МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

АЛГОРИТМ AES — ПРИМЕР СОВРЕМЕННОГО СИММЕТРИЧНОГО КРИПТОПРЕОБРАЗОВАНИЯ

ПО КУРСУ «КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ»
ДЛЯ СТУДЕНТОВ ДНЕВНОЙ ФОРМЫ ОБУЧЕНИЯ СПЕЦИАЛЬНОСТЕЙ:
07.19.00 - КОМПЛЕКСНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ
БЕЗОПАСНОСТИ АВТОМАТИЗИРОВАННЫХ СИСТЕМ

2016

УДК 774:002:006.354

Составитель: Ю.А. Киселев, О. Е. Александров.

Научный редактор: канд. физ.-мат. наук О. Е. Александров

Алгоритм AES — Пример современного симметричного криптопреобразования: Методические указания к лабораторной работе / Ю.А. Киселев, О. Е. Александров. Екатеринбург: кафедра молекулярной физики УГТУ-УПИ, 2007. 28 с.

Описаны история возникновения и основы криптопреобразования AES. Подробно рассмотрен пример реализации шифрования этим методом, в виде демонстрационной программы на MathCAD. Описано устройство программы и работа с демонстрационной программой.

Материалы предназначены для студентов кафедры «Молекулярная физика».

Библиогр. 4 назв. Рис. . Табл. . Прил. 1.

Подготовлено кафедрой «Молекулярная физика».

Содержание

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ СИМВОЛОВ, ЕДИНИЦ И ТЕРМИНОВ

абонент	–	абонент криптосистемы, т.е. получатель или отправитель шифрованных сообщений;
алгоритм шифрования	–	последовательность действий (обычно математических), которые нужно проделать над исходным сообщением, чтобы получить шифрованное сообщение. Алгоритм шифрования обычно включает в себя и описание обратных действий — дешифровку. Все современные алгоритмы шифрования базируются на открытых алгоритмах, т.е. описание действий шифрования/дешифрования общедоступно.
криптосистема	–	информационная система, состоящая как минимум из двух абонентов, которые могут обмениваться шифрованными сообщениями по некоему каналу связи;
противник	–	кто угодно, кроме абонента криптосистемы, для которого предназначено шифрованное сообщение;
Массив	–	Упорядоченный набор идентичных объектов (например, массив байтов).
Бит	–	Бинарная цифра, принимающая значения 0 или 1.
Байт	–	Группа из восьми битов, которую рассматривают как единое целое или как массив из 8 битов.
Блок	–	Массив битов или байтов конечной длины.
Шифрование	–	Ряд преобразований, которые преобразуют обычный текст в зашифрованный, используя *Ключ шифрования*.
Ключ шифрования	–	Секретный ключ, который используется как параметр криптопреобразования при шифровании и дешифровании.
Зашифрованный текст	–	Текст получаемый в результате процесса шифрования.
Дешифрование	–	Ряд преобразований, которые преобразовывают зашифрованный текст в исходный текст, используя Ключ шифрования.
Расширение ключа	–	Операция алгоритма AES по созданию массива Раундовых ключей из Ключа шифра.

Раундовый ключ – Раундовые ключи получаются из Ключа шифра используя операцию Расширение ключа и используются на различных этавах криптопреобразования.

state – Блок данных, к которому применяется криптопреобразование. В AES *state* может быть представлен в виде прямоугольной матрицы байт из 4 строк и 4 колонок.

S-box – Таблица замены, используемая в нескольких байтовых преобразованиях и в процессе Расширения ключа, для замены одного байта другим.

Слово – Группа из 32 бит, которую рассматривают или как единое целое, или как множество 4 байтов.

m – исходное сообщение;

c – шифрованное сообщение, полученное из m;

0111b – суффикс «b» означает число записанное по основанию 2 — двоичное число;

0ABCh – суффикс «h» означает число, записанное по основанию 16 — шестнадцатеричное число;

111. – суффикс «.» означает число, записанное по основанию 10 — десятичное число;

$N_1..N_2$ – диапазон целых чисел от N_1 до N_2;

AES – Advanced Encryption Standard

ВВЕДЕНИЕ

XXI век - век информатики и информатизации, основной ценностью является информация. Технология дает возможность передавать и хранить все большие объемы информации. Это благо имеет и оборотную сторону. Информация становится все более уязвимой по разным причинам:

- возрастающие объемы хранимых и передаваемых данных;
- расширение круга пользователей, имеющих доступ к информационным системам;
- усложнение информационных систем.

Большую важность имеет проблема защиты информации от несанкционированного доступа (НСД) при передаче и/или хранении. Испытанный метод защиты информации от НСД — шифрование. Шифрованием (encryption) называют процесс преобразования исходных данных (plaintext, например, текстового документа на естественном языке) в шифрованные

(ciphertext), нечитаемые без знания специальных параметров преобразования — ключа. Дешифрованием называют обратное преобразование шифрованных данных в открытые. В англоязычной литературе шифрование/дешифрование — enciphering/deciphering.

Все известные алгоритмы шифрования делятся на два типа.

- Симметричные — с единственным секретным ключом для шифрования и дешифрования (они же single-key). Симметричные алгоритмы также подразделяются на два семейства.
  - Потоковые — шифрование данных посимвольно.
  - Блочные — шифрование данных кусками (блоками) из нескольких символов конечной и фиксированной длины.
- Асимметричные — с двумя ключами: открытым ключом (или public-key) и закрытым ключом (или private-key). Первый ключ служит только для шифрования, второй — для дешифрования.

Каждый из указанных типов криптоалгоритмов имеет свои достоинства и недостатки. Так основным недостатком симметричных методов является необходимость организации закрытого канала для передачи ключа. А основным достоинством — быстрота выполнения криптопреобразования. И наоборот, асимметричные алгоритмы более медленные в выполнении криптопреобразования, но не требуют закрытых каналов обмена ключами, т.к. открытый ключ не позволяет произвести дешифровку, а передавать закрытый ключ не нужно.

В данной лабораторной работе будет рассмотрен один из современных стандартов симметричного шифрования – AES. С другим примером симметричного криптоалгоритма — ГОСТ 28147-89 — можно ознакомиться в [1].

