Немного про блочные шифры

ОПР(Блочный шифр)

Блочный шифр это криптосистема $(\{0,1\}^n, \{0,1\}^k, \{0,1\}^n, E, D)$ где:

- $\mathcal{M} = \{0,1\}^n$
- $\mathcal{K} = \{0,1\}^k$
- $\mathcal{C} = \{0,1\}^n$
- n - длина блока
- k - длина ключа

Идея применять к маленьким кусочкам открытого текста сложные функции(которые нужно задать таблицей)

Затем перемешаем эти блоки(например с помощью линейного преобразования либо другая простая функция). Действуем этой функцией на весь большой блок открытого текста

Итеративная схема блочного шифра

Есть

- $f : \{0,1\}^n \rightarrowtail\!\!\!\twoheadrightarrow \{0,1\}^n$ - сложное, локальное преобразование
- $g : \{0,1\}^n \rightarrowtail\!\!\!\twoheadrightarrow \{0,1\}^n$ - простое, глобальное преобразование
- $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ берёт че-то и ключ и возращает че-то другое
- $h_k = h(\_, k)$ **т.е в преобразование закладываем ключ k**

По итогу получаем формулу для криптограммы

$$c = (h_k \circ f \circ g)^r(m)$$

- r - это число раундов

Конструкция фейстеля

Открытый текст разобъем его на 2 части(n - длина открытого текста - четное число)

$m = L_0R_0$, где:

- $L_0, R_0 \in \{0,1\}^{\frac{n}{2}}$

теперь преобразовываем эти полублоки

$\forall i = \{1, \cdots, r\}$:

- $L_i = R_{i-1}$
- $R_i = L_{i-1} \bigoplus f(R_{i-1}, k_i)$
  - k_i - раундовый ключ, как-то получается из основного ключа

Проделываем процедуру и в конце получаем L_r, R_r.

Формула для криптограмы с:

$$c = R_rL_r$$

Как расшифровывать криптограмму с?

пусть $c = u_0v_0$

$\forall i \in \{1, \cdots, r\}$

- $u_i = v_{i-1}$
- $v_i = u_{i-1} \bigoplus f(v_{i-1}, k_{r+1-i})$

Проделываем процедуру и получаем v_r, u_r

Тогда формула для открытого текста это

$$m = v_ru_r$$

Д-ВО

С помощью индукции по i нужно показать, что:

- $u_i = R_{r-i}$
- $v_i = Lr - i$

Б.И

$i = 0 : \begin{cases} u_0 = R_r \\ v_0 = L_r \end{cases}$

Ш.И от $(i-1) \to i$

- по опр конструкции Фейстеля $L_i = R_{i-1}$

$u_i = v_{i-1} = [\text{П.И}] = L_{r-i+1} = R_{r-i}$

$v_i = u_{i-1} \bigoplus f(v_{i-1}, k_{r+1-i})$

- **по П.И:**
  - $u_{i-1} = R_{r-i+1} =$
  - $v_{i-1} = L_{r-i+1} = R_{r-i}$
  - подставим

$v_i = R_{i-i+1} \bigoplus f(R_{r-i}, k_{r+1-i})$

из $R_i = L_{i-1} \bigoplus f(R_{i-1}, k_i) \Rightarrow$

$L_{i-1} = R_i \bigoplus f(R_{i-1}, k_i) \Rightarrow$

По итогу получаем, что

$L_{r-i} = R_{r-i+1} + (R_{r-i}, k_{r-i+1})$

■

Конструкция расшифрования такая же как и шифрования, кроме порядка ключей

- **При шифровании ключи используются по возрастанию**
- **При расшифровании ключи используются по убыванию**

Нам не важно какую функцию f использовать, т.к не имеет значение её обратимость $\to$ **можем выбрать сколь угодно сложную функцию**

AES (Advanced Encryption Standard)

