeto* por permutação

A	B	…	Y	Z
↓	↓	…	↓	↓
E	Z	…	G	A

tem $26 \cdot 25 \cdots 1 = 26! > 10^{26}$ chaves,

$\Longrightarrow$ ataque de força bruta computacionalmente inviável.

Mas viola os *princípios de difusão e confusão*. Se a chave (= permutação do alfabeto) permuta a letra α pela letra β, então há

- má *confusão* porque a troca de β na chave implica unicamente a troca da letra β no texto cifrado,
- má *difusão* porque a troca de α no texto claro implica unicamente a troca da letra β no texto cifrado.

Ataque pela Frequência Estatística

Com efeito, permite ataques estatísticas sobre a frequência de

- letras,
- bigramas (= pares de letras) e
- trigramas (= triplos de letras).

em português. Por exemplo,

- a letra mais frequente em português é “a”
- o bigrama mais frequente em português é “de”
- o trigrama mais frequente em português é “que”

$\Longrightarrow$ se há suficientemente texto, associando

- a letra mais frequente do *texto cifrado* à letra mais frequente *em português* (= “a”),
- o bigrama mais frequente do *texto cifrado* ao bigrama mais frequente *em português* (= “de”), …

Falhas da Cítala

A cítala viola

- o *princípio de Kerckhoff*, que o algoritmo seja público.

Com efeito, o valor máximo da circunferência é $< n/2$ onde n = o número das letras do texto cifrado. Por isso, um ataque de força bruta, é viável.

Ela tem

- má *difusão* porque a substituição de uma letra α no texto claro implica unicamente a substituição da mesma letra α no texto cifrado.

Com efeito, o algoritmo permite ataques estatísticas sobre a frequência de

- bigramas (= pares de letras)
- trigramas (= triplos de letras), e assim por diante.

Criptografia Moderna

A criptografia estuda a transformação de

dados inteligíveis

$$\Downarrow$$

dados *in*inteligíveis

tal que só uma informação adicional secreta, a **chave**, permite desfazê-la.

Dados

Outrora (na época análoga):

dados = textos.

Hoje (na época digital):

dados = arquivo digital (de texto, imagem, som, vídeo, ...)
= sequência de bits (= 0,1)
= sequência de bytes (= 00, 01, ..., FE, FF)
= número (= 0,1,2,3 ...).

Codificação de Textos

Texto inglês (= sequência de letras latinos)

A codificação **`ASCII`** cobre todas as letras inglesas (além dos símbolos de pontuação, por exemplo) e envia um símbolo a um byte. Por exemplo,

$$\mathrm{A} \mapsto 65, ..., \mathrm{Z} \mapsto 90; a \mapsto 97, ..., z \mapsto 122.$$

Texto Internacional (= sequência de símbolos)

A codificação **`UTF-8`** inclui a `ASCII` e cobre todas as letras de todos os idiomas (por exemplo, chinês, coreano, ... e além disso por exemplo todos os símbolos matemáticos).

UTF-8

Ela envia um símbolo a 1, 2, 3 ou até 4 bytes, onde

$$\#\{\text{bytes}\} = \#\{\text{dígitos bin. consecutivos iniciais iguais a } 1\} + 1$$

Por exemplo, o símbolo ç, a letra c com uma cedilha, tem 2 bytes e

$$\text{ç} \mapsto 1011001111000111.$$

Com efeito, o número dos primeiros dígitos binários consecutivos inicias iguais a 1 é 1 (o segundo dígito já sendo igual a 0), o que acrescentado por 1 resulta em 2, o número de bytes de ç.

Codificação de Imagens (por um *bitmap* = mapa de graus das cores primárias)

Uma imagem

- é um conjunto de pixeis (por exemplo, 1024×768),
- cada pixel é uma cor, e
- cada cor é os seus graus de intensidade da luz das três cores primárias
  - Vermelho (= *R*ed),
  - Verde (= *G*reen) e
  - Azul (= *B*lue).

  O homem não consegue distinguir mais de 256 graus de cada cor primária,

  $\Longrightarrow$ basta codificar cada pixel por três bytes.

Mapa de graus das cores primárias (RGB = Vermelho, Verde e Azul)

