

Министерство образования Республики Беларусь  
Учреждение образования  
«Белорусский государственный университет  
информатики и радиоэлектроники»

Кафедра радиотехнических систем

**С.Б. Саломатин**

***ИССЛЕДОВАНИЕ СВОЙСТВ БЛОЧНЫХ  
КРИПТОСИСТЕМ***

Учебно-методическое пособие  
по курсу «Кодирование и защита информации»  
для студентов специальностей  
I-39 01 02 «Радиоэлектронные системы»,  
I-39 01 03 «Радиоинформатика»,  
I-39 01 04 «Радиоэлектронная защита информации»  
дневной формы обучения

Минск 2006

УДК 621. 391.25 (075.8)  
ББК 32.811 я 73  
С 16

Рецензент:  
д-р техн. наук., проф. В.К. Конопелько

**Саломатин С.Б.**

С 16      Исследование свойств блочных криптосистем. Учебно-метод. пособие по курсу «Кодирование и защита информации» для студ. спец. I-39 01 02 «Радиоэлектронные системы», I-39 01 03 «Радиоинформатика», I-39 01 04 «Радиоэлектронная защита информации» дневн. формы обуч. / С.Б. Саломатин. – Мн.: БГУИР, 2006. – 31 с. : ил.  
ISBN 985-444-992-0

Пособие содержит теоретические сведения, алгоритмы, программы моделирования базовых криптографических функций, процессов шифрования и расшифрования, а также криптографического анализа стандарта AES. Оценивается устойчивость криптосистемы по отношению к дифференциальному и линейному криптоанализу, атаке «Квадрат», уровню нелинейности криптограмм шифра.

**УДК 621. 391. 25 (075.8)**  
**ББК 32.811 я 73**

**ISBN 985-444-992-0**

© Саломатин С.Б., 2006  
© БГУИР, 2006

## ВВЕДЕНИЕ

Криптографические преобразования данных являются наиболее эффективным средством обеспечения конфиденциальности данных, их целостности и доступности. Во всем разнообразии криптографических механизмов защиты блочные криптосистемы занимают центральное место, как системы с высоким быстродействием надежностью.

В течении ряда лет самым распространенным криптоалгоритмом в мире, по сути многолетним стандартом криптозащиты, считался алгоритм DES. Его основные размеры: размер блока 64 бита, длина ключа 56 бит, 16 раундов. В основе шифра лежит структура Фейстела. Однако высокий уровень развития вычислительных средств позволяет сегодня вскрывать DES путем перебора ключей, хотя в остальных отношениях шифр подтверждает свою теоретическую стойкость.

В 2001 году после проведения открытого международного конкурса в качестве американского стандарта криптографической защиты AES принят шифр Rijndael, построенный на математическом аппарате конечных полей.

Криптосистема AES (Rijndael) имеет новую архитектуру («Квадрат»), обеспечивающую быстрое рассеивание и перемешивание информации, при этом за один раунд преобразованию подвергается весь входной блок. Все раундовые преобразования суть операции в конечных полях и допускают эффективную программную и аппаратную реализации на различных платформах. Криптосистема имеет байт-ориентированную структуру. Однако Rijndael заметно сложнее в описании, чем криптосистемы RC6 и ГОСТ 28147-89.

В методическом пособии рассматриваются основы криптологии блочных, симметричных систем. Приводится описание шифра AES с использованием алгебраических структур и конечных полей, а также основные свойства базовых операций, их обоснование с точки зрения устойчивости шифра по отношению атак криптоанализа.

Приводятся примеры моделирования в среде Maple процессов шифрования, расшифрования, работы базовых операций и криптографических функций, анализ их структурных свойств.

## 1. БЛОЧНЫЕ ШИФРЫ

Блочный шифр можно определить как зависящее от ключа  $K$  обратимое преобразование (обратимая функция) слова (входного значения)  $X$  из  $n$  двоичных символов:

$$C = E_K(X), X = D_K(C) = E_K^{-1}(C) = X,$$

где  $D_K = E_K^{-1}$  - обозначает расшифровывающее преобразование,  $C$  – слово (или блок, выходное значение) шифра.

Для криптографических приложений блочный шифр должен удовлетворять ряду требований, зависящих от ситуации, в которой он используется. Шифр считается стойким (при атаке по выбранному тексту), если для него неизвестны алгоритмы взлома, существенно более эффективные, чем прямой перебор ключей.

