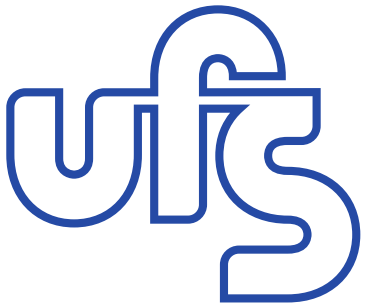
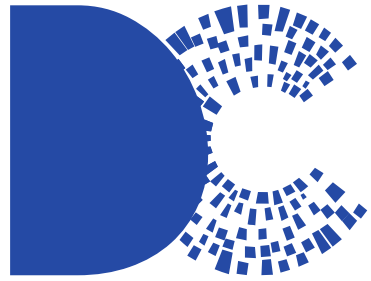




Criptografia simétrica

Projeto e Análise de Algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- Criptografia simétrica
  - Possui implementação eficiente das operações em hardware e software, por utilizar de iterações, substituições, permutações e operações binárias
  - Sua construção permite o uso repetitivo de chaves sem comprometer a segurança do sistema

Introdução

- Criptografia simétrica
  - Possui implementação eficiente das operações em hardware e software, por utilizar de iterações, substituições, permutações e operações binárias
  - Sua construção permite o uso repetitivo de chaves sem comprometer a segurança do sistema

Algoritmos: AES, DES, RC5, ...

Introdução

- Criptografia simétrica
  - O padrão NIST FIPS 197[1], criado em 2001 para o *Advanced Encryption Standard* (AES), utiliza o algoritmo Rijndael que foi desenvolvido pelos criptólogos belgas Joan Daemen e Vincent Rijmen

[1] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Introdução

- Criptografia simétrica
  - O padrão NIST FIPS 197[1], criado em 2001 para o *Advanced Encryption Standard* (AES), utiliza o algoritmo Rijndael que foi desenvolvido pelos criptólogos belgas Joan Daemen e Vincent Rijmen
  - Neste padrão de criptografia simétrica são utilizados blocos de dados com tamanho fixo de 128 bits e chaves privadas com 128, 192 ou 256 bits

[1] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Introdução

- Criptografia simétrica
  - O padrão NIST FIPS 197[1], criado em 2001 para o *Advanced Encryption Standard* (AES), utiliza o algoritmo Rijndael que foi desenvolvido pelos criptólogos belgas Joan Daemen e Vincent Rijmen
  - Neste padrão de criptografia simétrica são utilizados blocos de dados com tamanho fixo de 128 bits e chaves privadas com 128, 192 ou 256 bits

Nenhum ataque se mostrou viável ainda

[1] https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Introdução

- Definição do Campo de Galois de 8 bits $GF\left(2^8\right)$
  - Cada byte é representado por $b_7b_6b_5b_4b_3b_2b_1b_0$, sendo interpretado como um polinômio p

$$p\left(x\right) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

Introdução

- Definição do Campo de Galois de 8 bits $GF\left(2^8\right)$
  - Cada byte é representado por $b_7b_6b_5b_4b_3b_2b_1b_0$, sendo interpretado como um polinômio p

$$p\left(x\right) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

$$0x63 = 0b01100011 \longrightarrow x^6 + x^5 + x + 1$$

Introdução

- Adição em $GF\left(2^8\right)$
  - É realizada através do operador bit a bit de ou-exclusivo (xor), denotado pelo símbolo $\oplus$

Introdução

- Adição em $GF\left(2^8\right)$
  - É realizada através do operador bit a bit de ou-exclusivo (xor), denotado pelo símbolo $\oplus$
  - Considerando dois bytes $a = 0x57$ e $b = 0x83$

$$\begin{aligned} 0x57 \oplus 0x83 &= 0xD4 \\ 0b01010111 \oplus 0b10000011 &= 0b11010100 \\ \left(x^6 + x^4 + x^2 + x + 1\right) + \left(x^7 + x + 1\right) &= x^7 + x^6 + x^4 + x^2 \end{aligned}$$

Introdução

- Multiplicação em $GF\left(2^8\right)$
  - É implementada através de deslocamento de bits (função $xTimes$) e utilizando um polinômio fixo $m\left(x\right)$ para a redução modular, denotada pelo símbolo $\bullet$

$$xTimes\left(b\right) = \begin{cases} b_6b_5b_4b_3b_2b_1b_00 & se\ b_7 = 0 \\ b_6b_5b_4b_3b_2b_1b_00 \oplus m & se\ b_7 = 1 \end{cases}$$

Introdução

- Multiplicação em $GF\left(2^8\right)$
  - É implementada através de deslocamento de bits (função $xTimes$) e utilizando um polinômio fixo $m\left(x\right)$ para a redução modular, denotada pelo símbolo $\bullet$

$$xTimes\left(b\right) = \begin{cases} b_6b_5b_4b_3b_2b_1b_00 & se\ b_7 = 0 \\ b_6b_5b_4b_3b_2b_1b_00 \oplus m & se\ b_7 = 1 \end{cases}$$

  - Para $m\left(x\right) = x^8 + x^4 + x^3 + x + 1$

$$\begin{aligned} 0x57 \bullet 0x13 &= 0x57 \bullet (0x01 \oplus 0x02 \oplus 0x10) \\ &= 0x57 \oplus 0xAE \oplus 0x07 \\ &= 0xFE \end{aligned}$$

Introdução

- Multiplicação em $GF\left(2^8\right)$
  - É implementada através de deslocamento de bits (função $xTimes$) e utilizando um polinômio fixo $m\left(x\right)$ para a redução modular, denotada pelo símbolo $\bullet$

$$xTimes\left(b\right) = \begin{cases} b_6b_5b_4b_3b_2b_1b_00 & se\ b_7 = 0 \\ b_6b_5b_4b_3b_2b_1b_00 \oplus m & se\ b_7 = 1 \end{cases}$$

  - Para $m\left(x\right) = x^8 + x^4 + x^3 + x + 1$

$$\begin{aligned} 0x57 \bullet 0x13 &= 0x57 \bullet (0x01 \oplus 0x02 \oplus 0x10) \\ &= 0x57 \oplus 0xAE \oplus 0x07 \\ &= 0xFE \end{aligned}$$

  - Para um byte $b \neq 0x00$, existe um inverso multiplicativo b^{-1} tal que $b \bullet b^{-1} = 0x01$