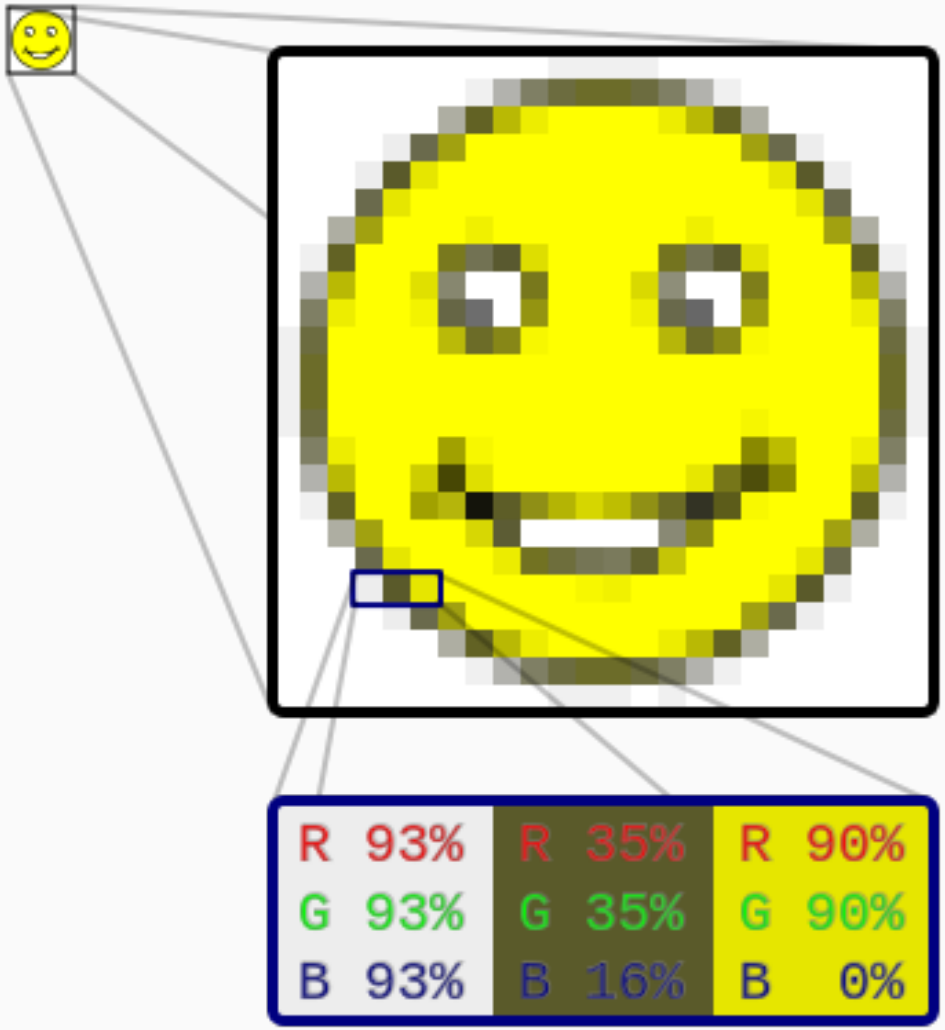


Figura 3: bitmap

One Time Pad

O `One-time pad` adiciona (pela operação `XOR`, isto é, ou exclusivo, a disjunção exclusiva) cada byte do texto claro t com o byte (na posição) correspondente de uma chave c que

- é do mesmo comprimento, e
- é descartada após uso, isto é, não cifrará outros textos claros.

Isto é, o texto cifrado $\mathrm{T} = (\mathrm{T}_1, \mathrm{T}_2, ...)$ é

$$\mathrm{T} = t \oplus c = (t_1 \oplus c_1, t_2 \oplus c_2, ...).$$

Este método de cifração é **tão seguro quanto teoricamente possível** (se o texto claro tem um único bloco t)!

Porém, se o texto claro tem mais de um, por exemplo, *dois* blocos, t' e t'' então, com este algoritmo, a soma (XOR)

$$\mathrm{T}' \oplus \mathrm{T}'' = (t' \oplus c) \oplus (t' \oplus c) = t' \oplus t''$$

dos dois blocos **cifrados** T' e T'' é a soma $t' \oplus t''$ dos dois blocos **claros** porque $x \oplus x = 0$ para cada bit x!

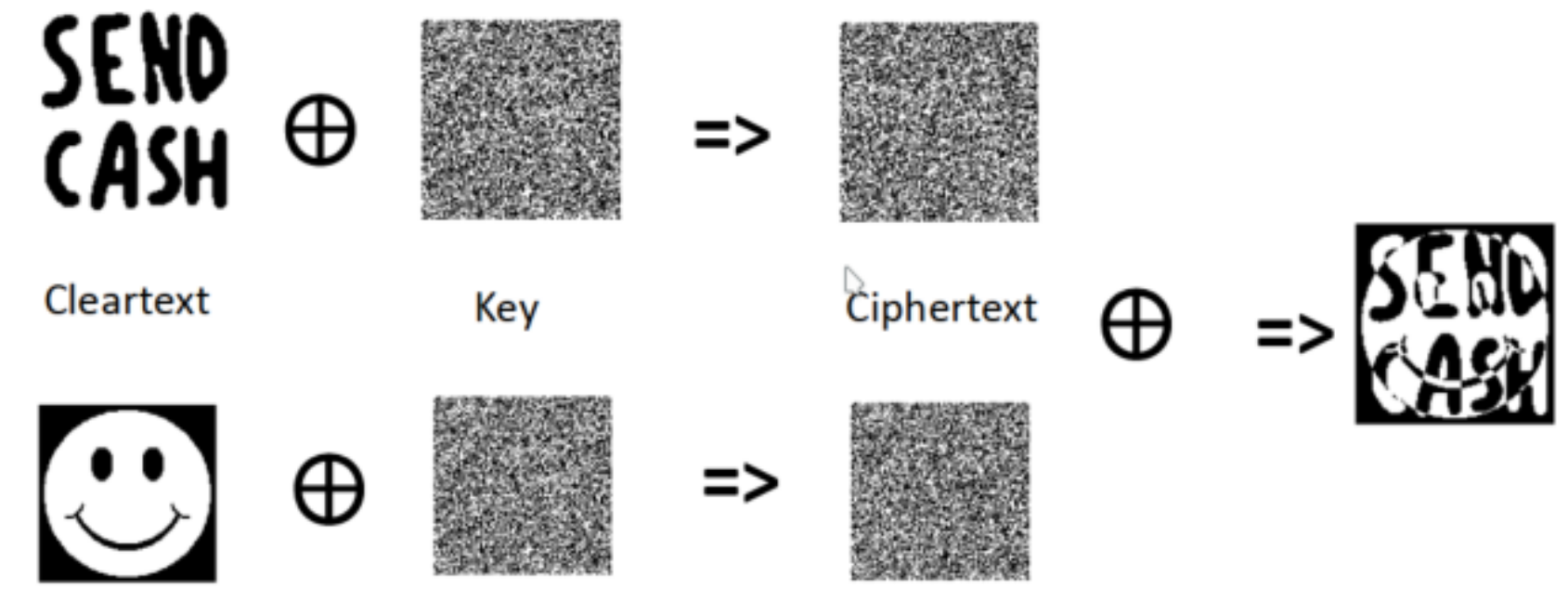


Figura 4: A soma de dois textos claros revela regularidades!

Por isso, é mais seguro aplicar após a adição da chave uma substituição do bloco por outro. **Idealmente**, a rodada consistiria unicamente

- na adição da chave e
- na substituição do bloco **inteiro** por outro.