1. ИСТОРИЯ ВОЗНИКНОВЕНИЯ И ОСНОВНЫЕ ПОНЯТИЯ AES

В 1998 году NIST (National Institute of Standards and Technology) объявил конкурс на создание алгоритма симметричного шифрования, алгоритм получил название AES (Advanced Encryption Standard). Алгоритм планировалось принять как стандарт Соединенных Штатов Америки взамен устаревшего к этому времени стандарту DES (Digital Encryption Standard), являвшегося американским стандартом с 1977 года. Необходимость в принятии нового стандарта была вызвана небольшой длиной ключа DES (56 бит), что позволяло успешно применять метод прямого перебора ключей для взлома DES. Кроме того, архитектура DES была ориентирована на аппаратную реализацию, и

программная реализация алгоритма на платформах с ограниченными ресурсами не давала достаточного быстродействия. Модификация TripleDES обладала достаточной длиной ключа, но при этом была еще медленнее.

2 января 1997 года NIST объявляет о намерении выбрать преемника для DES. Конкурс был объявлен 12 сентября 1997г. Свой алгоритм могла предложить любая организация или группа исследователей. NIST опубликовал все данные о тестировании кандидатов на роль AES и потребовал от авторов алгоритмов сообщить о базовых принципах построения алгоритмов, используемых в них константах, таблицах для замен (S-box) и т.п. В отличие от ситуации с DES, NIST при выборе AES не стал опираться на секретные и, как следствие, запрещенные к публикации данные об исследовании алгоритмов-кандидатов.

Требования к кандидатам на новый стандарт были следующими:

- блочный шифр;
- длина блока, равная 128 битам;
- ключи длиной 128, 192 и 256 бит.

Дополнительно кандидатам рекомендовалось:

- использовать операции, легко реализуемые как аппаратно (в микрочипах), так и программно (на персональных компьютерах и серверах);
- ориентироваться на 32-разрядные процессоры
- не усложнять без необходимости структуру шифра для того, чтобы все заинтересованные стороны были в состоянии самостоятельно провести независимый криптоанализ алгоритма и убедиться, что в нём не заложено каких-либо недокументированных возможностей.

Кроме того, алгоритм, претендующий на то, чтобы стать стандартом, должен распространяться по всему миру на не эксклюзивных условиях и без платы за пользование патентом.

Алгоритм AES прежде всего должен предлагать высокую степень защиты, обладать простой структурой и высокой производительностью. Уже на уровне внутренней архитектуры он должен обладать надежностью, достаточной для того, чтобы противостоять будущим попыткам его взлома.

Был проведен конкурс среди алгоритмов шифрования на роль AES, 2 октября 2000 года было объявлено, что победителем конкурса стал алгоритм Rijndael и началась процедура стандартизации. 28 февраля 2001 года был опубликован проект, а 26 ноября 2001 года AES был принят как стандарт FIPS 197.

AES не тождественен Rijndael, т.к. оригинальный алгоритм Rijndael поддерживает более широкий диапазон длин ключей и блоков. В AES размер ключа фиксирован и равен 128 бит, а в Rijndael поддерживаются различные

длины ключей - от 128 до 256 бит, с шагом 32 бита. AES оперирует с блоком данных 16 байт, а Rijndael позволяет выбирать размер блока.

Rijndael – быстрый и компактный алгоритм с простой математической структурой. Он продемонстрировал хорошую устойчивость к атакам на реализацию, при которых пытаются декодировать зашифрованное сообщение, анализируя внешние проявления алгоритма, в том числе уровень энергопотребления и время выполнения. Алгоритму присущ внутренний параллелизм, что позволяет без труда обеспечить эффективное использование процессорных ресурсов. Далее под AES можно понимать Rijndael с ключом в 128 бит и блоком данных 16 байт.

2. ОПИСАНИЕ АЛГОРИТМА AES

2.1. Обозначения, используемые в описании алгоритма

В табл. приведены основные обозначения функций и параметров ариптопреобразования AES.

Таблица 2.1

Параметры алгоритма, символы и функции

Обозначение	Смысл обозначения
AddRoundKey()	Преобразование в Шифровании и Дешифровании, в котором Раундовый ключ добавляется к *state*, используя операцию XOR.
InvMixColumns()	Преобразование в Дешифровании, которое является инверсией *MixColumns*.
InvShiftRows()	Преобразование в Дешифровании, которое является инверсией *ShiftRows*.
InvSubBytes()	Преобразование в Дешифровании, которое является инверсией *SubBytes*.
K	Ключ шифрования — массив из 128 бит или 16 байт.
MixColumns()	Преобразование в процессе Шифрования, которое берет все столбцы *state* и смешивает их данные (независимо друг от друга), чтобы получить новые столбцы.
Rcon()	Массив постоянных раундовых Слов.
RotWord()	Функция используемая в операции Расширения ключа, она берет четырехбайтовое слово и выполняет циклическую

	перестановку.
ShiftRows()	Преобразование в процессе Шифрования, которое выполняется над *state*, циклически сдвигая последние 3 строки *state* на различные значения.
SubBytes()	Преобразование в процессе Шифрования, которое выполняется над *state* и состоит в замене каждого байта, используя таблицу замены (*S-box*).
N_k	Длина ключа в словах, для AES 4 слова.
N_b	Длина блока в словах, для AES 4 слова.
N_r	Число раундов при шифровании – 10.
XOR	Операция «исключающее ИЛИ» над битами.
$\otimes$	Операция «исключающее ИЛИ» над битами.
$\bullet$	Умножение в конечном поле.

3. МАТЕМАТИЧЕСКИЕ ОСНОВЫ AES

Алгоритм AES оперирует байтами, т.е. числами от 0 до 255 или 8-и разрядными двоичными числами. Все байты в AES интерпретируются как элементы конечного поля $F(2^8)$. Конечное поле F — это множество произвольных элементов (членов множества), содержащее конечное число членов. Для элементов поля определены две бинарные операции: сложение двух элементов поля и умножение двух элементов поля, результатом которых для любых двух элементов поля является элемент поля. Среди элементов поля есть элемент «ноль», сложение с которым любого элемента дает в результате этот же элемент и умножение на который любого элемента дает элемент «ноль». Среди элементов поля есть элемент «единица», умножение на который любого элемента дает в результате этот же элемент. Для каждого элемента поля существует «мультипликативно обратный» элемент, произведение которых дает в результате элемент «единица».

Элементы конечного поля могут складываться и умножатся, но эти операции несколько отличаются от операций используемых для простых чисел.

3.1. Сложение

Сложение выполняется с помощью операции побитового сложения по модулю 2. Операция обозначается XOR или $\oplus$ и выполняется над двоичным представление числа поразрядно так, что $1 \oplus 1 = 0$, $1 \oplus 0 = 0 \oplus 1 = 1$, $0 \oplus 0 = 0$.