- Блок 128 бит
- Ключ 128, 192, 256 бит

$M_b = 4$ - длина блока в машинных словах, где одно слово это 32 бита

$N_k = 4, 6, 8$ - длика ключа в машинных словах

$N_r = N_k + 6$ - число раундов

- **Структура "квадрат"**

$State \in (\{0,1\}^8)^{4\times4}$

- текущее состояние обрабатываемого блока открытого текста это квадрат 4×4 байт
- в начале помещаем в state открытый текст

State = m

происходит обработка

c = State

Помещать и извлекать нужно по столбикам

$m = m_0 m_1 \cdots m_{15}$ разбили открытый текст на 16 кусочков размером с байт. Текст размещается по столбикам, как показано на картинке

Функции

SB - функция замены байт

$SB : \{0,1\}^8 \rightarrowtail\!\!\!\!\!\twoheadrightarrow \{0,1\}^8$

Байт - триедин:

- Это битовая цепочка $\{0,1\}^8$
- это число от 0 до 255
- байт это элемент 256 элементного поля $GF(256) = F$

чтобы построить $GF(256)$ используют неприводимый многочлен:

$f(x) = x^8 + x^4 + x^3 + x + 1$

Убедимся, что он неприводимый, т.е $\in N(\mathbb{Z}_2[x])$

все неприводимые над $\mathbb{Z}_2$ до 4 степени:

- x т.к нет корней, то не делитель

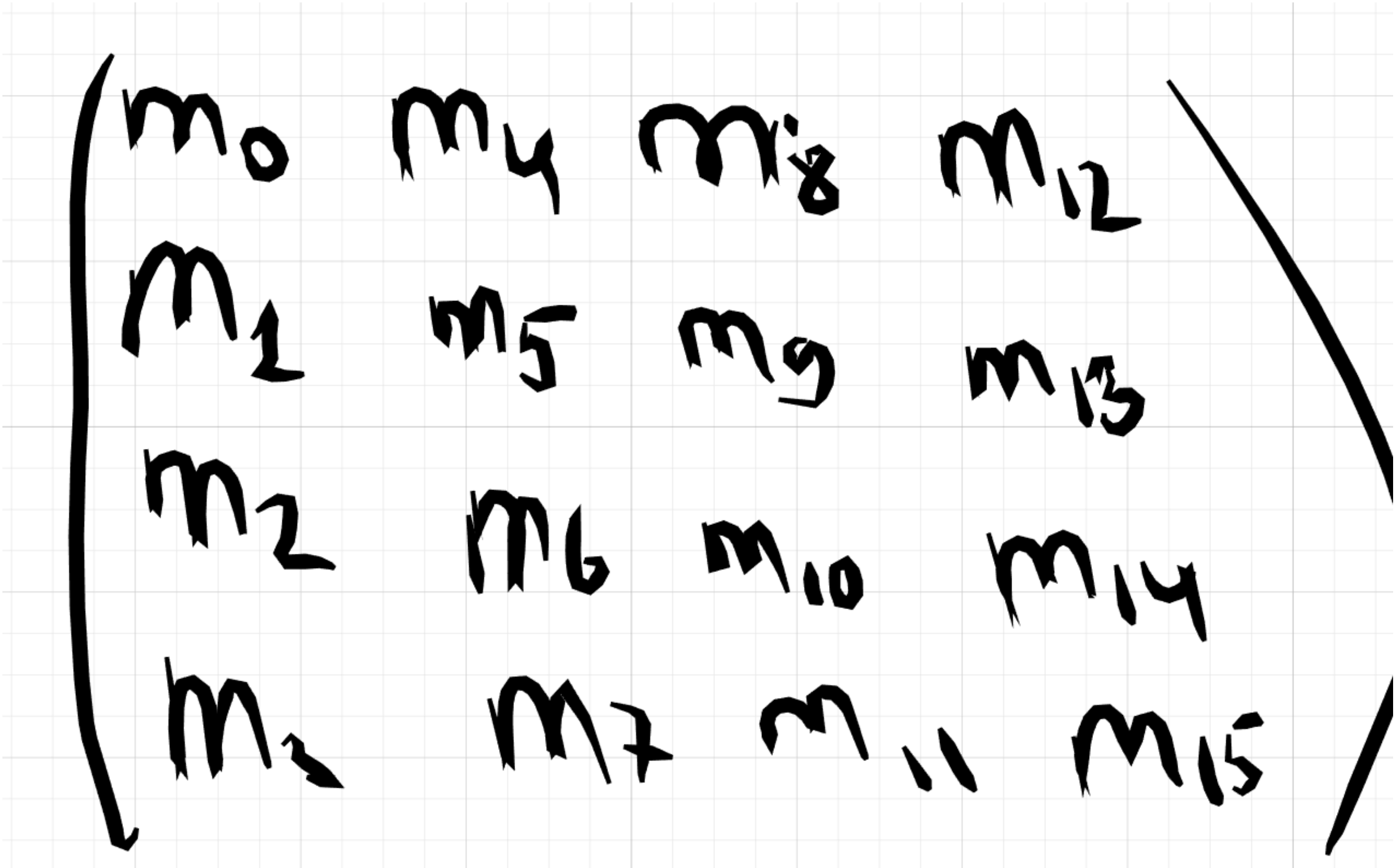


Рис. 1: alt text

- $x+1$ т.к нет корней, то не делитель
- x^2+x+1 не делитель, т.к он делит только x^4+x^3+x+1.
- x^3+x+1 не делитель. $x^4+x^8=x^4(1+x^4)$- здесь нет делителей
- x^3+x^2+1 не делитель. $x^8+1=(x+1)^8$ по биномиальной теореме
- x^4+x+1 не делитель.$x^8+x^3=x^3(x^5+1)=x^3(x+1)(x^4+x^3+x^2+x+1)$
- x^4+x^3+1 не делитель. $x^8+x=x(x^7+1)=x(x+1)(x^6+x^5+x^4+x^3+x^2+x+1)=x(x+1)(x^3+x^2+1)\cdot(x^3+x+1)$
- $x^4+x^3+x^2+x+1$ - не делитель. $x^8+x^2=x^2(x^6+1)=x^2(x^3+1)^2$

т.е $F=\mathbb{Z}_2[x]_{/f(x)\mathbb{Z}_2[x]}$ - фактор кольцо

- Каждый элемент этого поля - 8 битовая цепочка
- Каждая цепочка понимается как многочлен
- Т.е складывать и умножать их надо как многочлены - сложил/умножил, затем взял остато от деления на f(x)

Идеал - множество всех многочленов, замкнутых относительно сложения и умножения на любой элемент.