Porém, o alfabeto desta substituição seria **gigantesco**: Para um bloco de 16 bytes, esta tabela de substituição teria

$$2^{256} \cdot 16 > 10^{79}$$

bytes. Por isso, o AES substitui só cada byte, cada casa do bloco; uma tabela de substituição de $2^8 = 256$ entradas de 1 byte; em seguida, permuta as casas.

Estas operações se complementam tão bem que são quase tão seguros como uma substituição do bloco inteiro. Isto é, a permutação imita a substituição gigantesca da melhor maneira.

Cifra de Feistel

Uma *Cifra de Feistel* ou uma *rede de substituição e permutação* (= SPN = Substitution Permutation Network)

- agrupa o texto (= sequência de bytes) em blocos de n bytes (no AES $n = 16$ e no modelo prototípico $n = 2$) e
- itera (10 vezes no AES e 5 vezes no modelo prototípico) os três seguintes passos:
    1. adição (XOR) da chave,
    2. substituição do alfabeto (que opera em sub-blocos do bloco), e
    3. permutação (entre os sub-blocos do bloco).

Cifra de Feistel Prototípica segundo Heys

Usamos o algoritmo de Heys (2002) como modelo simples de uma cifra de Feistel que

- divide o texto claro em blocos de 16 bits, e
- subdivide cada bloco em 4 blocos de 4 bits,

e em cada uma das primeiras rodadas número 1, 2 e 3:

A tabela origina do algoritmo DES e e comumente chamada de *S-box*, caixa de substituição.

1. Adiciona a chave da rodada (para cada rodada, tem uma chave correspondente independente) ao bloco, $\mathrm{B} \mapsto \mathrm{B} \oplus \mathrm{C}$.
2. Substitui cada um dos 4 sub-blocos de 4 bits pela tabela

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

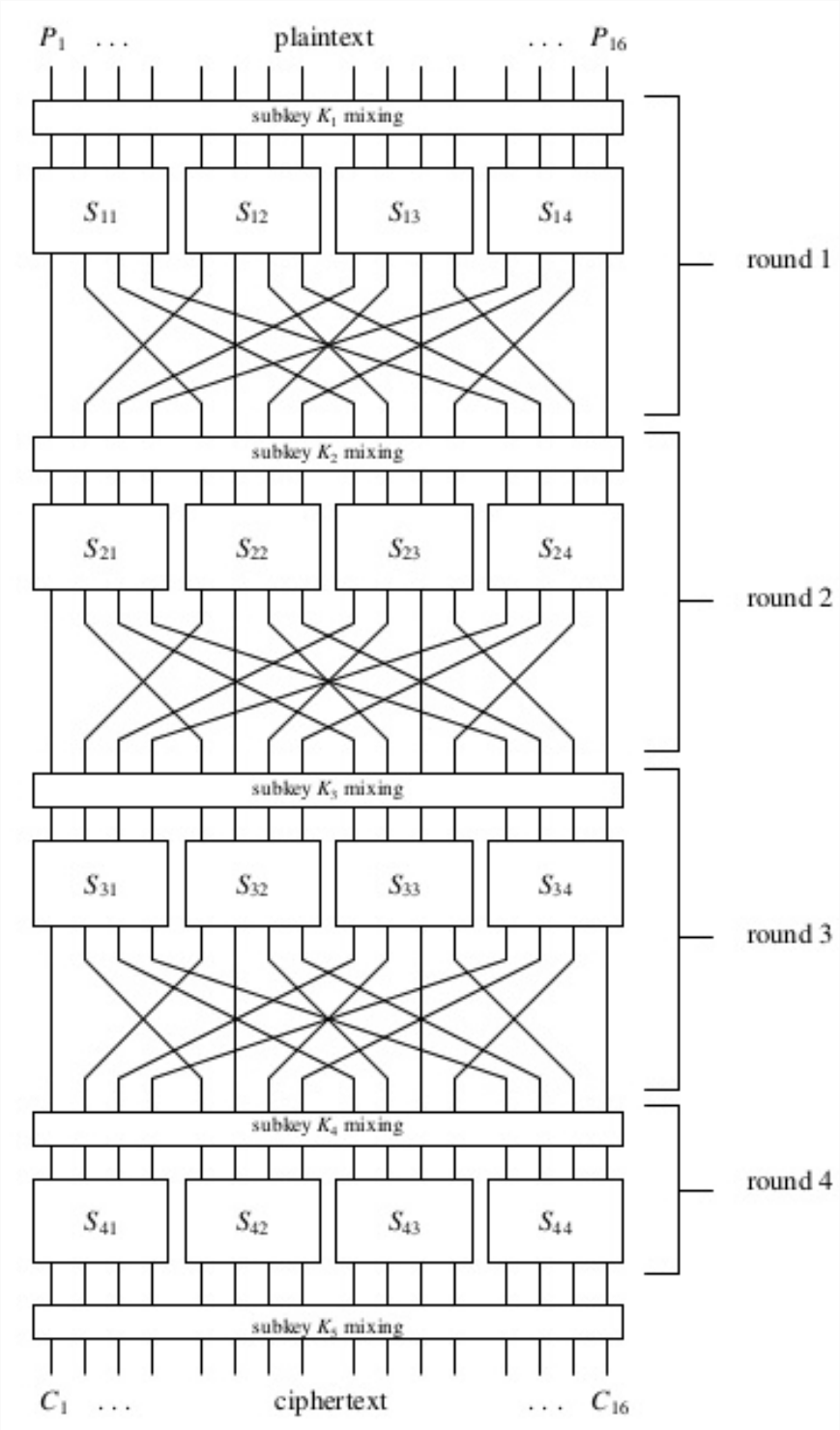
3. Troca o bit i do sub-bloco j com o bit j do sub-bloco i;

Na penúltima 4^{a} rodada

1. Adiciona a chave da rodada ao bloco, $\mathrm{B} \mapsto \mathrm{B} \oplus \mathrm{C}$.
2. Substitui cada um dos 4 sub-blocos de 4 bits pela tabela.