Для двух байт $A = \{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}$, где a_i - i-й разряд двоичного представления числа $A \in [0..255]$ и $B = \{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$ сумма будет равна байту $C = \{c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0\}$, где $c_i = a_i \oplus b_i$.

3.2. Умножение

Для умножения используется полиномиальное представление двоичного числа, так байт $A = \{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}$ может быть представлен в виде полинома

$$a(x) = a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0 \cdot x^0, \qquad (3.1)$$

где a_i - принимает значения 0 или 1.

В полиномиальном представлении, умножение (обозначенное $\bullet$) в конечном поле $F(2^8)$ (здесь число в скобках 2^8 указывает на число элементов конечного поля) выполняется между полиномами по модулю неприводимого в поле F полинома степени 8. Полином неприводим[1)] в поле F, если он делится только на себя и на единицу поля. Для алгоритма AES неприводимый полином это:

$$m(x) = x^8 + x^4 + x^3 + x + 1. \qquad (3.2)$$

Результат умножения двух байтов A и B есть остаток от деления произведения полиномиального представления для A на аналогичное полиномиального представление для B, выполненного по обычным правилам произведения полиномов на неприводимый полином $m(x)$.

Например, вычислим произведение

$$57 \bullet 83.$$

Полиномиальное представление 57 есть

$$x^6 + x^4 + x^2 + x + 1,$$

полиномиальное представление 83 есть

$$x^7 + x + 1.$$

Результат произведения полиномов

$$\begin{gathered}(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = \\ x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ + x^7 + x^5 + x^3 + x^2 + x + \\ x^6 + x^4 + x^2 + x + 1.\end{gathered}$$

[1)] Многочлен, который можно представить в виде произведения многочленов низших степеней с коэффициентами из данного поля, называется приводимым (в данном поле), в противном случае — неприводимым. Например, многочлен $x^4 + 2$, неприводимый в поле рациональных чисел, разлагается на два множителя в поле вещественных чисел и на четыре множителя в поле комплексных чисел.

Суммирование коэффициентов при одинаковых степенях производится по правилам сложения в поле, т.е. по модулю 2, например

$$1 \cdot x^7 + 1 \cdot x^7 = (1 \oplus 1) \cdot x^7 = 0 \cdot x^7.$$

Получим

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) =$$
$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1. \quad (3.3)$$

Вычисляя остаток от деления на , получим

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) =$$
$$= x^7 + x^6 + 1,$$

что является полиномиальным представлением числа 193. Таким образом

$$57 \bullet 83 = 193.$$

Взятие по модулю $m(x)$ гарантирует, что результат будет двоичным полиномом степени меньше 8, и таким образом может быть представлен байтом. В отличии от операции сложения, нет другой простой операции на уровне битов, чтобы выполнить умножение.

4. АЛГОРИТМ AES

4.1. Шифрование

Для шифрования необходим ключ. Размер ключа в алгоритме AES 128 бит, обычно ключ представляют как матрицу 4×4 байта.

В начале процесса шифрования, входные данные разбиваются на блоки размером 16 байт или 128 бит. Если полный размер данных не кратен 16 байтам — данные дополняются до размера кратного 16 байтам. Блок данных в алгоритме AES называется *state* и обычно представляется в виде матрицы 4×4 байта, см. рис. . Операция шифрования каждого блока данных проводится независимо от содержимого других блоков. По окончанию шифрования блока — матрица заполняется следующей порцией данных и процесс повторяется. В силу независимости шифрования одного блока от другого процесс шифрования хорошо поддается распараллеливанию.

ПРЕДСТАВЛЕНИЕ БЛОКА ДАННЫХ AES — STATE

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

Рис. 4.1

Каждый блок шифруется в несколько этапов — раундов. Схема криптопреобразования может быть записана так.

1. Расширение ключа *KeyExpansion*.
2. Начальная операция — *AddRoundKey* — суммирование с основным ключом.
3. 9 раундов из четырех шагов каждый.
   3.1. *SubBytes* — замена байтов *state* по таблице замен.
   3.2. *ShiftRows* — циклический сдвиг строк *state*.
   3.3. *MixColumns* — перестановка столбцов *state*.
   3.4. *AddRoundKey* — суммирование с раундовым ключом.
4. Заключительный 10-й раунд
   4.1. *SubBytes* — замена байтов *state* по таблице замен.
   4.2. *ShiftRows* — циклический сдвиг строк *state*.
   4.3. *AddRoundKey* — суммирование с раундовым ключом.

Далее подробно описано каждое из преобразований.

4.1.1. Преобразование *SubBytes*

Преобразование *SubBytes* – это нелинейная замена байт, проводящаяся над каждым байтом *state*, используя таблицу замены *S-box*, см. табл. . Цель применения таблицы замен — затруднить линейный и дифференциальный криптоанализ. Таблица замен в алгоритме AES фиксированная. В табл. числа представлены в шестнадцатеричной системе счисления, в этой системе счисления любое значение байта представимо не более чем двумя шестнадцатеричными разрядами.

Таблица 4.1

ТАБЛИЦА ЗАМЕНЫ БАЙТ *S-BOX* АЛГОРИТМА AES

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0

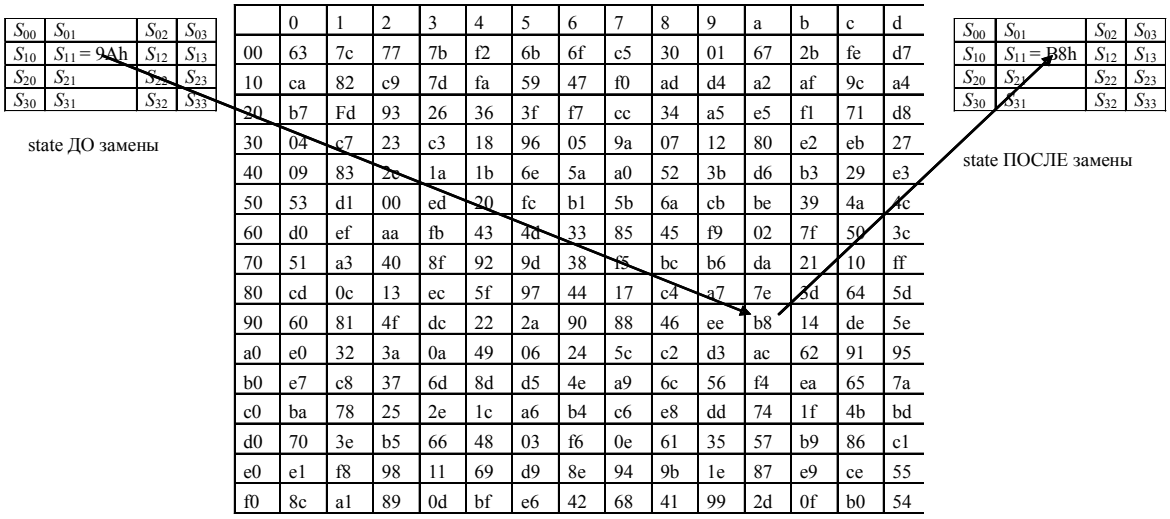
20	b7	Fd	93	26	36	3f	f7	cc	34	a5	e5	fl	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3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