Сложение это обычный $\bigoplus$

Умножение - особенное

Функция SB

$SB(h)=(x^4+x^3+x^2+x+1)\cdot h^{-1}\bigoplus(x^6+x^5+x+1)\ (mod(x^8+1))$

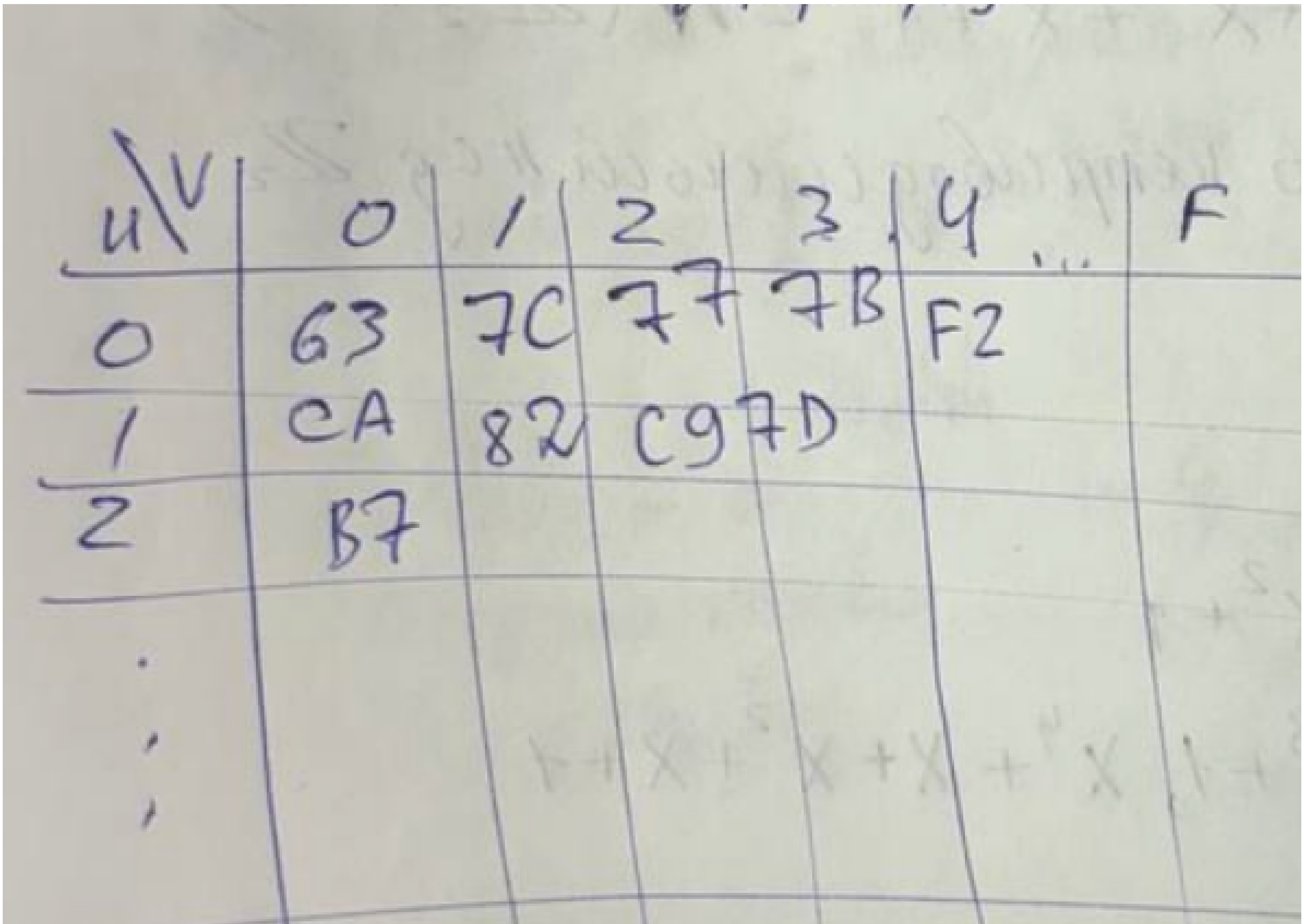
- h^{-1} **- это обращение в поле F**, т.е обращение делаем с помощью такого многочлена $f(x)=x^8+x^4+x^3+x+1$
- по договоренности $0^{-1}=0$
- умножение многочленов обычное, но берем остаток от x^8+1

Сделаем препроцессинг:

Каждому многочлену $h\leftrightarrow u||v$ * $u,v\in\{0,1,\cdots,E,F\}$ - 16 ричные цифры

$SB(h)\leftrightarrow ab$ * $a,b\in\{0,1,\cdots,E,F\}$ - 16 ричные цифры

для более удобного вычисления SB(h) заполняют специальную таблицу по типу такой



u\v	0	1	2	3	4	…	F
0	63	7C	77	7B	F2		
1	CA	82	C9	7D			
2	B7						
⋮							
F							

Рис. 2: alt text

	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

Рис. 3: alt text

Функция ISB

Попроси ещё раз её вывести

$ISB = SB^{-1}$
если $p = SB(h)$, то $h = ISB(p) =$
$(x^4 + x^3 + x^2 + x + 1)$ - неприводим над $\mathbb{Z}_2$
$(x^8 + 1) = (x + 1)^8$
$\Rightarrow (x^8 + 1)$ и $(x^4 + x^3 + x^2 + x + 1)$ - взаимо просты, ищем обратный по расширенному алгоритму евклида