Na última 5^{a} rodada

1. Adiciona a chave da rodada ao bloco, $\mathrm{B} \mapsto \mathrm{B} \oplus \mathrm{C}$.


P_1 . . . plaintext . . . P_{16}
subkey K_1 mixing
S_{11} S_{12} S_{13} S_{14}
round 1
subkey K_2 mixing
S_{21} S_{22} S_{23} S_{24}
round 2
subkey K_3 mixing
S_{31} S_{32} S_{33} S_{34}
round 3
subkey K_4 mixing
S_{41} S_{42} S_{43} S_{44}
round 4
subkey K_5 mixing
C_1 . . . ciphertext . . . C_{16}

Competição

O algoritmo AES é o vencedor de uma competição anunciada pelo National Institute of Standards and Technology of the United States (NIST) de 1997 a 2000 para substituir o então padrão de criptografia simétrico, o Data Encryption Standard (DES).

Antes de tornar-se o AES, este algoritmo foi denominado pelos seus criadores Vincent Rijmen e Joan Daemen Rijndael, pelas letras iniciais dos seus sobrenomes.

No final da competição, restaram estes cinco algoritmos com as seguintes votações:

- `Rijndael`: 86 votos positivos, 10 negativos
- `Serpent`: 59 votos positivos, 7 negativos
- `Twofish`: 31 votos positivos, 21 negativos
- RC6: 23 votos positivos, 37 negativos
- MARS: 13 votos positivos, 84 negativos

Entre eles, nenhum deles se destacou pela sua maior segurança, mas o `Rijndael`, sim, pela sua simplicidade, ou clareza, e em particular economia computacional, na implementação. Como este algoritmo será a rodar por toda parte, por exemplo, nos processadores minúsculos de 8 bits nos cartões inteligentes (`smartcards`), a decisão foi tomada em favor do `Rijndael`.

Até hoje, este algoritmo continua firme e forte. Por exemplo, para cifrar uma rede sem fio, a opção mais segura é o AES.

Figura 6: Cifração de uma rede sem fio pelo AES

Cifração em Blocos

O algoritmo AES agrupa o texto claro (e as chaves) em retângulos de $4 \times \mathrm{B}$ bytes onde

$$\mathrm{B} := \text{ número das colunas do bloco } = 4, 6 \text{ ou } 8.$$

Comumente, e para nós de agora para diante, $\mathrm{B} = 4$, isto é, os retângulos são quadrados. Em base hexadecimal (= cujos dígitos são $0 - 9$, $\mathrm{A} = 10$, $\mathrm{B} = 11$, $\mathrm{C} = 12$, $\mathrm{D} = 13$, $\mathrm{E} = 14$ e $\mathrm{F} = 15$), um tal quadrado tem por exemplo a forma

A1	13	B1	4A
A3	AF	04	1E
3D	13	C1	55
B1	92	83	72

Rodadas

O AES cifra cada bloco iterativamente, em rodadas. Seja

$$\mathrm{R} := \text{ o número de rodadas}$$

que depende de B, há

- $\mathrm{R} = 10$ rodadas para $\mathrm{B} = 4$ colunas,
- $\mathrm{R} = 12$ rodadas para $\mathrm{B} = 6$ colunas
- $\mathrm{R} = 14$ rodadas para $\mathrm{B} = 8$ colunas.

Então, para nós, $\mathrm{R} = 10$. Nestas rodadas, são geradas chaves, o texto claro substituído e transposto pelas seguintes operações:

1. Rodada $r = 0$:
   - `AddRoundKey` para adicionar (por `XOR`) a chave ao quadrado
2. Rodadas $r = 1, ..., \mathrm{R} - 1$ para cifrar, aplicando as seguintes funções:
   2.1 `SubBytes` para substituir cada casa do quadrado por uma sequência de bits melhor distribuída,
   2.2 `ShiftRows` para transpôr as casas das linhas do quadrado,
   2.3 `MixColumn` para adicionar as casas das colunas do quadrado entre elas,
   2.4 `AddRoundKey` para gerar uma chave a partir da chave da rodada anterior e adicioná-la (por `XOR`) ao quadrado.
3. Rodada $r = \mathrm{R}$ para cifrar, aplicando as seguintes funções:
   3.1 `SubBytes`
   3.2 `ShiftRows`
   3.3 `AddRoundKey`

No CrypTool 1 no Menu Individual Procedures -> Visualization of Algorithms -> AES há uma entrada Animation para ver uma animação das rodadas, e