Замена байта по таблице *S-box* производится так:

1. Байт Z преобразуется в шестнадцатеричную систему счисления, например XYh, X — старший разряд, Y — младший разряд. Если старшего разряда нет — он заменяется нулем.
2. В *S-box* выбирается строка X и столбец Y.
3. Значение Z' на пересечении строки X и столбца Y таблицы *S-box* используется как замена Z.

Например, для значения байта $Z = 9$Ah, заменой согласно таблицы *S-box* будет $Z' = $ B8h. Полный процесс *SubBytes* состоит в замене всех 16 байт матрицы *state*.

ЗАМЕНА БАЙТА В STATE



Пример замены одного байта с помощью *S-box*.

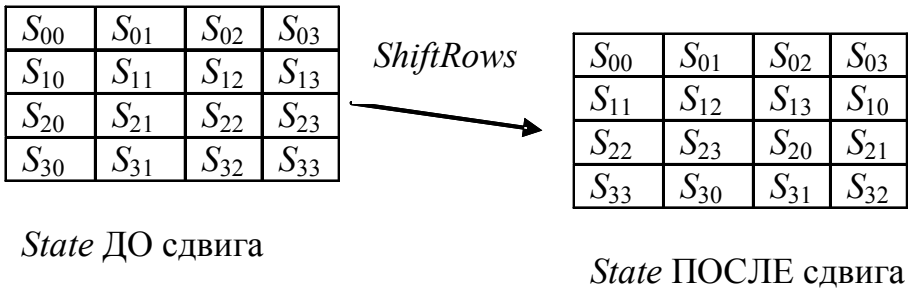
Рис. 4.2

ПРИМЕЧАНИЕ. Нет никакой необходимости использовать таблицу *S-box* в том виде, в котором она приведена в описании AES. Проще представить ее в виде одномерного массива байт, где индексом служит значение байта, а содержимым — содержимое соответствующей ячейки *S-box*. Например, *S-arr* = {63h, 7Ch, 77h, …}. Процесс замены по такой таблице сводится к $Z' = S\text{-}arr[Z]$, что заметно быстрее. Размер памяти, требующийся для размещения таблицы замен не изменится.

4.1.2. Преобразование *ShiftRows*

В преобразовании *ShiftRows* байты в последних трех строках *state* циклически смещаются влево на различное число байт. Строка 1 (нумерация строк с нуля, см. рис.) смещается на один байт, строка 2 – на два байта, строка 3 – на три байта. На рис. проиллюстрировано применение преобразования *ShiftRows* к *state*.

ПРЕОБРАЗОВАНИЕ СДВИГА AES — *SHIFTROWS*



State ДО сдвига

State ПОСЛЕ сдвига

Рис. 4.3

4.1.3. Преобразование *MixColumns*

В преобразовании *MixColumns* — перемешивание столбца — столбцы состояния (*state*) рассматриваются как полиномы над полем $F(2^8)$ и умножаются по модулю $x^4 + 1$ на постоянный полином:

$$a(x) = 3x^3 + 1x^2 + 1x^2 + 2 \qquad (4.1)$$

Процесс умножения полиномов эквивалентен матричному умножению

$$\begin{bmatrix} S'_{0c} \\ S'_{1c} \\ S'_{2c} \\ S'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S_{0c} \\ S_{1c} \\ S_{2c} \\ S_{3c} \end{bmatrix},$$

где c – номер столбца массива *state* и $0 \le c \le 3$. Операция сложения и умножения чисел выполняется по правилам, описанным в пунктах и .

В результате такого умножения, байты столбца c $\{S_{0c}, S_{1c}, S_{2c}, S_{3c}\}$ заменяются, соответственно, на байты

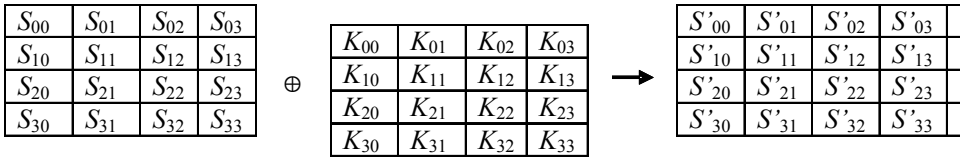
$$\begin{aligned} S'_{0c} &= (2 \bullet S_{0c}) \oplus (3 \bullet S_{1c}) \oplus S_{2c} \oplus S_{3c}; \\ S'_{1c} &= S_{0c} \oplus (2 \bullet S_{1c}) \oplus (3 \bullet S_{2c}) \oplus S_{3c}; \\ S'_{2c} &= S_{0c} \oplus S_{1c} \oplus (2 \bullet S_{2c}) \oplus (3 \bullet S_{3c}); \\ S'_{3c} &= (3 \bullet S_{0c}) \oplus S_{1c} \oplus S_{2c} \oplus (2 \bullet S_{3c}). \end{aligned} \qquad (4.2)$$

Преобразование применяется к каждому из четырех столбцов *state*.

4.1.4. Преобразование *AddRoundKey*

В преобразовании *AddRoundKey*, раундовый ключ *RK* добавляется к *state* посредством поразрядного XOR. Каждый раундовый ключ состоит из 16 байт расширенного ключа. Байты раундового ключа записываются в матрицу 4×4, подобную *state*. Каждый байт раундового ключа суммируется с соответствующим байтом из *state*, как показано на рис. .