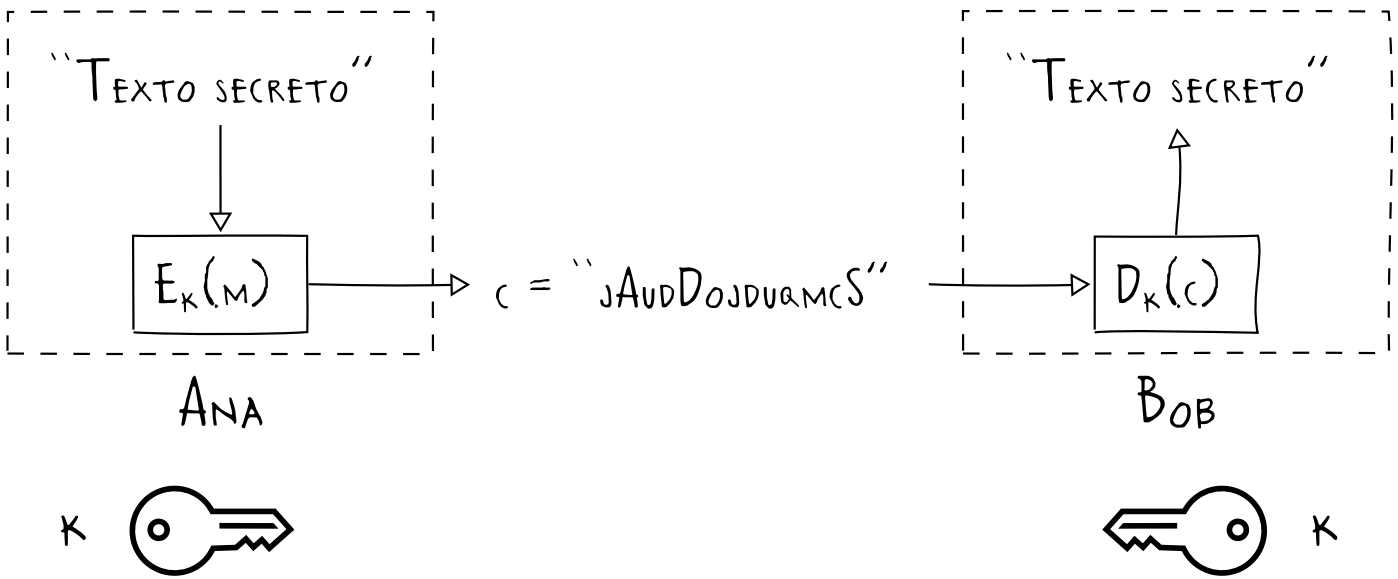
Introdução

- Multiplicação em $GF\left(2^8\right)$
  - $a\left(x\right) \bullet b\left(x\right) \bmod m\left(x\right)$, onde $m\left(x\right) = x^8 + x^4 + x^3 + x + 1$

```
1  // Função MultiplyGF
2  uint8_t MultiplyGF(uint8_t a, uint8_t b) {
3      // c(x) = 0, m(x) = x^4 + x^3 + x + 1
4      uint8_t c = 0, m = 0x1B;
5      // Enquanto b for maior que 0
6      while(b > 0) {
7          // b é impar (b[0] = 1) -> c(x) = c(x) + a(x)
8          c = c ^ ((b & 1) * a);
9          // Multiplica a(x) por 2
10         // Overflow (a[7] = 1) -> a(x) mod m(x)
11         a = (a << 1) ^ ((a >> 7) * m);
12         // Divide b por 2
13         b = b >> 1;
14     }
15     // c(x) = (a(x) + b(x)) mod m(x)
16     return c;
17 }
```

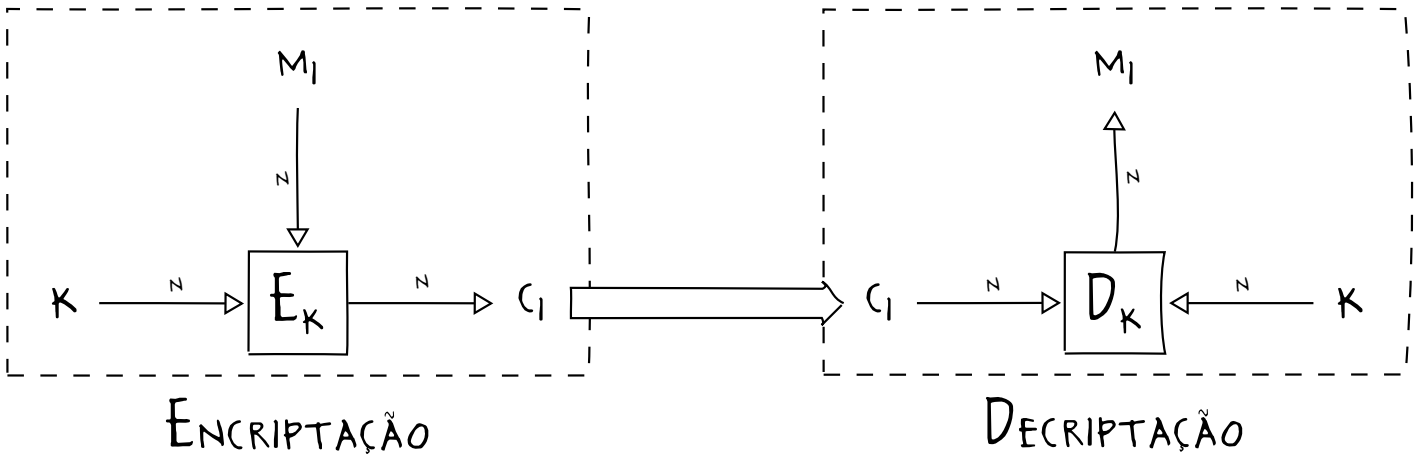
Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - É um esquema de criptografia que utiliza um conjunto de chaves privadas $k \in K$, tal que $E_k(m) = c$ e $D_k(c) = m$, que precisam ser previamente compartilhadas pelas partes envolvidas na comunicação



Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - Os dados são encriptados ou decriptados utilizando blocos de tamanho fixo com *n* bits (128, 192 ou 256)



Electronic Codebook (ECB)