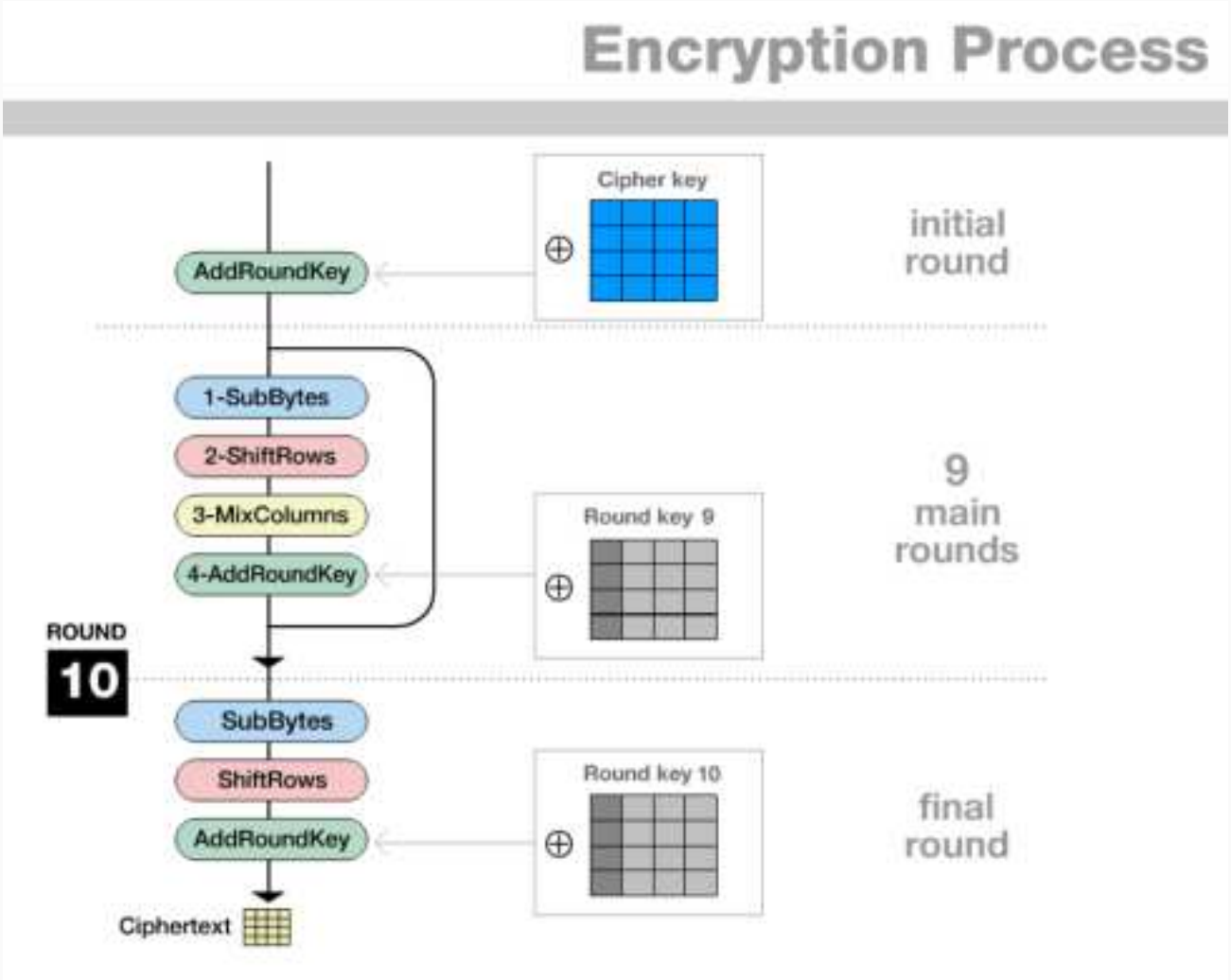


Figura 7: As rodadas do AES no CrypTool 1

Uma entrada `Inspector` para experimentar com os valores do texto claro e da chave:

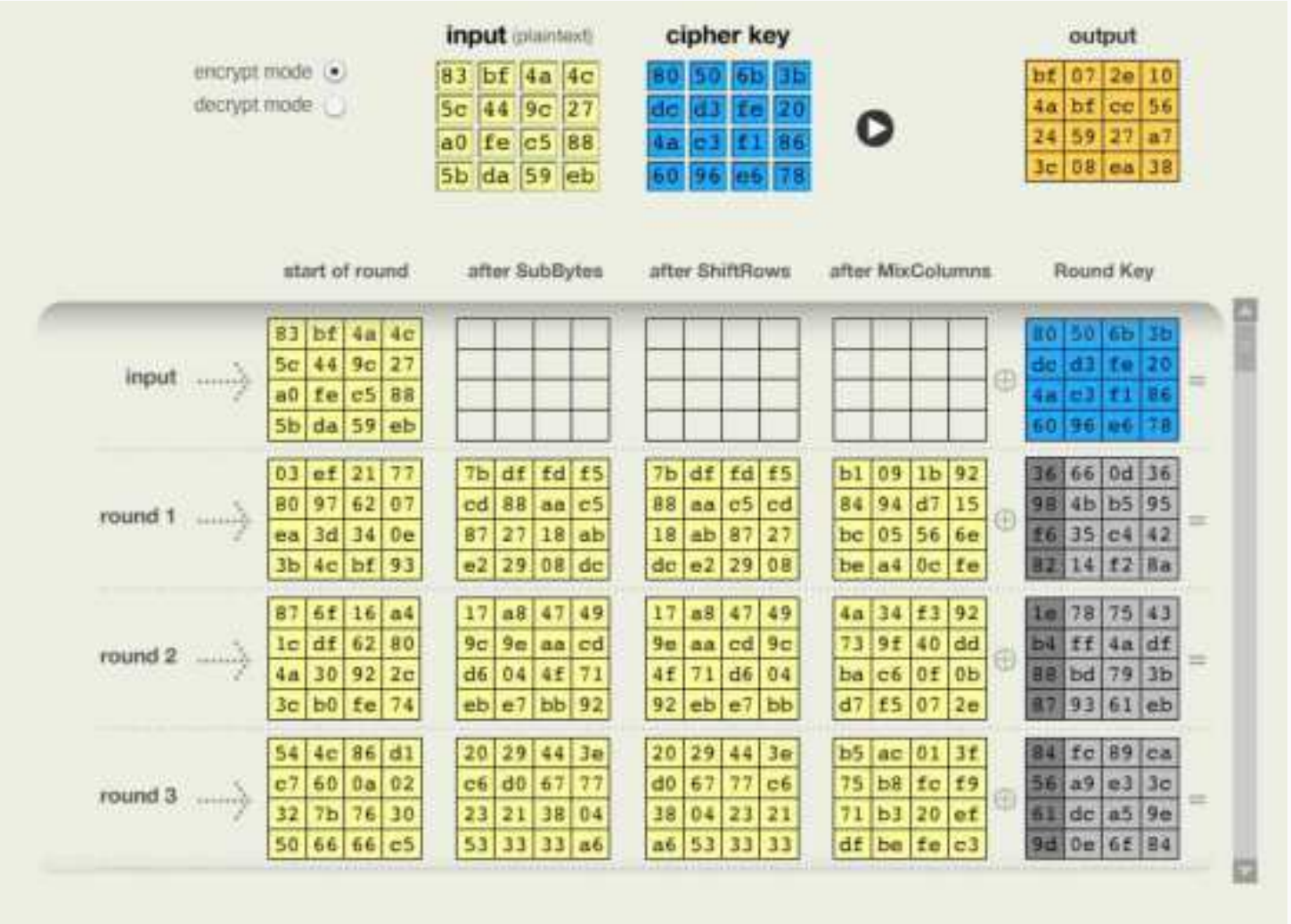


Figura 8: Os valores dos bloques ao longo das rodadas no `AES` no `CrypTool 1`

SubBytes

A transformação SubBytes substitui cada byte do bloco por outro byte pela tabela de substituição S-box dada abaixo.