СУММИРОВАНИЕ С РАУНДОВЫМ КЛЮЧОМ —
ADDROUNDKEY



State ДО суммирования

раундовый ключ

State ПОСЛЕ суммирования

Рис. 4.4

4.1.5. Расширение ключа *KeyExpansion*

Алгоритм AES берет ключ шифрования K и выполняет операцию расширения ключа, чтобы создать набор данных для раундовых ключей. Расширенный ключ W содержит 4×(10+1) слов — начальный ключ в 4 слова и по 4 слова расширенного ключа на каждый из 10 раундов. Расширенный ключ W состоит из слов (четыре байта на слово), обозначаемых ниже как w_i, где i находится в диапазоне [0..44]. Полная длина расширенного ключа 1408 бит, по 128 бит на каждый раунд.

В процессе расширения ключа используется массив констант *Rcon*. Элементы массива *Rcon* пронумерованы от 1 до 256+3. Значения элементов массива определены так

$Rcon_1 = 1$;

$Rcon_k = 2 \bullet Rcon_{k-1} = 2^{k-1}$, для $k = 2, 3, \ldots 255$;

$Rcon_k = 0$, для $k = 256, 257, 258$;

Расширение ключа можно описать следующей последовательностью операций.

1) Четыре слова ключа шифрования K копируются в первые четыре слова расширенного ключа W: $w_i = k_i$ для $i = 0, 1, 2, 3$.
2) Остальные слова расширенного ключа W для $i = 4, 5, \ldots 44$ генерируются так:
   - если i кратно 4, то $w_i = SubBytes(RotByte(w_{i-1})) \oplus Rcon_{(i/4)}$;
   - если i не кратно 4, то $w_i = w_{i-4} \oplus w_{i-1}$.

Функция *RotByte* переставляет четыре байта исходного слова $\{a_0, a_1, a_2, a_3\}$ с помощью циклической перестановки, превращая в слово $\{a_3, a_1, a_2, a_0\}$. Функция *SubBytes* применяет к каждому из четырех байтов слова замену по таблице *S-box*, как это описано в пункте .

4.1.6. Выбор раундового ключа

Раундовый ключ *RK* для раунда *k* выбирается из расширенного ключа *W* как слова с w_{4k} по $w_{4(k+1)}$.

4.2. Дешифрование

Все преобразования шифрования однозначны и, следовательно, имеют обратное преобразование, т.е. могут быть инвертированы и выполнены в обратном порядке, чтобы выполнить дешифрование для алгоритма AES.

Схема криптопреобразования может быть записана так.

1. Расширение ключа *KeyExpansion*.
3. 9 раундов из четырех шагов каждый.
   3.1. *AddRoundKey* — суммирование с раундовым ключом.
   3.2. *InvMixColumns* — обратная перестановка столбцов *state*.
   3.3. *InvShiftRows* — обратный циклический сдвиг строк *state*.
   3.4. *InvSubBytes* — обратная замена байтов *state* по таблице замен.
4. Заключительный 10-й раунд
   4.1. *AddRoundKey* — суммирование с раундовым ключом.
   4.2. *InvShiftRows* — обратный циклический сдвиг строк *state*.
   4.3. *InvSubBytes* — обратная замена байтов *state* по таблице замен.

Далее подробно описано каждое из преобразований.

4.2.1. Преобразование *InvMixColumns*

Преобразование *InvMixColumns* является обратным для преобразования *MixColumns*. В преобразовании *InvMixColumns*, столбцы состояния (*state*) рассматриваются как полиномы над полем $F(2^8)$ и умножаются по модулю $x^4 + 1$ с постоянным полиномом $d(x) = a^{-1}(x)$, в поле $F(2^8)$:

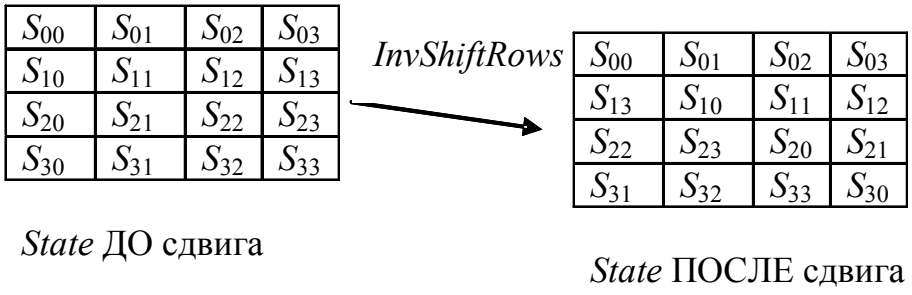
$$d(x) = 0\text{Bh}x^3 + 0\text{Dh}x^2 + 09\text{h}x + 0\text{Eh}. \qquad (4.3)$$

Подробнее о процессе умножения см. пункт .

4.2.2. Преобразование *InvShiftRows*

Преобразование *InvShiftRows* обратно преобразованию *ShiftRows*. Байты последних трех рядов массива *state* циклически сдвигаются вправо. Строка 1 (нумерация с нуля) смещается на 1 байт, строка 2 – на 2 байта, строка 3 – на 3 байта.

ПРЕОБРАЗОВАНИЕ ОБРАТНОГО СДВИГА AES — *INVSHIFTROWS*



State ДО сдвига

State ПОСЛЕ сдвига

Рис. 4.5

4.2.3. Преобразование *InvSubBytes*

Преобразование *InvSubBytes* выполняет обратную замену байт с помощью обратной таблицы замен, назовем ее *InvS-box*. Обратная таблица замен приведена в

Таблица 4.2

ТАБЛИЦА *INVS-BOX* ДЛЯ ОБРАТНОЙ ЗАМЕНЫ БАЙТ

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

Процесс замены байта по таблице подробно описан в пункте .

4.2.4. Преобразование обратное *AddRoundKey*

Преобразование *AddRoundKey*, описанное в пункте , является обратным для самого себя, так как использует операцию *XOR*.

5. ДЕМОНСТРАЦИОННАЯ РЕАЛИЗАЦИЯ КРИПТОАЛГОРИТМА AES НА MATHCAD

5.1. Общие замечания

Для наглядного представления о работе алгоритма шифрования AES он реализован в виде программы на MathCAD. Все преобразования, используемые алгоритмом AES при шифровании и дешифровании, реализованы в виде отдельных функций, что позволяет знакомится с ними детально. Далее будут подробно описаны основные функции.