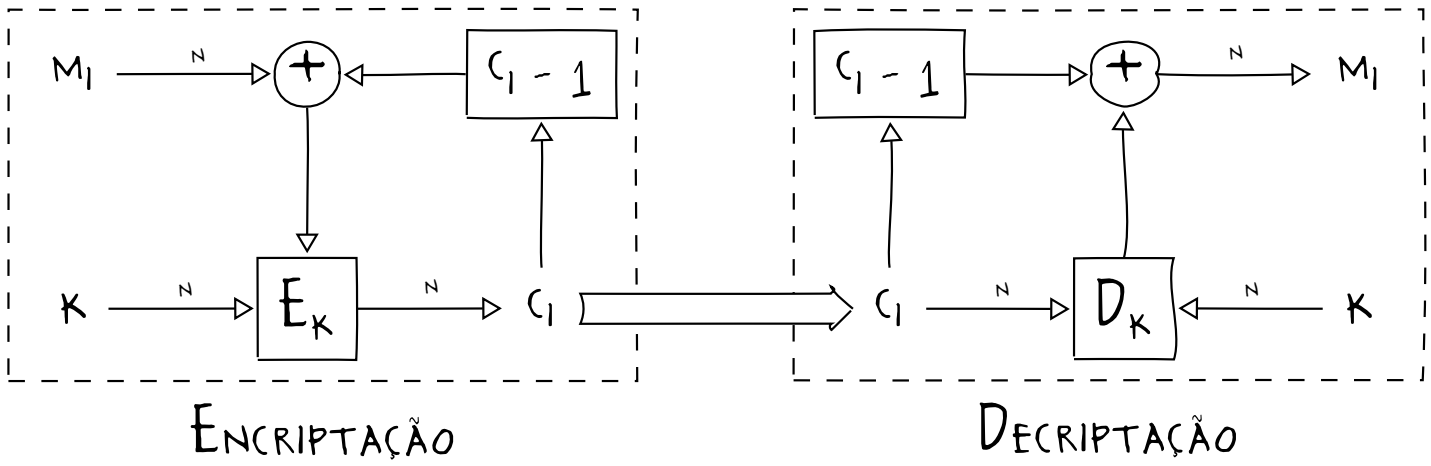
Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de decriptação (AES-ECB)
2  void aes_d_ecb(uint8_t* m, const uint8_t* c, aes_t*
       aes) {
3      // Nr = Número de rodadas
4      const uint8_t Nr = aes->Nk + 6;
5      // Decriptação AES-EBC
6      Decipher(m, c, aes->ke, Nr);
7  }
8  // Procedimento de encriptação (AES-ECB)
9  void aes_e_ecb(uint8_t* c, const uint8_t* m, aes_t*
       aes) {
10     // Nr = Número de rodadas
11     const uint8_t Nr = aes->Nk + 6;
12     // Encriptação AES-EBC
13     Cipher(c, m, aes->ke, Nr);
14 }
```

Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - Os dados são encriptados ou decriptados utilizando blocos de tamanho fixo com *n* bits (128, 192 ou 256)



Cipher-Block Chaining (CBC)

Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de decriptação (AES-CBC)
2  void aes_d_cbc(uint8_t* m, const uint8_t* c, size_t l,
       aes_t* aes) {
3      // Nr = Número de rodadas
4      const uint8_t Nr = aes->Nk + 6;
5      // Ponteiro para valor anterior
6      const uint8_t* ci1 = aes->c0;
7      // Decriptação AES-CBC
8      for(size_t i = 0; i < l; i = i + 16) {
9          Decipher(m + i, c + i, aes->ke, Nr);
10         Xor(m + i, m + i, ci1);
11         ci1 = c + i;
12     }
13     // Salvando c[i - 1]
14     memcpy(aes->c0, ci1, 16);
15 }
```

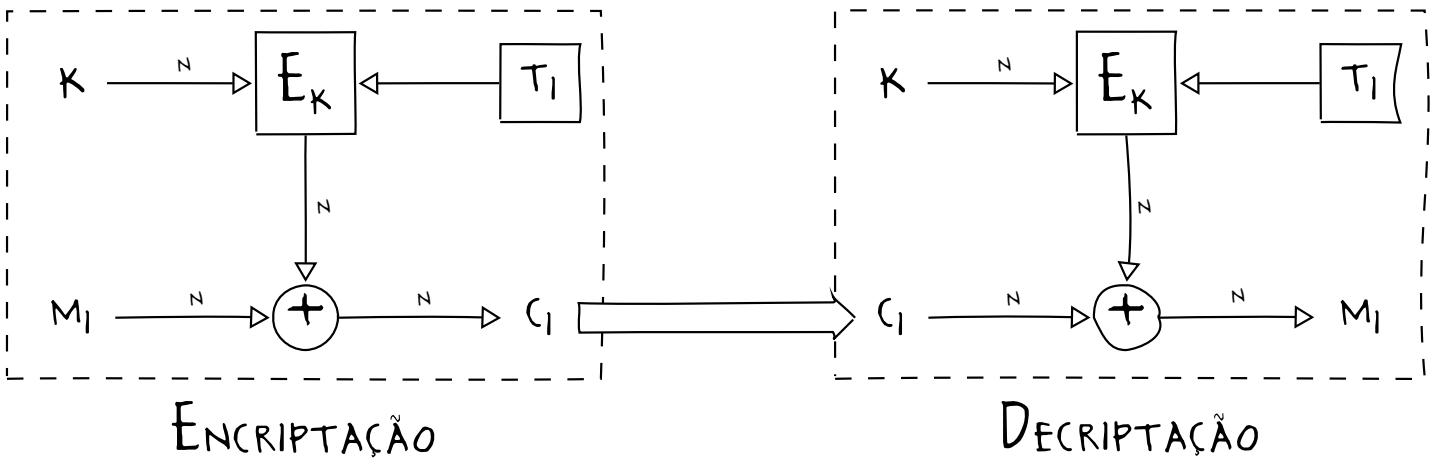
Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de encriptação (AES-CBC)
2  void aes_e_cbc(uint8_t* c, const uint8_t* m, size_t l,
       aes_t* aes) {
3      // Nr = Número de rodadas
4      const uint8_t Nr = aes->Nk + 6;
5      // Armazenamento de m[i] xor c[i - 1]
6      static uint8_t* mxci1 = (uint8_t*)(malloc(16));
7      // Ponteiro para valor anterior
8      uint8_t* ci1 = aes->c0;
9      // Encriptação AES-CBC
10     for(size_t i = 0; i < l; i = i + 16) {
11         Xor(mxci1, m + i, ci1);
12         Cipher(c + i, mxci1, aes->ke, Nr);
13         ci1 = c + i;
14     }
15     // Salvando c[i - 1]
16     memcpy(aes->c0, ci1, 16);
17 }
```

Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - Os dados são encriptados ou decriptados utilizando blocos de tamanho fixo com *n* bits (128, 192 ou 256)



Counter Mode (CTR)

Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de decriptação/encriptação (AES-CTR)
2  void aes_x_ctr(uint8_t* out, uint8_t* in, size_t l,
       aes_t* aes) {
3      // Nr = Número de rodadas
4      const uint8_t Nr = aes->Nk + 6;
5      // Ponteiro para contador
6      uint8_t* ti = aes->c0;
7      // Encriptação AES-CTR
8      for(size_t i = 0; i < l; i = i + 16) {
9          Cipher(out + i, ti, aes->ke, Nr);
10         Xor(out + i, out + i, in + i);
11         AddCounter(ti);
12     }
13 }
```