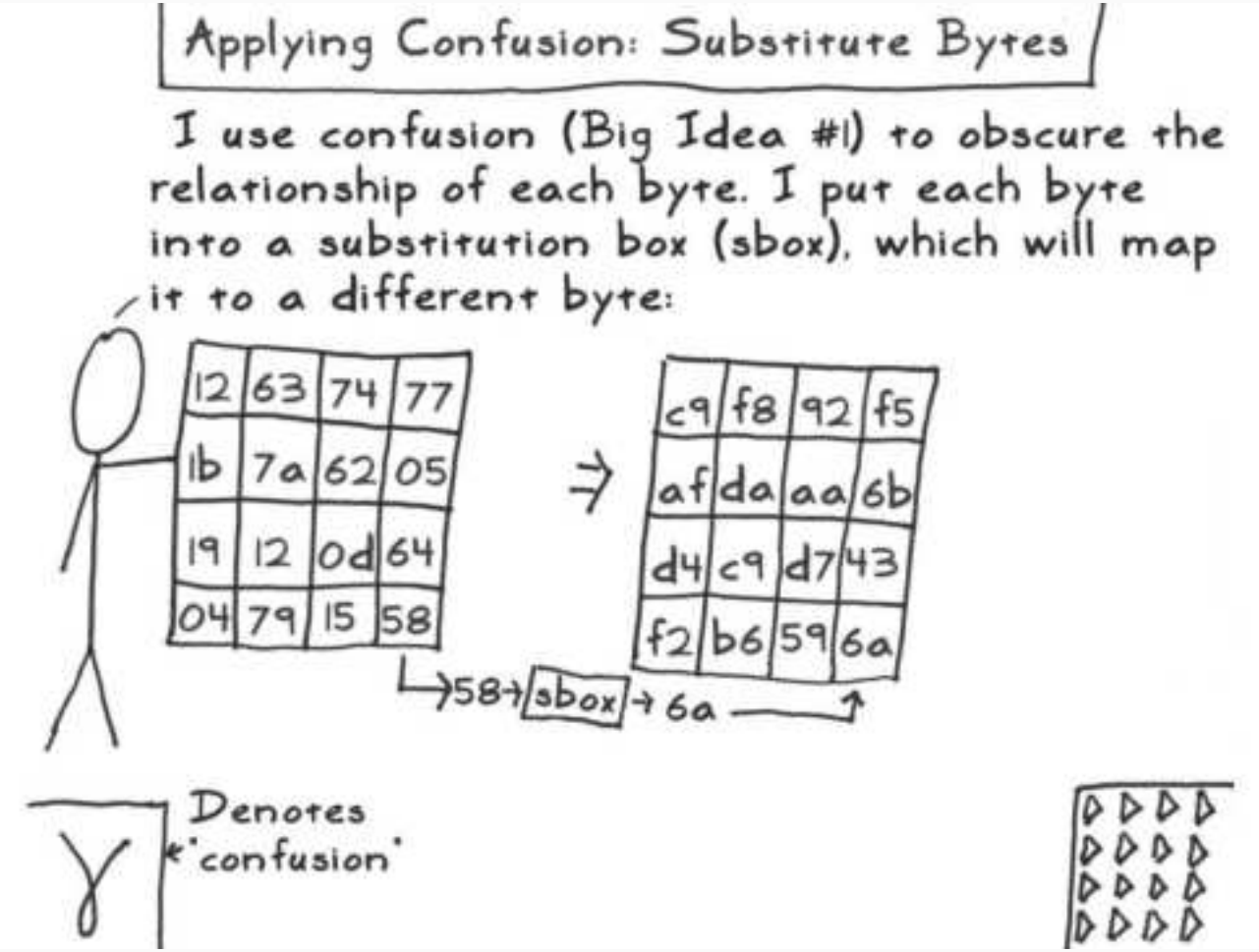


Figura 9: substituição no algoritmo AES

Em forma matricial,

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Como tabela calculada de antemão em notação hexadecimal (onde o número da linha corresponde ao primeiro e o número da coluna ao segundo digito do byte a ser substituído):

63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff
cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1

ShiftRows

A permutação ShiftRows rotaciona a linha $r = 0, 1, 2, 3$ do quadrado r posições para a esquerda. (Em particular, primeira linha *não* é rotacionada.) Visualmente,