Реализация AES на MathCAD-е имеет основной целью наглядную демонстрацию работы алгоритма, что приводит к существенному проигрышу в скорости работы. Кроме того, MathCAD 2001i не предоставляет встроенных функций для работы с битами, как следствие, такие функции приходится также реализовывать самостоятельно.

5.2. Вспомогательные функции работы с битовым представлением

Все вспомогательные функции, не относящиеся напрямую к криптоалгоритму AES, собраны в файле «Вспомогательные функции.mcd».

Для реализации криптоалгоритма AES требуются следующие вспомогательные функции.

Разложение целого числа в массив битов.

$$Int2Bits(i) := \left| \begin{array}{l} error(\text{"Int2Bits: ÷èñëî ñëèøêîì áîëüøîå"}) \ if \ i > cMaxInt \\ o \leftarrow ORIGIN \\ \mathbf{b}_o \leftarrow 0 \\ n \leftarrow o \\ while \ i > 0 \\ \quad \left| \begin{array}{l} \mathbf{b}_n \leftarrow mod(i, 2) \\ i \leftarrow floor\left(\frac{i}{2}\right) \\ n \leftarrow n + 1 \end{array} \right. \\ \mathbf{b} \end{array} \right.$$

Данная функция преобразует любое целое i число, меньшее $2^{50}+1$, в массив битов — вектор из 0 и 1 — где младшие биты идут первыми. Аналогично работает функция

$$Int2Bits2(i, r) := \left| \begin{array}{l} error(\text{"Int2Bits2: ÷èñëî i ñëèøêîì áîëüøîå"}) \ if \ i > cMaxInt \\ error(\text{"Int2Bits2: ÷èñëî r ñëèøêîì ìàëî"}) \ if \ i > 2^r \\ n \leftarrow ORIGIN \\ \mathbf{b}_{n+r-1} \leftarrow 0 \\ while \ i > 0 \\ \quad \left| \begin{array}{l} \mathbf{b}_n \leftarrow mod(i, 2) \\ i \leftarrow floor\left(\frac{i}{2}\right) \\ n \leftarrow n + 1 \end{array} \right. \\ \mathbf{b} \end{array} \right.$$

.

Которая преобразует любое целое число i, меньшее $2^{50}+1$, в массив битов фиксированной ширины r. Если целое число i имеет меньше ненулевых битов, чем r — массив дополняется нулевыми старшими битами до нужной ширины.

Преобразование массива битов в целое число.

$$Bits2Int(\mathbf{b}) := \left| \begin{array}{l} error(\text{"Bits2Int: ñòðîêà ñëèøêîì äëèííàÿ"}) \ if \ length(\mathbf{b}) > length(cBinStr2Int) \\ \mathbf{b} \cdot submatrix(cBinStr2Int, ORIGIN, last(\mathbf{b}), ORIGIN, ORIGIN) \end{array} \right.$$

Преобразует вектор из 0 и 1, где младшие биты идут первыми, в целое число.

Операция побитового XOR над двумя целыми числами.

```
XOR(a, e) := | error("T: ñëèøêîì áîëüøîå a"      ) if a > cMaxInt
             | error("T: ñëèøêîì áîëüøîå e"      ) if e > cMaxInt
             | a ← Int2Bits(a)
             | e ← Int2Bits(e)
             | e_last(a) ← 0  if length(a) > length(e)
             | a_last(e) ← 0  if length(e) > length(a)
             |      ---→
             | b ← (a ⊕ e)
             | Bits2Int(b)
```

Возвращает целое число, равное результату $a \oplus e$.

Вычисления остатка деления бинарного полинома X на бинарный полином **x**.

```
Modulo(X, x) := | o ← ORIGIN
                | error("Modulo: îïèñàíî x çàäàí íåâåðíî!"      ) if x_o = 0
                | lX ← last(X)
                | lx ← last(x)
                | return X  if lX < lx
                | x ← reverse(x)
                | X ← reverse(X)
                | for i ∈ o .. lX - lx
                |   if X_i = 1
                |     | n ← i - o
                |     | for j ∈ o .. lx
                |     |   X_{n+j} ← X_{n+j} ⊕ x_j
                | reverse(X)
```

Полиномы должны быть представлены как вектора из 0 и 1, младшие биты/степени ПЕРВЫЕ.

5.3. Вспомогательные функции преобразования данных

Эти функции приведены непосредственно в основном файле, демонстрирующем работу AES: «Криптоалгоритм AES.mcd».

Преобразование вектора из 16 байт в матрицу 4×4.

```
Vec2State(v, check) := if check
                         error(concat("Vec2State: íâåâðíàÿ äëèíà âåêòîðà v: ", num2str(length(v)), ". Äîëæíî áûòü 16."))  if length(v) ≠ 16
                         error(concat("Vec2State: â âåêòîðå v ñîäåðæèòñÿ ñëèøêîì áîëüøîå ÷èñëî: ", num2str(max(v)), ". Äîëæíî áûòü íå áîëüøå 255."))  if max(v) > 255
                         error(concat("Vec2State: â âåêòîðå k ñîäåðæèòñÿ ñëèøêîì ìàëåíüêîå ÷èñëî: ", num2str(min(v)), ". Äîëæíî áûòü íå ìåíüøå 0."))  if min(v) < 0
                         error("Vec2State: â âåêòîðå k ñîäåðæèòñÿ ÍÅöåëîå ÷èñëî.")  if v ≠ floor(v)
                       o ← ORIGIN
                       for i ∈ o .. o + 3
                         state^⟨i⟩ ← submatrix[v, o + (i - o)·4, o + (i - o)·4 + 3, o, o]
                       state
```

Данная функция используется при преобразовании строки в вектор блоков.

Перечислять все функции этого типа нет смысла — все они доступны в файле с комментариями.

5.4. Основные преобразования AES

Эти функции приведены непосредственно в основном файле, демонстрирующем работу AES: «Криптоалгоритм AES.mcd»

5.4.1. *Функция* умножения *AES — AESmult*

Функция выполняет операцию умножения двух байтов, как это описано в пункте .

$$
\begin{aligned}
&AESmul(a,b) := \\
&\quad o \leftarrow ORIGIN \\
&\quad \mathbf{a} \leftarrow \mathbf{Int2Bits2}(a,8) \\
&\quad \mathbf{b} \leftarrow \mathbf{Int2Bits2}(b,8) \\
&\quad al \leftarrow last(\mathbf{a}) \\
&\quad bl \leftarrow last(\mathbf{b}) \\
&\quad \mathbf{R}_{al+bl} \leftarrow 0 \\
&\quad \text{for } i \in al..o \\
&\quad\quad \text{for } j \in bl..o \\
&\quad\quad\quad \mathbf{R}_{i+j} \leftarrow \mathbf{R}_{i+j} \oplus \left(\mathbf{a}_i \wedge \mathbf{b}_j\right) \\
&\quad Bits2Int\left(\mathbf{Modulo}\left(\mathbf{R}, (1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1)^T\right)\right)
\end{aligned}
$$

Параметрами функции являются два целых числа. Функция представляет их в виде полиномов и умножает их между собой по модулю $x^8 + x^4 + x^3 + x + 1$.