Criptografia simétrica

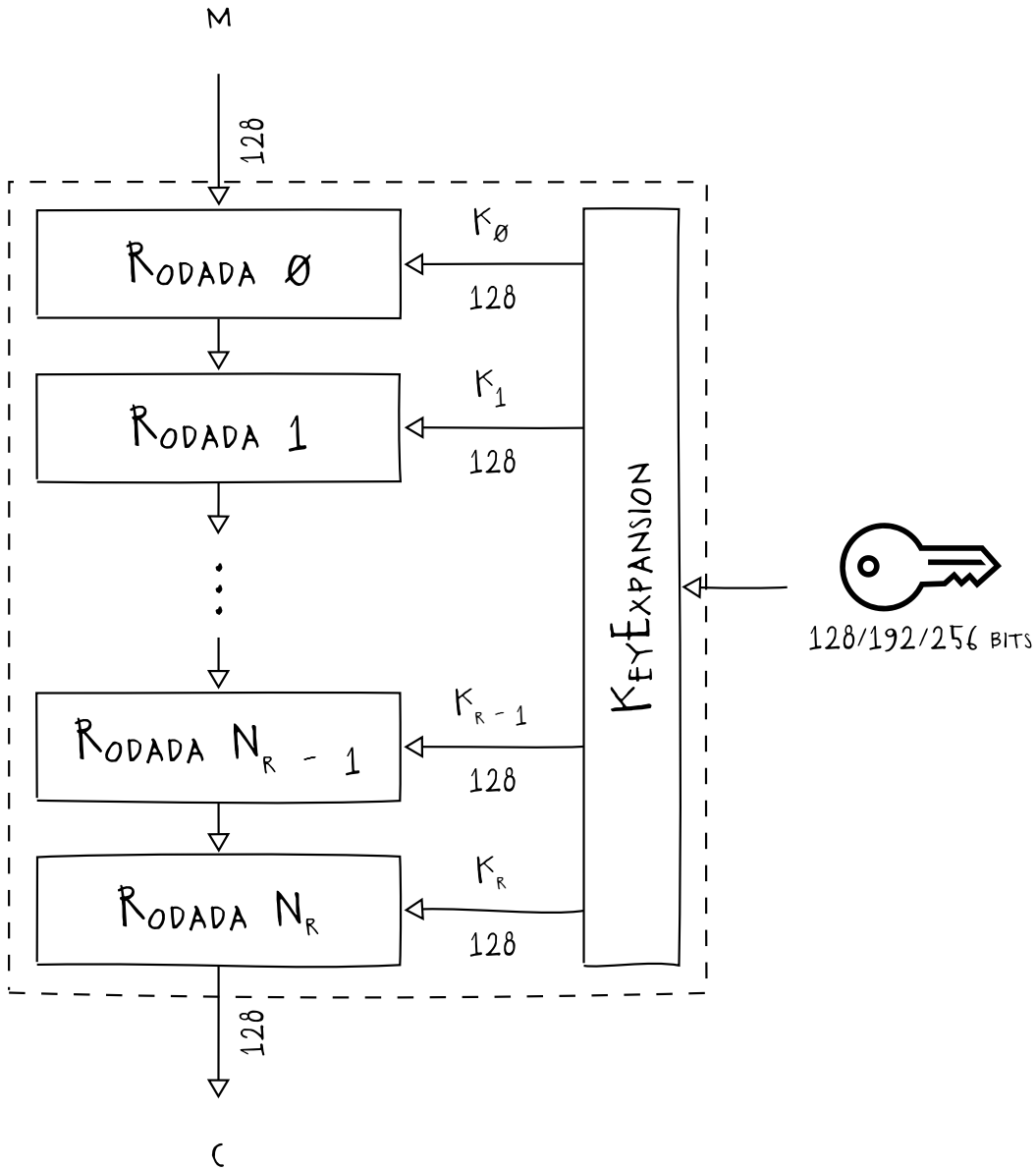
- *Advanced Encryption Standard* (AES)

	Tamanho da chave N_K	Tamanho do bloco N_B	Número de rodadas N_R
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Palavras de 32 bits

Criptografia simétrica

- *Advanced Encryption Standard* (AES)



Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Estrutura do AES
2  typedef struct aes_t {
3      uint8_t* c0;
4      uint8_t* k;
5      uint8_t* ke;
6      size_t Nk;
7  } aes_t;
8  // Round Constant (1 <= i <= 10 -> i^(i - 1))
9  const uint8_t rcon[11] = { 0x00, 0x01, 0x02, 0x04,
       0x08, 0x10, 0x20, 0x40, 0x80, 0x1B, 0x36 };
```

Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - Tabela de valores de substituição (*S – BOX*)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x63	0x7C	0x77	0x7B	0xF2	0x6B	0x6F	0xC5	0x30	0x01	0x67	0x2B	0xFE	0xD7	0xAB	0x76
1	0xCA	0x82	0xC9	0x7D	0xFA	0x59	0x47	0xF0	0xAD	0xD4	0xA2	0xAF	0x9C	0xA4	0x72	0xC0
2	0xB7	0xFD	0x93	0x26	0x36	0x3F	0xF7	0xCC	0x34	0xA5	0xE5	0xF1	0x71	0xD8	0x31	0x15
3	0x04	0xC7	0x23	0xC3	0x18	0x96	0x05	0x9A	0x07	0x12	0x80	0xE2	0xEB	0x27	0xB2	0x75
4	0x09	0x83	0x2C	0x1A	0x1B	0x6E	0x5A	0xA0	0x52	0x3B	0xD6	0xB3	0x29	0xE3	0x2F	0x84
5	0x53	0xD1	0x00	0xED	0x20	0xFC	0xB1	0x5B	0x6A	0xCB	0xBE	0x39	0x4A	0x4C	0x58	0xCF
6	0xD0	0xEF	0xAA	0xFB	0x43	0x4D	0x33	0x85	0x45	0xF9	0x02	0x7F	0x50	0x3C	0x9F	0xA8
7	0x51	0xA3	0x40	0x8F	0x92	0x9D	0x38	0xF5	0xBC	0xB6	0xDA	0x21	0x10	0xFF	0xF3	0xD2
8	0xCD	0x0C	0x13	0xEC	0x5F	0x97	0x44	0x17	0xC4	0xA7	0x7E	0x3D	0x64	0x5D	0x19	0x73
9	0x60	0x81	0x4F	0xDC	0x22	0x2A	0x90	0x88	0x46	0xEE	0xB8	0x14	0xDE	0x5E	0x0B	0xDB
A	0xE0	0x32	0x3A	0x0A	0x49	0x06	0x24	0x5C	0xC2	0xD3	0xAC	0x62	0x91	0x95	0xE4	0x79
B	0xE7	0xC8	0x37	0x6D	0x8D	0xD5	0x4E	0xA9	0x6C	0x56	0xF4	0xEA	0x65	0x7A	0xAE	0x08
C	0xBA	0x78	0x25	0x2E	0x1C	0xA6	0xB4	0xC6	0xE8	0xDD	0x74	0x1F	0x4B	0xBD	0x8B	0x8A
D	0x70	0x3E	0xB5	0x66	0x48	0x03	0xF6	0x0E	0x61	0x35	0x57	0xB9	0x86	0xC1	0x1D	0x9E
E	0xE1	0xF8	0x98	0x11	0x69	0xD9	0x8E	0x94	0x9B	0x1E	0x87	0xE9	0xCE	0x55	0x28	0xDF
F	0x8C	0xA1	0x89	0x0D	0xBF	0xE6	0x42	0x68	0x41	0x99	0x2D	0x0F	0xB0	0x54	0xBB	0x16

Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - Tabela de valores de substituição ($S-BOX^{-1}$)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x52	0x09	0x6A	0xD5	0x30	0x36	0xA5	0x38	0xBF	0x40	0xA3	0x9E	0x81	0xF3	0xD7	0xFB
1	0x7C	0xE3	0x39	0x82	0x9B	0x2F	0xFF	0x87	0x34	0x8E	0x43	0x44	0xC4	0xDE	0xE9	0xCB
2	0x54	0x7B	0x94	0x32	0xA6	0xC2	0x23	0x3D	0xEE	0x4C	0x95	0x0B	0x42	0xFA	0xC3	0x4E
3	0x08	0x2E	0xA1	0x66	0x28	0xD9	0x24	0xB2	0x76	0x5B	0xA2	0x49	0x6D	0x8B	0xD1	0x25
4	0x72	0xF8	0xF6	0x64	0x86	0x68	0x98	0x16	0xD4	0xA4	0x5C	0xCC	0x5D	0x65	0xB6	0x92
5	0x6C	0x70	0x48	0x50	0xFD	0xED	0xB9	0xDA	0x5E	0x15	0x46	0x57	0xA7	0x8D	0x9D	0x84
6	0x90	0xD8	0xAB	0x00	0x8C	0xBC	0xD3	0x0A	0xF7	0xE4	0x58	0x05	0xB8	0xB3	0x45	0x06
7	0xD0	0x2C	0x1E	0x8F	0xCA	0x3F	0x0F	0x02	0xC1	0xAF	0xBD	0x03	0x01	0x13	0x8A	0x6B
8	0x3A	0x91	0x11	0x41	0x4F	0x67	0xDC	0xEA	0x97	0xF2	0xCF	0xCE	0xF0	0xB4	0xE6	0x73
9	0x96	0xAC	0x74	0x22	0xE7	0xAD	0x35	0x85	0xE2	0xF9	0x37	0xE8	0x1C	0x75	0xDF	0x6E
A	0x47	0xF1	0x1A	0x71	0x1D	0x29	0xC5	0x89	0x6F	0xB7	0x62	0x0E	0xAA	0x18	0xBE	0x1B
B	0xFC	0x56	0x3E	0x4B	0xC6	0xD2	0x79	0x20	0x9A	0xDB	0xC0	0xFE	0x78	0xCD	0x5A	0xF4
C	0x1F	0xDD	0xA8	0x33	0x88	0x07	0xC7	0x31	0xB1	0x12	0x10	0x59	0x27	0x80	0xEC	0x5F
D	0x60	0x51	0x7F	0xA9	0x19	0xB5	0x4A	0x0D	0x2D	0xE5	0x7A	0x9F	0x93	0xC9	0x9C	0xEF
E	0xA0	0xE0	0x3B	0x4D	0xAE	0x2A	0xF5	0xB0	0xC8	0xEB	0xBB	0x3C	0x83	0x53	0x99	0x61
F	0x17	0x2B	0x04	0x7E	0xBA	0x77	0xD6	0x26	0xE1	0x69	0x14	0x63	0x55	0x21	0x0C	0x7D

Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento RotWord
2  void RotWord(uint8_t* data) {
3      // [d1, d2, d3, d0] <- [d0, d1, d2, d3]
4      uint8_t rotation[4] = { data[1], data[2], data[3],
           data[0] };
5      memcpy(data, rotation, 4);
6  }
7  // Procedimento SubWord
8  void SubWord(uint8_t* data) {
9      // [sbox[d0], sbox[d1], sbox[d2], sbox[d3]] <- [d0,
           d1, d2, d3]
10     data[0] = sbox[data[0]];
11     data[1] = sbox[data[1]];
12     data[2] = sbox[data[2]];
13     data[3] = sbox[data[3]];
14 }
```

Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento KeyExpansion
2  void KeyExpansion(uint8_t* out, uint8_t* in, uint8_t
       Nk) {
3      const uint8_t Nr = Nk + 6; uint8_t temp[4];
4      // Primeira rodada é a própria chave
5      for(uint8_t i = 0; i < Nk; i++)
6          WriteWord(out, i << 2, in, i << 2);
7      // Gerando as rodadas a partir das anteriores
8      for(uint8_t i = Nk; i < (Nr + 1) << 2; i++) {
9          WriteWord(temp, 0, out, (i - 1) << 2);
10         if(i % Nk == 0) {
11             RotWord(temp); SubWord(temp);
12             temp[0] = temp[0] ^ rcon[i / Nk];
13         }
14         else if(Nk > 6 && i % Nk == 4) SubWord(temp);
15         WriteWordXor(out, i << 2, out, (i - Nk) << 2,
               temp);
16     }
17 }
```

Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de encriptação
2  void Cipher(uint8_t* c, const uint8_t* m, uint8_t* k,
       uint8_t Nr) {
3      uint8_t state[4][4];
4      ReadInput(state, m);
5      AddRoundKey(state, k, 0);
6      for(uint8_t i = 1; i < Nr; i++) {
7          SubBytes(state);
8          ShiftRows(state);
9          MixColumns(state);
10         AddRoundKey(state, k, i);
11     }
12     SubBytes(state);
13     ShiftRows(state);
14     AddRoundKey(state, k, Nr);
15     WriteOutput(c, state);
16 }
```