B_{00}	B_{01}	B_{02}	B_{03}
B_{10}	B_{11}	B_{12}	B_{13}
B_{20}	B_{21}	B_{22}	B_{23}
B_{30}	B_{31}	B_{32}	B_{33}

é transformada em

B_{00}	B_{01}	B_{02}	B_{03}
B_{11}	B_{12}	B_{13}	B_{10}
B_{22}	B_{23}	B_{20}	B_{21}
B_{33}	B_{30}	B_{31}	B_{32}

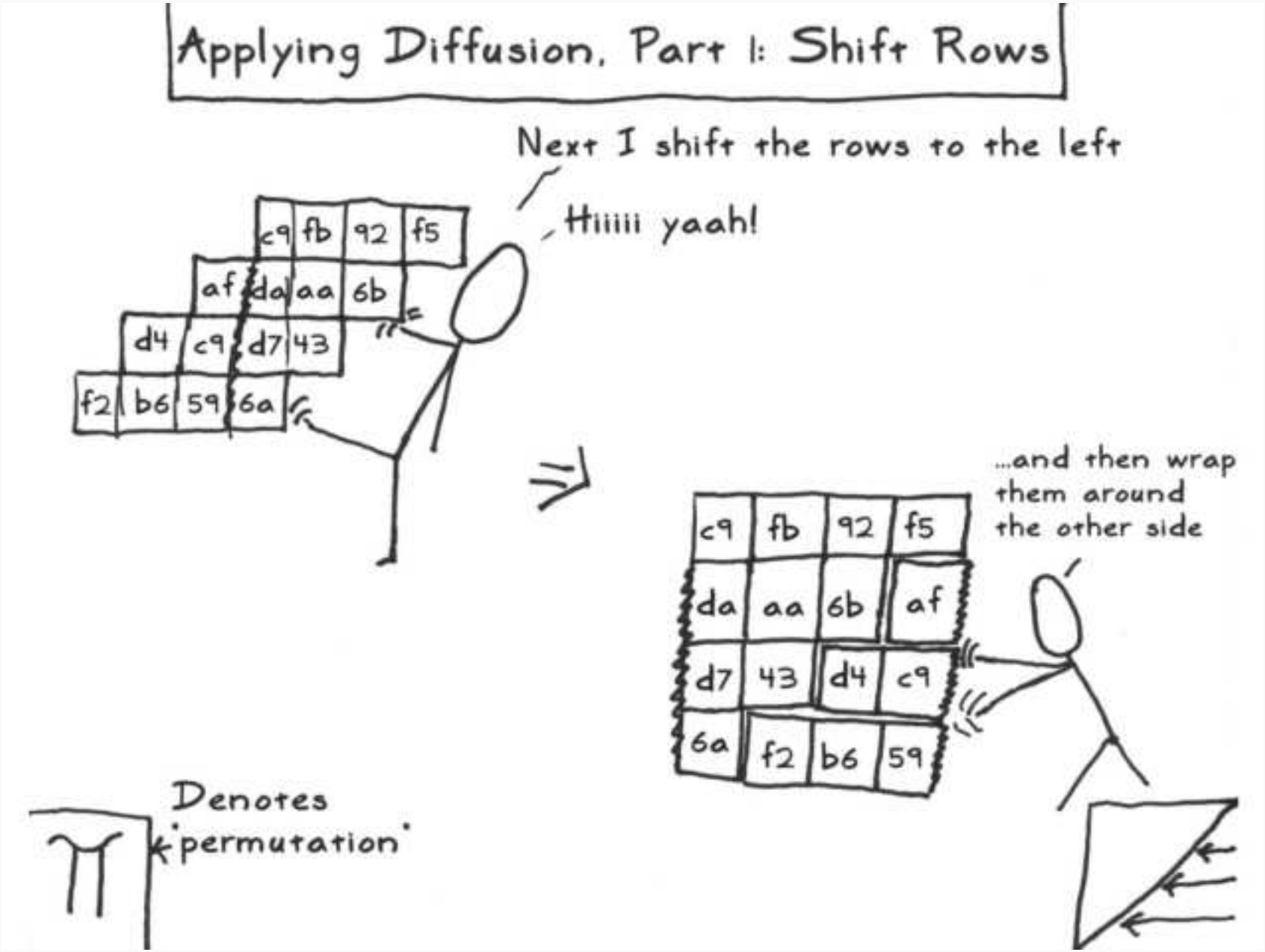


Figura 10: transposição no algoritmo AES

MixColumn

A transformação MixColumn multiplica cada coluna do bloco por uma matriz fixa. Mais exatamente,

- se B_j (com coeficientes $b_{0,j}$, $b_{1,j}$, $b_{2,j}$ e $b_{3,j}$) corresponde à coluna j do bloco de entrada, e
- se A_j (com coeficientes $a_{0,j}$, $a_{1,j}$, $a_{2,j}$ e $a_{3,j}$) corresponde à coluna j do bloco de saída da operação, temos

$$\begin{pmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{pmatrix}$$

Por exemplo, o byte $a_{0,j}$ é dado por

$$a_{0,j} = 2 \cdot b_{0,j} + 3 \cdot b_{1,j} + b_{2,j} + b_{3,j}$$

AddRoundKey

Esta transformação adiciona, pela operação `XOR`, a chave $\mathrm{W}(r)$ da rodada atual ao quadrado B do texto cifrado isto é,

$$\mathrm{B} \oplus \mathrm{W}(r).$$

A chave é gerada coluna a coluna. Denotemo-las por $\mathrm{W}(r)_0$, $\mathrm{W}(r)_1$, $\mathrm{W}(r)_2$ e $\mathrm{W}(r)_3$; isto é,

$$\mathrm{W}(r) = \mathrm{W}(r)_0 \mid \mathrm{W}(r)_1 \mid \mathrm{W}(r)_2 \mid \mathrm{W}(r)_3.$$

Como a chave tem 16 bytes, cada coluna tem 4 bytes.

1. A primeira chave $\mathrm{W}(0)$ é dada pela chave inicial W.

2. Para $r = 1, \ldots, \mathrm{R}$ (= 10 para nós), as quatro colunas $\mathrm{W}(r)_0$, $\mathrm{W}(r)_1$, $\mathrm{W}(r)_2$ e $\mathrm{W}(r)_3$ da nova chave são geradas a partir das colunas da antiga chave $\mathrm{W}(r-1)$ como segue:

   2.1 A primeira coluna $\mathrm{W}(r)_0$ é dada por

   $$\mathrm{W}(r)_0 = \mathrm{ScheduleCore}(\mathrm{W}(r-1)_3) \oplus \mathrm{W}(r-1)_0;$$

   isto é, a aplição de `ScheduleCore` à T = última coluna da chave da rodada precedente mais a primeira coluna da chave da rodada precedente, onde `ScheduleChore` aplica

   2.1.1 `SubWord`: Transforma cada um dos 4 bytes de T pela `S-box` dada em `SubBytes`.