5.4.2. Замена байтов по таблице замен

Функция выполняет замену байтов блока *state* по таблице замен *S-box*.

$$\textbf{SubBytes}(\textbf{state}, \textbf{Sbox}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ \text{for } i \in o \mathinner{..} o + rows(\textbf{state}) - 1 \\ \quad \text{for } j \in o \mathinner{..} o + cols(\textbf{state}) - 1 \\ \quad\quad \textbf{state}_{i,j} \leftarrow \textbf{Sbox}_{o+floor\left(\frac{\textbf{state}_{i,j}}{16}\right), o+mod(\textbf{state}_{i,j}, 16)} \\ \textbf{state} \end{array} \right.$$

Обратное преобразование выполняется этой же функцией, только вместо таблицы замен *S-box*, ей передают обратную таблицу замен *InvS-box*.

5.4.3. Сдвиг строк блока

Функция выполняет циклический сдвиг строк блока *state* ВЛЕВО на различное число байт, как это описано в пункте .

$$\textbf{ShiftRows}(\textbf{state}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ \textbf{s} \leftarrow \textbf{state} \\ \text{for } i \in o + 1 \mathinner{..} o + rows(\textbf{state}) - 1 \\ \quad \text{for } j \in o \mathinner{..} o + cols(\textbf{state}) - 1 \\ \quad\quad \textbf{s}_{i,j} \leftarrow \textbf{state}_{i, mod(j+i-2o, 4)} \\ \textbf{s} \end{array} \right.$$

Обратная функция, см. пункт

$$\textbf{invShiftRows}(\textbf{state}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ \textbf{s} \leftarrow \textbf{state} \\ \text{for } i \in o + 1 \mathinner{..} o + rows(\textbf{state}) - 1 \\ \quad \text{for } j \in o \mathinner{..} o + cols(\textbf{state}) - 1 \\ \quad\quad \textbf{s}_{i, mod(j+i-2o, 4)} \leftarrow \textbf{state}_{i,j} \\ \textbf{s} \end{array} \right.$$

5.4.4. Операция перемешивания столбцов

Функция выполняет операцию перемешивания столбцов блока *state*, как это описано в пункте

$$\textbf{MixColumns}(\textbf{state}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ \text{for } i \in o \mathinner{..} o + 3 \\ \quad \textbf{s}^{\langle i \rangle} \leftarrow \begin{pmatrix} (2\ AESmul\ \textbf{state}_{o,i})\ XOR\ (3\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ \textbf{state}_{o+2,i}\ XOR\ \textbf{state}_{o+3,i} \\ \textbf{state}_{o,i}\ XOR\ (2\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ (3\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ \textbf{state}_{o+3,i} \\ \textbf{state}_{o,i}\ XOR\ \textbf{state}_{o+1,i}\ XOR\ (2\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ (3\ AESmul\ \textbf{state}_{o+3,i}) \\ (3\ AESmul\ \textbf{state}_{o,i})\ XOR\ \textbf{state}_{o+1,i}\ XOR\ \textbf{state}_{o+2,i}\ XOR\ (2\ AESmul\ \textbf{state}_{o+3,i}) \end{pmatrix} \\ \textbf{s} \end{array} \right.$$

Обратное преобразование (см. пункт)

$$\textbf{invMixColumns}(\textbf{state}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ for\ i \in o..o+3 \\ \quad \left\langle i \right\rangle \leftarrow \begin{pmatrix} (0eh\ AESmul\ \textbf{state}_{o,i})\ XOR\ (0bh\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ (0dh\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ (09h\ AESmul\ \textbf{state}_{o+3,i}) \\ (09h\ AESmul\ \textbf{state}_{o,i})\ XOR\ (0eh\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ (0bh\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ (0dh\ AESmul\ \textbf{state}_{o+3,i}) \\ (0dh\ AESmul\ \textbf{state}_{o,i})\ XOR\ (09h\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ (0eh\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ (0bh\ AESmul\ \textbf{state}_{o+3,i}) \\ (0bh\ AESmul\ \textbf{state}_{o,i})\ XOR\ (0dh\ AESmul\ \textbf{state}_{o+1,i})\ XOR\ (09h\ AESmul\ \textbf{state}_{o+2,i})\ XOR\ (0eh\ AESmul\ \textbf{state}_{o+3,i}) \end{pmatrix} \\ \textbf{s} \end{array} \right.$$

5.4.5. Суммирование с раундовым ключом

Функция выполняет операцию добавления раундового ключа к блоку *state*, как это описано в пункте

$$\textbf{AddRoundKey}(\textbf{state}, \textbf{xK}, r) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ for\ i \in o..o+3 \\ \quad \textbf{s} \leftarrow \overrightarrow{XOR}(submatrix(\textbf{xK}, o, o+3, o+r \cdot 4, o+r \cdot 4+3), \textbf{state}) \end{array} \right.$$

5.4.6. Расширение ключа

Функция выполняет операцию расширения ключа шифрования *K*, как это описано в пункте .

$$\textbf{KeyExpand}(\textbf{K}, \textbf{Sbox}) := \left| \begin{array}{l} o \leftarrow ORIGIN \\ for\ i \in o+4..o+43 \\ \quad if\ mod(i,4) = 0 \\ \quad\quad \left\langle i \right\rangle \leftarrow \textbf{SubBytesV}\begin{pmatrix} \textbf{K}_{o+1,i-1} \\ \textbf{K}_{o+2,i-1} \\ \textbf{K}_{o+3,i-1} \\ \textbf{K}_{o,i-1} \end{pmatrix}, \textbf{Sbox} \\ \quad\quad \left\langle i \right\rangle \leftarrow \overrightarrow{XOR}\left(\left\langle i \right\rangle, \textbf{Rcon}_{o+\frac{i}{4}-1} \right) \\ \quad\quad \left\langle i \right\rangle \leftarrow \overrightarrow{XOR}\left(\left\langle i \right\rangle, \left\langle i-4 \right\rangle \right) \\ \quad otherwise \\ \quad\quad 1 \\ \quad\quad \left\langle i \right\rangle \leftarrow \overrightarrow{XOR}\left(\left\langle i-1 \right\rangle, \left\langle i-4 \right\rangle \right) \\ \textbf{K} \end{array} \right.$$