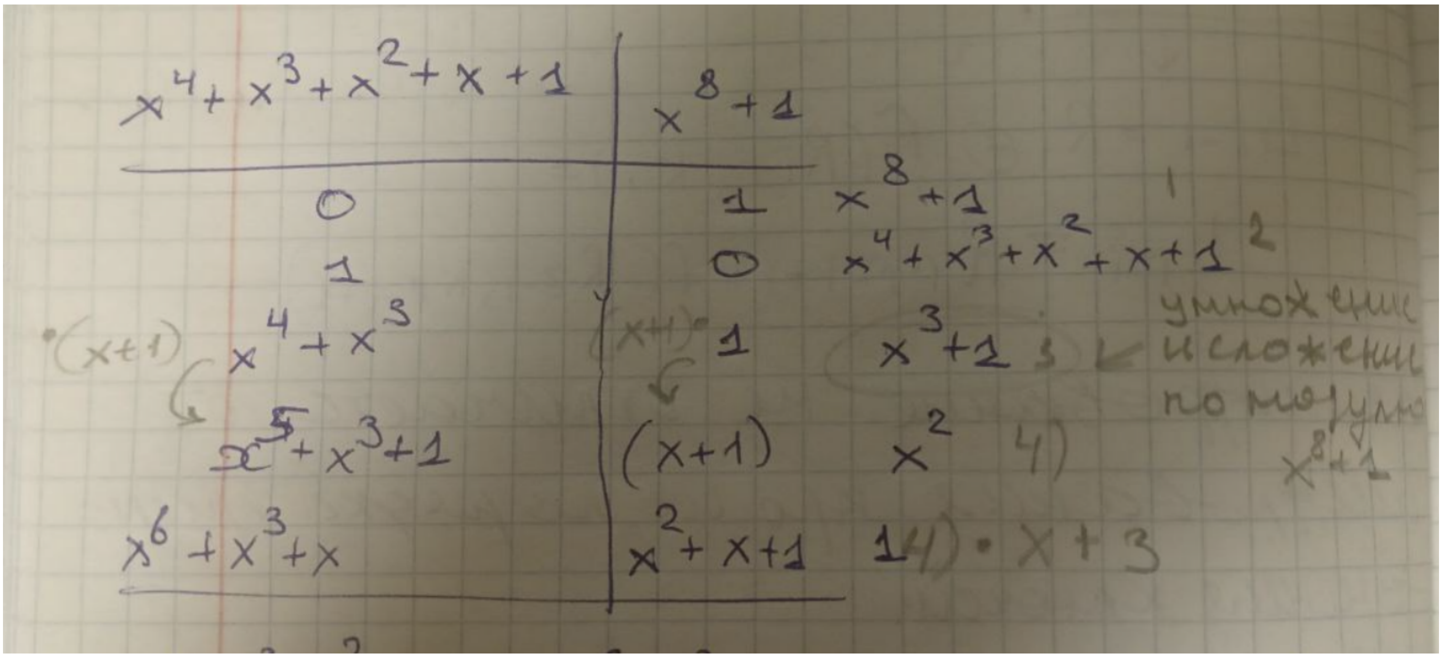


Рис. 4: alt text

нашли это $x^6 + x^3 + x$

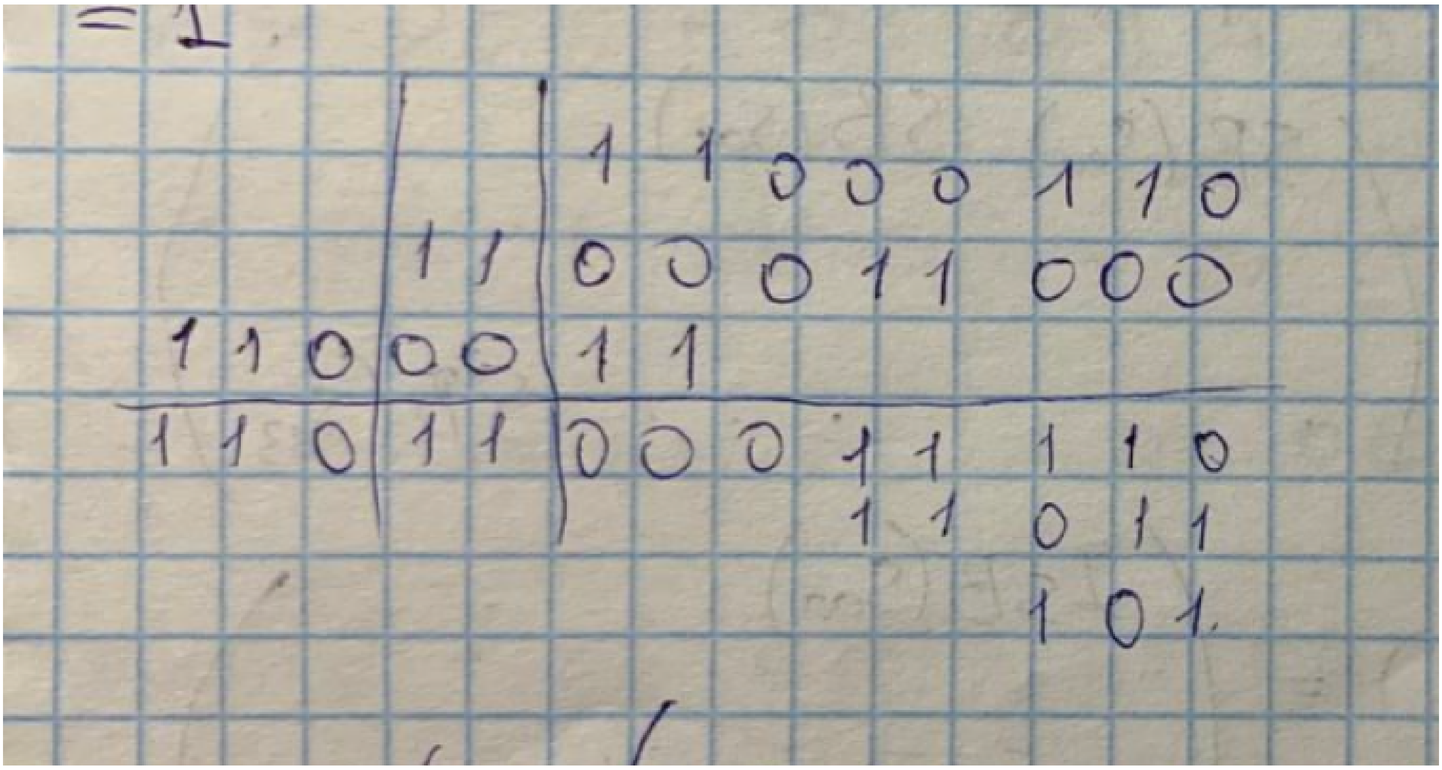


Рис. 5: alt text

$f = ISB(h) = ((x^6 + x^3 + x) \cdot h + (x^2 + 1) Mod (x^8 + 1))^{-1}$

- Для удобства $ISB(X||Y)$ можно задать табличкой
- X и Y это тоже 16ричные цифры

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
A	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
B	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
C	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
D	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
E	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
F	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рис. 6: alt text

- SB - сложное преобразование
- из SB можно изготовить функцию SubBytes(State)

$$State = \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{3,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix}$$

$$SubBytes(State) = \begin{pmatrix} SB(S_{0,0}) \cdots SB(S_{0,3}) \\ SB(S_{1,0}) \cdots SB(S_{1,3}) \\ SB(S_{2,0}) \cdots SB(S_{3,3}) \\ SB(S_{3,0}) \cdots SB(S_{3,3}) \end{pmatrix}$$