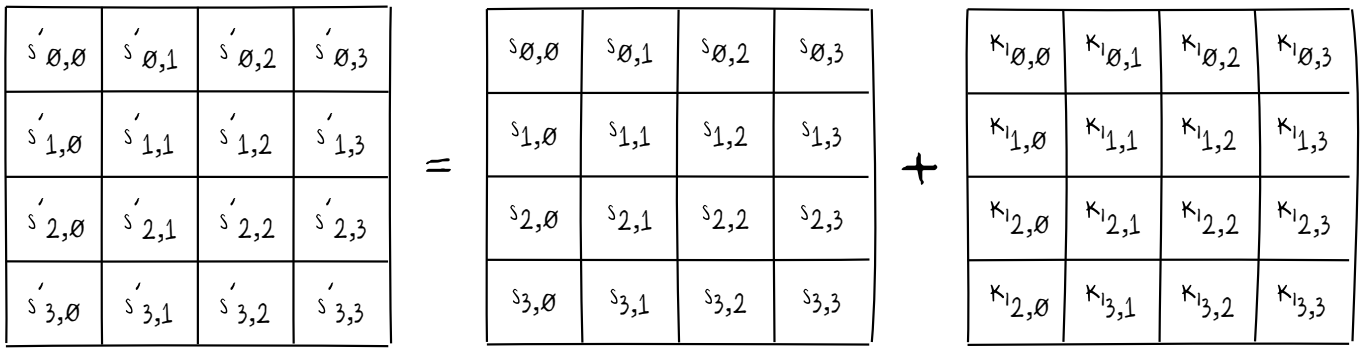
Criptografia simétrica

- *Advanced Encryption Standard* (AES)

```
1  // Procedimento de decriptação
2  void Decipher(uint8_t* m, const uint8_t* c, uint8_t* k,
       uint8_t Nr) {
3      uint8_t state[4][4];
4      ReadInput(state, c);
5      AddRoundKey(state, k, Nr);
6      for(int8_t i = Nr - 1; i >= 1; i--) {
7          InvShiftRows(state);
8          InvSubBytes(state);
9          AddRoundKey(state, k, i);
10         InvMixColumns(state);
11     }
12     InvShiftRows(state);
13     InvSubBytes(state);
14     AddRoundKey(state, k, 0);
15     WriteOutput(m, state);
16 }
```

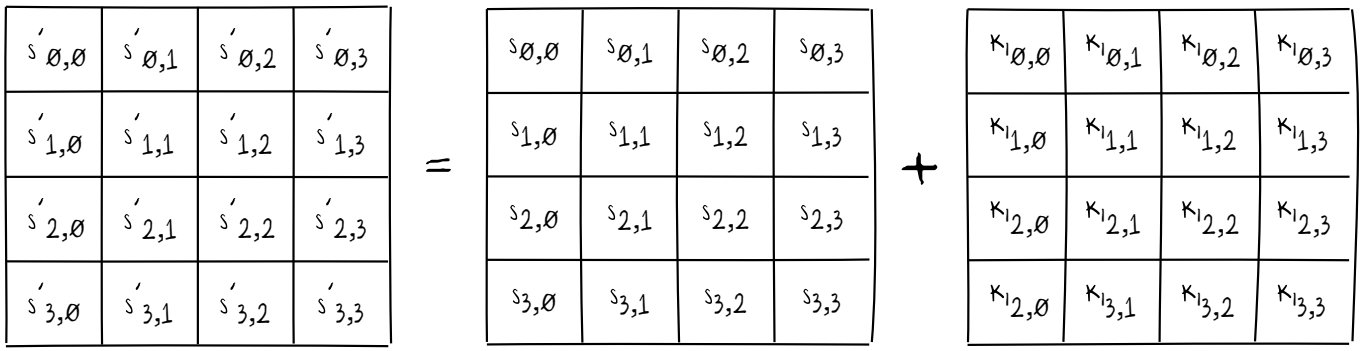
Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - AddRoundKey



Criptografia simétrica

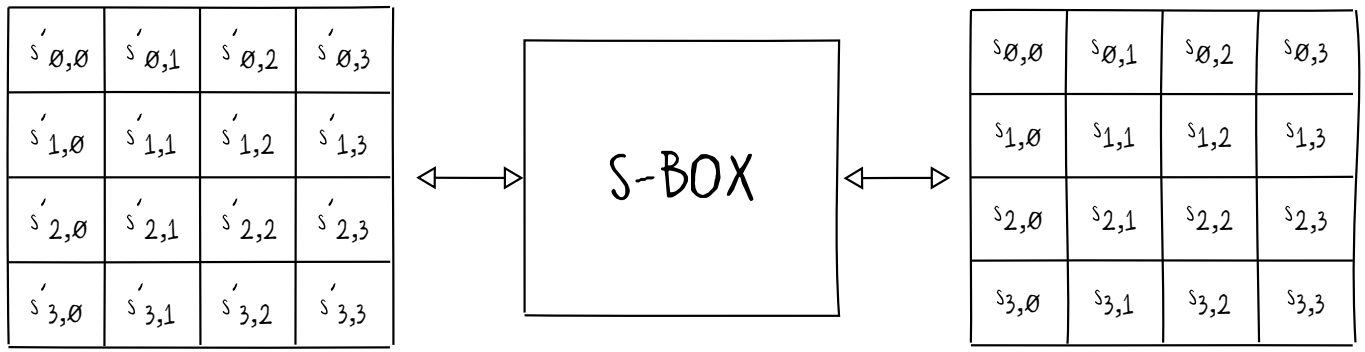
- *Advanced Encryption Standard* (AES)
  - AddRoundKey



NA ARITMÉTICA $GF(2^8)$, A ADIÇÃO É EQUIVALENTE AO OU-EXCLUSIVO BIT A BIT

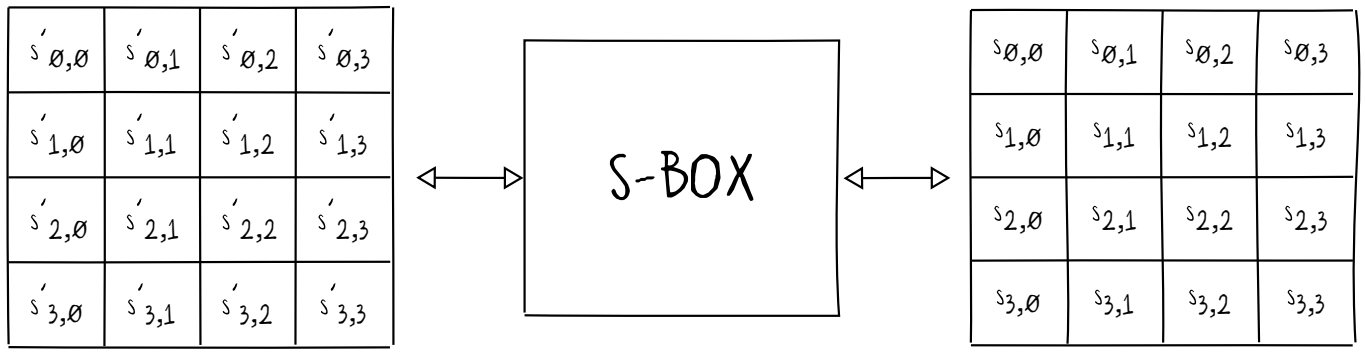
Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - SubBytes/InvSubBytes