5.4.7. Шифрование и дешифрование блока

Функция выполняет полный цикл шифрования одного блока *state*, с расширенным ключом *xK*, как это описано в пункте

```
Encrypt(state, Sbox, xK) := | state ← AddRoundKey(state, xK, 0)
                            | for i ∈ 1 .. 9
                            |     | state ← SubBytes(state, Sbox)
                            |     | state ← ShiftRows(state)
                            |     | state ← MixColumns(state)
                            |     | state ← AddRoundKey(state, xK, i)
                            | state ← SubBytes(state, Sbox)
                            | state ← ShiftRows(state)
                            | AddRoundKey(state, xK, 10)
```

Обратная операция — дешифрование блока:

```
Decrypt(state, invSbox, xK) := | state ← AddRoundKey(state, xK, 10)
                               | for i ∈ 9 .. 1
                               |     | state ← invShiftRows(state)
                               |     | state ← SubBytes(state, invSbox)
                               |     | state ← AddRoundKey(state, xK, i)
                               |     | state ← invMixColumns(state)
                               | state ← invShiftRows(state)
                               | state ← SubBytes(state, invSbox)
                               | AddRoundKey(state, xK, 0)
```

5.4.8. Шифрование и дешифрование произвольной строки

Функция выполняет шифрование произвольной строки ключом K

```
EncryptStr(s, K) := | o ← ORIGIN
                    | xK ← KeyExpand(K, Sbox)
                    | m ← Str2Blocks(s)
                    | for i ∈ o .. last(m)
                    |   m_i ← Encrypt(m_i, Sbox, xK)
                    | Blocks2Str(m, 0)
```

Дешифрование произвольной строки ключом K

```
DecryptStr(s, K) := | o ← ORIGIN
                    | xK ← KeyExpand(K, Sbox)
                    | m ← Str2Blocks(s)
                    | for i ∈ o .. last(m)
                    |   m_i ← Decrypt(m_i, invSbox, xK)
                    | Blocks2Str(m, 0)
```

6. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

Варианты 0.*n* — стандартные задания. Выполнение такого варианта позволяет гордо говорить: «Я делал лабораторную работу». Одно из стандартных заданий выполняется в обязательном порядке, независимо от выполнения/невыполнения заданий повышенной сложности.

Варианты 1.*n* — задания повышенной сложности, за ПОЛНОЕ выполнение любого из них автоматически выставляется зачет по курсу. Выполнение заданий повышенной сложности — по собственному желанию и только как ДОПОЛНЕНИЕ к основному. Т.е. сначала надо сделать основное, а затем — повышенной сложности.

Выполненное задание состоит из

- отчета по лабораторной работе, где описан СОБСТВЕННЫЙ вклад в решение проблемы. Переписывать методичку и Интернет не надо.
- работающей программы на MathCAD. Другие варианты программ не принимаются к рассмотрению, кроме случаев, оговоренных в заданиях повышенной сложности.

Вариант 0.1

Используя методичку и программу изучить пошаговое действие алгоритма AES. Установить экспериментально значения исходных строк, при которых программа ПЕРЕСТАЕТ функционировать правильно. Попытаться объяснить эти ограничения. Изменяя размер исходной строки N пределах от одного символа до 1000 символов произвести измерения затрат времени Δt на шифрование, дешифрование AES. Замеры произвести на одном и том же ком компьютере не менее чем в 20 точках диапазона. Построить графики зависимости Δt от N. Предложить и обосновать теоретическую зависимость Δt от N в пределе больших N. Прокомментировать результаты с точки зрения оценки производительности AES.

Ответить на вопросы:

1. Зачем (ну свои соображения) нужна таблица замен и как она повышает криптостойкость алгоритма?
2. Оценить сравнительную криптостойкость AES и ГОСТ 28147-89. Какой из алгоритмов более стоек?

Вариант 0.2

Используя методичку и программу изучить пошаговое действие алгоритма AES. Разработать алгоритм взлома AES прямым перебором ключа. Прокомментировать результаты с точки зрения проблем реализации метода и оценки криптостойкости AES к этому методу криптоанализа.

Вариант 0.3

Используя методичку и программу изучить пошаговое действие алгоритма AES. Исследовать сохранение частотных и корреляционных характеристик длинных строк при шифровании алгоритмом AES. Прокомментировать результаты с точки зрения проблем реализации методов криптоанализа и оценки криптостойкости AES.

Вариант 1.1

Используя данную программу как пример для подражания, реализовать любой другой БОЛЕЕ ЭФФЕКТИВНЫЙ вариант алгоритма AES. Под эффективностью понимается скорость выполнения криптопреобразования. Для каждого шага своей реализации алгоритма привести подробное описание. Создать демонстрационную программу, подтверждающую работоспособность реализации алгоритма. Дать оценки производительности алгоритма по сравнению с готовым примером. Программы принимаются ТОЛЬКО в Math-CAD.

Вариант 1.2

Реализовать на Delphi (ну или на C/C++/pithon/perl/…) программу шифрования/дешифрования произвольного файла. Программа должна иметь интерфейс командной строки. В качестве образца для подражания использовать программу шифрования произвольного файла из лабораторной работы [Error: Reference source not found]. Сравнить быстродействие реализации ГОСТ 28147-89 и AES. Прокомментировать причины различия.

Вариант 1.3

Реализовать ЛЮБОЙ другой метод симметричного криптопреобразования (из числа реально применяемых). Написать руководство (в качестве примера см. настоящее руководство) и демонстрационную программу. Программы принимаются ТОЛЬКО в MathCAD.

Вариант 1.4

Найти или придумать иной алгоритм криптоанализа AES (кроме прямого перебора). Продемонстрировать его работоспособность в виде программы. Программы принимаются ТОЛЬКО в MathCAD.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 . Криптографические алгоритмы. Алгоритм криптопреобразования - ГОСТ 28147-89: Методические указания к лабораторной работе / С.А.Кабаков, О.Е.Александров. Екатеринбург: УГТУ-УПИ, 2002. 32 с