- из ISB можно изготовить аналогичную функцию InvSubBytes(State)

$$InvSubBytes(State) = \begin{pmatrix} ISB(S_{0,0}) \cdots ISB(S_{0,3}) \\ ISB(S_{1,0}) \cdots ISB(S_{1,3}) \\ ISB(S_{2,0}) \cdots ISB(S_{3,3}) \\ ISB(S_{3,0}) \cdots ISB(S_{3,3}) \end{pmatrix}$$

Функция ShiftRows(State)

- Сдвигает строки таим образом, чтобы диагональ стала первым столбиком

$$ShiftRows(State) = \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{pmatrix}$$

- эта функция обратима, т.е можно ввести InvShiftRows =
- InvShiftRows применяет к строкам циклический сдвиг вправо на 0, 1, 2, 3 байта соответственно:

$$\mathrm{InvShiftRows}(S) = \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,3} & s_{1,0} & s_{1,1} & s_{1,2} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,1} & s_{3,2} & s_{3,3} & s_{3,0} \end{pmatrix}$$

Рис. 7: alt text

Функция MixColumns(State)

$MixColumns(State) = A \cdot State$ где

- $A = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$
- умножение в кольце матриц $F^{4\times 4}$
- F - основное поле F = GF(256)
- эта функция обратима, для этого нужно показать, что A - обратимая матрица

$$\measuredangle B = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & OE & OB & OD \\ 0D & 09 & OE & OB \\ OB & OD & O9 & OE \end{pmatrix}$$

$\measuredangle A \times B$ * рассмотрим, только первую строку, т.к строки этой матрицы получены циклическим сдвигом влево

$\measuredangle 02 \cdot 0E + 03 \cdot 09 + 01 \cdot 0D + 01 \cdot 0B = 01$

$\measuredangle 02 \cdot 0B + 03 \cdot 0E + 01 \cdot 09 + 01 \cdot 0D = 00$

$\measuredangle 02 \cdot 0D + 03 \cdot 0B + 01 \cdot 0E + 01 \cdot 09 = 00$

$\measuredangle 02 \cdot 09 + 03 \cdot 0D + 01 \cdot 0B + 01 \cdot 0E = 00$

$\Rightarrow InvMixColumns(State) = B \cdot State$ * умножение в кольце $F^{4\times 4}$

Шифрование

Вход:

- массив State,
- $w[0,(N_r+1)\cdot N_b-1]$
  - элементы этого массива это 32 битные машинные слова.
  - $\forall w_i \in \{0,1\}^{32}$
- $\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3$ - 4 32битовых слова(4 байтных)

$$AddRoundKey(State, \vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3) = \begin{pmatrix} S_{0,0} \bigoplus \vec{x}_0[0] & \cdots & S_{0,3} \bigoplus \vec{x}_3[0] \\ S_{1,0} \bigoplus \vec{x}_0[0] & \cdots & S_{1,3} \bigoplus \vec{x}_3[0] \\ S_{2,0} \bigoplus \vec{x}_0[0] & \cdots & S_{2,3} \bigoplus \vec{x}_3[0] \\ S_{3,0} \bigoplus \vec{x}_0[0] & \cdots & S_{3,3} \bigoplus \vec{x}_3[0] \end{pmatrix}$$

- В начале в State помещаем открытый текст State = m

$AddRoundKey(State, w[0,\cdots,N_0-1])$

For Round = 1; Step = 1 To N_{r-1}:

$SubBytes(State)$

$ShiftRows(State)$

$MixColumns(State)$

$AddRoundKey$(State, w[Round $\cdot N_b$, Round $\cdot N_b + N_b - 1$])

EndFor

$SubBytes(State)$

$ShitRows(State)$

AddRoundKey(State, w[$N_r \cdot N_b, \cdots, (N_r+1)\cdot N_b - 1$])

c = State # шифрование окончено

Расшифровка

Это обратные функции в обратном поряке, ключи в обратном порядке

- функция AddRoundKey - имеет обратную, т.к это просто xor
- Можно применять обратные функции в том же порядке, что и прямые, но ключи в обратном порядке