Criptografia simétrica

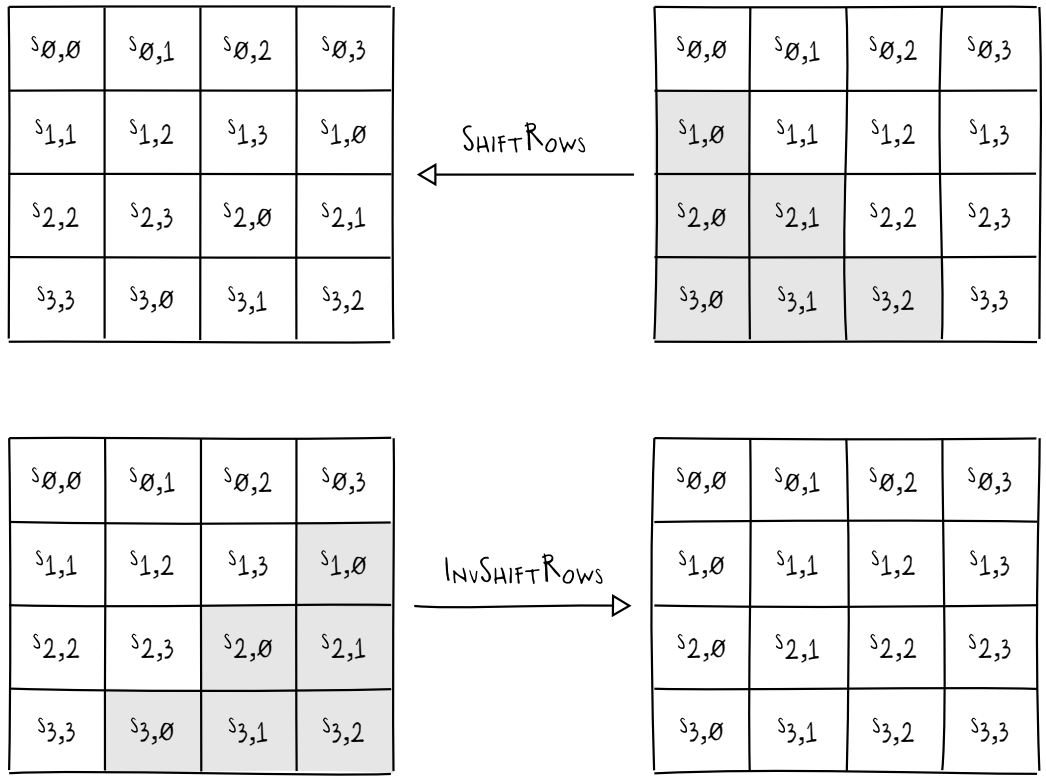
- *Advanced Encryption Standard* (AES)
  - SubBytes/InvSubBytes



$$s'[i,j] = \text{s-box}[s[i,j]]$$
$$s[i,j] = \text{s-box}^{-1}[s'[i,j]]$$

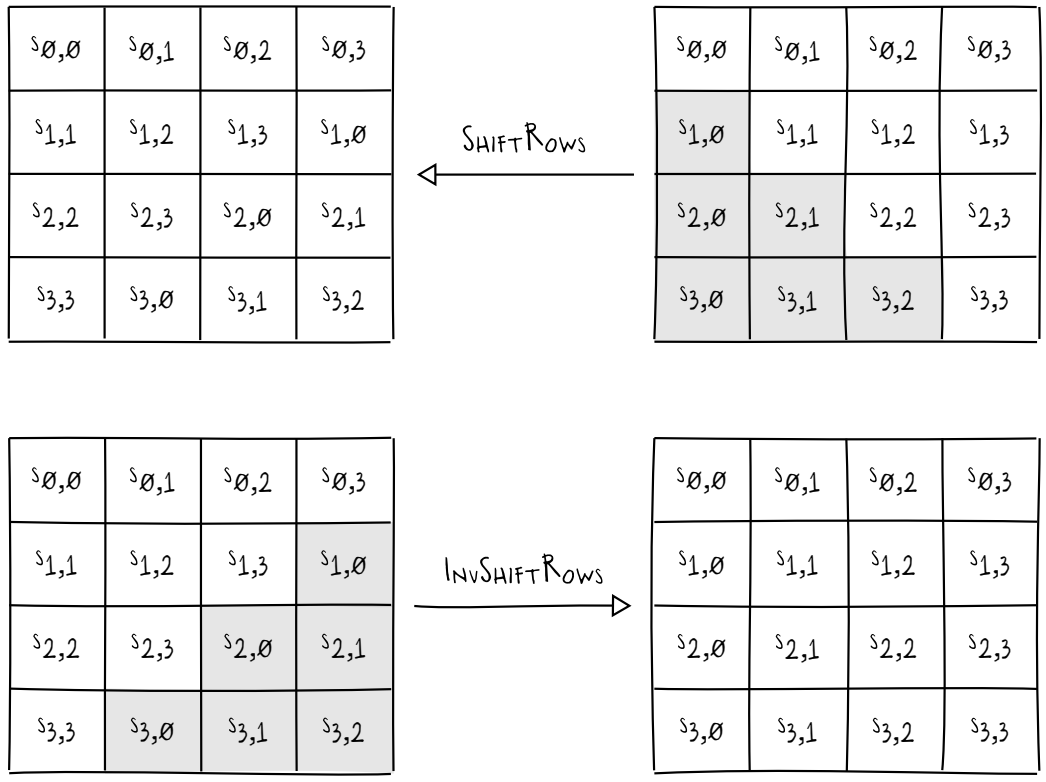
Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - ShiftRows/InvShiftRows