   2.1.2 `RotWord`: Rotaciona T um byte à esquerda.

   2.1.3 Rcon(r): Adiciona (por `XOR`) a T o valor constante, em notação hexadecimal, $[(02)^{r-1}\ 00\ 00\ 00]$ (para a chave diferir).

   2.2 As colunas seguintes $\mathrm{W}(r)_1$, $\mathrm{W}(r)_2$ e $\mathrm{W}(r)_3$ são dadas, para $i = 1, 2$ e 3, por

   $$\mathrm{W}(r)_i = \mathrm{W}(r)_{i-1} \oplus \mathrm{W}(r-1)_i.$$

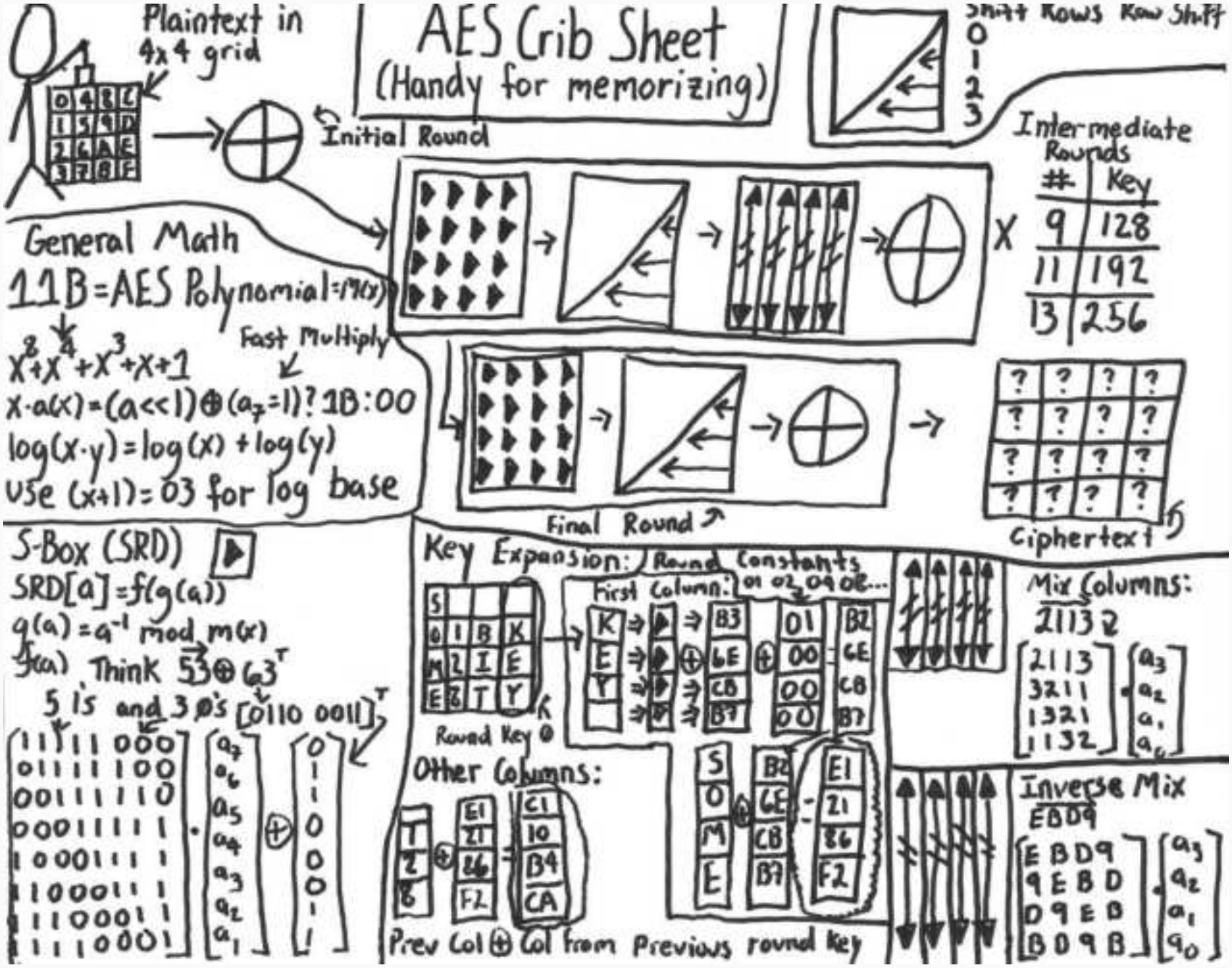


Figura 11: Algoritmo Completo

Observamos que a única transformação que *não é afim* (isto é, dada por um polinômio de grau 1, ou, equivalentemente, a composição de uma aplicação linear e um traslação) é a inversão no corpo $\mathbb{F}_{2^8}$ na operação `SubBytes`, com efeito

1. Na operação `SubBytes` são aplicadas, nesta ordem,
   1.1 a inversão em $\mathbb{F}_{2^8}$,
   1.2 uma aplicação linear, e
   1.3 a traslação por um vetor constante.
2. A operação `ShiftRows` é uma permutação, em particular,linear.
3. A operação `MixColumn` é linear.
4. A operação `AddRoundKey` é a traslação pela chave da rodada.

Quanto às metas da *difusão* e *confusão*, podemos ressaltar que a cada etapa são substituídos e transpostos cerca da metade dos bits (no `SubBytes`) ou bytes (no `MixColumn` e `ShiftRows`).

Anotações

Estão **disponíveis**

- os **eslaides** desta palestra e
- um **manuscrito** sobre criptografia que aprofunda o que aprendemos

online em `konfekt.bitbucket.io/talks/criptografia`

Referências

Heys, Howard M. 2002. «A tutorial on Linear and Differential Cryptanalysis». *Cryptologia* 26 (3). Taylor & Francis: 189–221.

1. Exemplos Históricos
2. Criptografia Simétrica
3. Codificação
4. Algoritmos Modernos
5. Cifra de Feistel Básica
6. AES