Алгоритм построения ключа(Key schedule)

Вход

- ключ $Key \in \{0,1\}^{128 \text{ либо } 192 \text{ либо } 256}$
- $Key[0,1,\cdots N_{k-1}]$
  - $Key[i] \in 0, 1^{32}$_
  - массив 32 битовых слов длины N_k

I = 0

#инициализация WHILE(i < N_k): > w[I] = Key[I]

I += 1

End WHILE

#обработка

WHILE(I < $N_b \cdot (N_{r+1})$):

TEMP = w[i-1]

IF(I mod N_k == 0): > TEMP = ($Subword$($Rotword$(TEMP))) xor $R_{con}[\frac{I}{N_k}]$

else IF(N_k == 8 AND $I\ mod\ N_K$ == 4):

TEMP = $Subword$(TEMP)

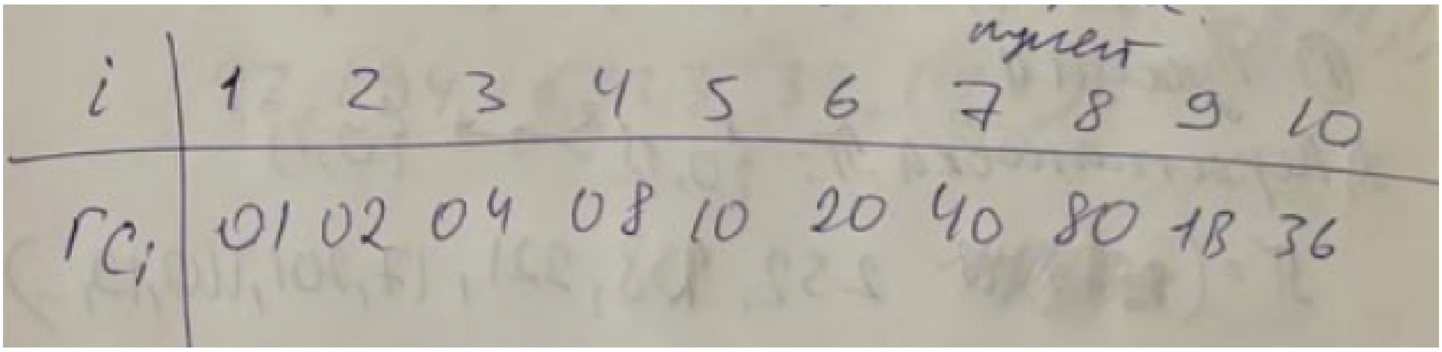
END IF

$w[i] = w[I - N_k]\ xor\ TEMP$

End WHILE

- $Subword : \{0,1\}^{32} >\twoheadrightarrow \{0,1\}^{32}$
  - $Subword(a_0, a_1, a_2, a_3) = (SB(a_0), SB(a_1), SB(a_2), SB(a_3))$
- $Rotword : \{0,1\}^{32} >\twoheadrightarrow \{0,1\}^{32}$

    - $Rotword(a_0, a_1, a_2, a_3) = (a_1, a_2, a_3, a_0)$
- $R_{con}[t] \in \{0, 1\}^{32}$
    - $R_{con}[t] = 0x\backslash rc_i\ 0x\backslash 00\ 0x\backslash 00\ 0x\backslash 00$
    - таблица значений для rc_i



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

Рис. 8: alt text

    –

Достоинства

1. AES быстро шифрует,т.к мало раундов, также большая длина блока, т.е быстрее обрабатывает открытый текст
2. Переменная длина ключа
3. Нет битовых операций. Кроме SB, а также умножения в поле F (всё это можно задать с помощью таблицы)

Недостатки

1. Могут быть секреты АМБ
2. Слишком молодой и не достаточно исследован