Criptografia simétrica

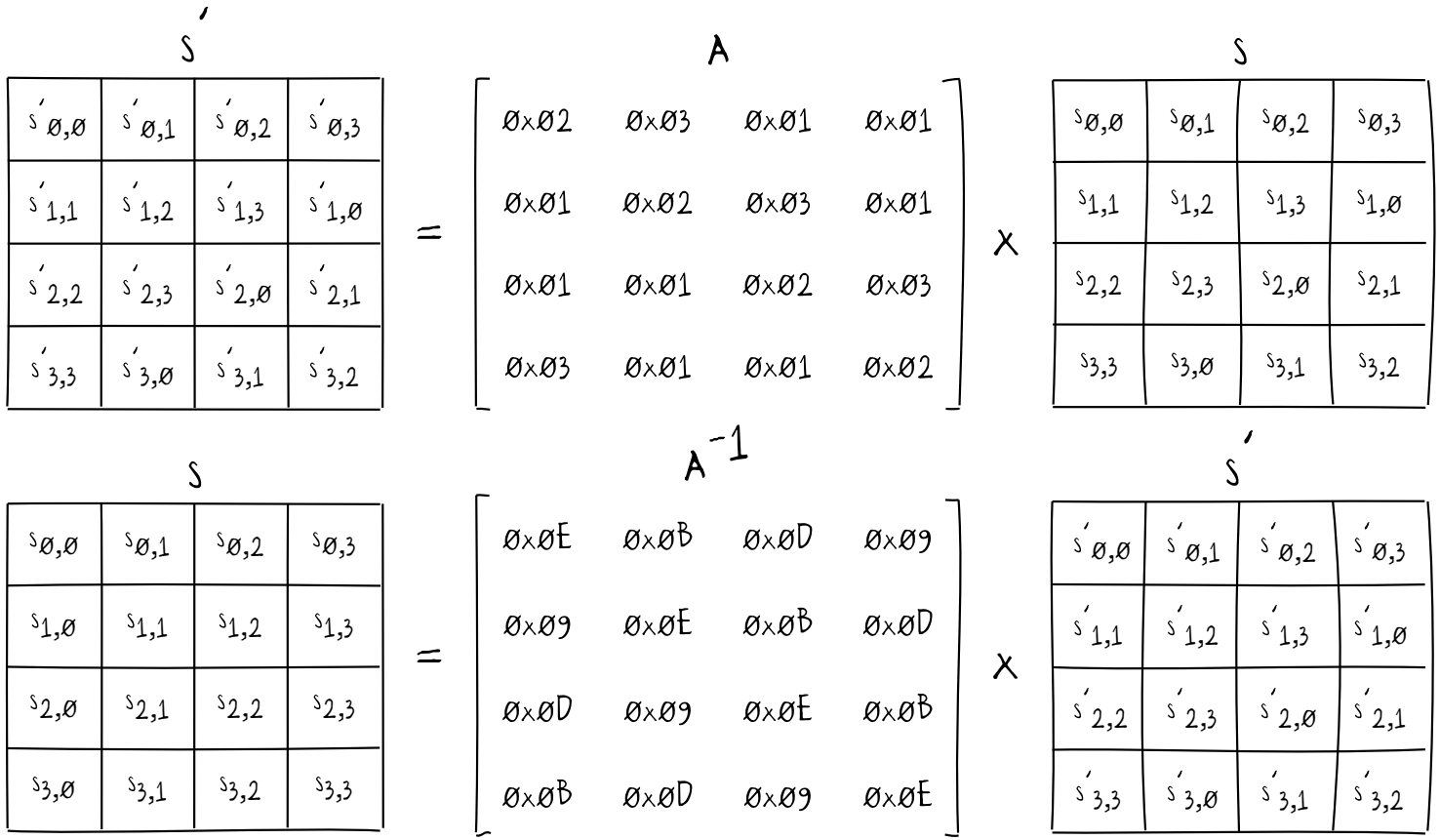
- *Advanced Encryption Standard* (AES)
  - ShiftRows/InvShiftRows



$$s = \text{InvShiftRows}(\text{ShiftRows}(s))$$

Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - MixColumns/InvMixColumns



Criptografia simétrica

- *Advanced Encryption Standard* (AES)
  - MixColumns/InvMixColumns

$$
\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,1} & s'_{1,2} & s'_{1,3} & s'_{1,0} \\ s'_{2,2} & s'_{2,3} & s'_{2,0} & s'_{2,1} \\ s'_{3,3} & s'_{3,0} & s'_{3,1} & s'_{3,2} \end{bmatrix} = \begin{bmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{bmatrix} \times \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{bmatrix}
$$

$$
\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} 0x0E & 0x0B & 0x0D & 0x09 \\ 0x09 & 0x0E & 0x0B & 0x0D \\ 0x0D & 0x09 & 0x0E & 0x0B \\ 0x0B & 0x0D & 0x09 & 0x0E \end{bmatrix} \times \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,1} & s'_{1,2} & s'_{1,3} & s'_{1,0} \\ s'_{2,2} & s'_{2,3} & s'_{2,0} & s'_{2,1} \\ s'_{3,3} & s'_{3,0} & s'_{3,1} & s'_{3,2} \end{bmatrix}
$$

As multiplicações são realizadas em $GF(2^8)$

Exercício

- A empresa de tecnologia Poxim Tech está aplicando técnicas de criptografia em todos os seus sistemas de transmissão de dados, visando para proteger os dados de acessos não autorizados
  - Os dados transmitidos entre as partes são representados por bytes no formato hexadecimal, com alinhamento de 128 bits (completando com zeros) para as mensagens utilizadas pelo AES
  - A criptografia simétrica AES é aplicada no modo ECB com chaves de 128, 192 ou 256 bits (tamanho definido a partir das chaves privadas a ou b)
  - No compartilhamento das chaves privadas é utilizado o Diffie-Hellman com parâmetros de até 2.048 bits

Exercício

- Formato do arquivo de entrada
  - *Número de operações* (n)
    - *dh a b g p*
    - *d c*
    - *e m*

```
1 3
2 dh␣2F333D84630F102FDA0B594D4FF7CA46␣FBDB83740FB1D83EE415
      C34725D377FF␣C54B073C6A2B3745AEAC545F8493439A568BBF29
      02BE07D20A359A20A9BBD26E06DAAA7005E2B5B48E0913129C57A
      CF2E26B1BE42923B63358505401 0B266F11␣1219C943937D661A8
      CA99AA1DC0CCBC2D28018D60CAB90A8D9097BC5981C99AA3662EE
      C9DF54E36CFD7D0DD98AD99B5C59B332655FC20E38CB89FE63A59
      70EDB
3 d␣F0FA40FAF0F0CA
4 e␣00112233445566778899AABBCCDDEEFF50C0440
```

Exercício

- Formato do arquivo de saída
  - Para cada comando é exibido seu resultado

```
1 s=E8613C49876806B074535ACF62DD673D
2 m=C6085D6C870841046C213D192E8979F3
3 c=4B0200E09FF592BA32668E18CAA7FDF8A4CDB7C7C48EF142575F4A
      5836AFBB4D
```