*Блочные составные шифры.* Детерминированный шифр  $G$  можно определить следующим образом  $G = (X, C, K, E)$ , где  $E: X \times K \rightarrow C$ . Допустим, что шифр определяется семейством преобразований  $G_i$ , имеющих общие пространства входных и выходных значений  $X_i = C_i = M$ , при этом результат действия функции  $E_i$  зависит от ключевого элемента  $k_i \in K_i$ . На основе этого семейства с помощью операции композиции можно построить шифр, задаваемый отображением

$$E: M \times (K_1 \times K_2 \times \dots \times K_r) \rightarrow M,$$

причем  $E = E_r \cdot \dots \cdot E_2 E_1$ , а ключом является вектор  $(k_1, k_2, \dots, k_r) \in K_1 \times \dots \times K_r$ .

Преобразование  $E_i$  называется  $i$ -м раундом шифрования, ключ  $k_i$  - раундовым ключом. Если ключевые пространства  $K_i$  и преобразования  $E_i$  для всех раундов совпадают, то такой составной шифр называется итерационным.

Идея (по К. Шеннону), лежащая в основе составных блочных шифров, состоит в построении системы путем многократного применения относительно простых преобразований, типа подстановка и перестановка, которые многократно используются.

К.Шенноном сформулированы принципы построения преобразований информации для криптографических систем. Важнейшими из них являются принципы:

- “перемешивания” – дает неформальную трактовку понятия для эргодических систем с конечным числом состояний;
- “рассеивания” (diffusion) посредством которого «...статистическая структура сообщений, которая приводит к избыточности в сообщениях, «распыляется» в статистику больших длин, то есть в статистику структур, включающую длинные комбинации букв криптограмм». Рассеивание предполагает

распространение влияние одного знака открытого текста, а также одного знака ключа на значительное количество знаков шифротекста.;

- “запутывания” (confusion), который “состоит в том, что соотношения между простыми статистиками в пространстве криптограмм и простыми подмножествами в пространстве ключей делаются весьма сложными и беспорядочными”.

Математическим объектом, моделирующим преобразования информации в криптосистемах, являются дискретные функции, которые представляют собой отображения конечных алгебраических объектов – множеств, групп, колец, полей, векторных пространств.

В блочных криптосистемах раундовые шифры строятся в основном с использованием операций замены двоичных кодов (схемы, реализующие эту нелинейную операцию называются S-блоками), перестановки элементов двоичных кодов, арифметических и логических операций над двоичными кодами. В качестве ограничения используется запрет на применение в двух соседних раундах шифрования преобразований, имеющих общую прозрачность. Пусть  $E: x \rightarrow y, y \in M$ , и на множестве  $M$  определены преобразования  $g$  и  $h$ . Если  $E(g(x)) = h(y)$ ,  $E$  прозрачно для  $g$ , а  $g$  прозрачно для  $E$ . Примерами прозрачных операций являются операции циклического сдвига, замены и т.п. Для разрушения прозрачности  $g$  используются преобразования (буфера), не прозрачные для  $g$ , которые размещаются между раундами шифрования. Все элементы системы подразделяются на легко сменяемые и долговременные, причем последние известны потенциальному злоумышленнику. К долговременным элементам относятся те элементы, которые относятся к разработке систем защиты и для изменения требуют вмешательства разработчиков или специалистов. К легко сменяемым элементам относятся элементы, которые предназначены для произвольного модифицирования, исходя из случайно выбираемых начальных параметров (ключ, пароль, идентификатор и т. п.). Надлежащий уровень секретности может быть обеспечен только по отношению к легко сменяемым элементам.

Согласно современным требованиям к криптосистемам они должны выдерживать криптоанализ на основе известного алгоритма, большого объема известного открытого текста и соответствующего ему шифротекста.

## 2. СТАНДАРТ БЛОЧНОЙ СИММЕТРИЧНОЙ КРИПТОСИСТЕМЫ AES

### 2.1. Математические предпосылки

Алгоритм AES оперирует байтами, которые рассматриваются как элементы расширенного конечного поля  $GF(2^8)$  по полиному  $m(x) = x^8 + x^4 + x^3 + x + 1$ . Выбранный полином  $m(x)$  нельзя представить как произведение других полиномов с двоичными коэффициентами. В результате любой полином  $a(x) \neq 0$  имеет инверсию, т.е. полином  $a^{-1}(x)$ , такой, что  $a(x) \cdot a^{-1}(x) \bmod m(x) = 1$ . На языке теории групп говорят, что в этом случае полиномы-байты образуют поле  $F_{2^8}$ .

Элементами поля  $GF(2^8)$  являются многочлены степени не более 7, которые могут быть заданы строкой своих коэффициентов. Если представить байт в виде

$$\{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}, a_i \in \{0, 1\}, i = \overline{0, 7}$$

то элемент поля описывается многочленом

$$a_7 x^7 + a_6 x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0.$$

*Пример.* Байту {11001011} (или {cb} в шестнадцатеричной форме) соответствует многочлен  $x^7 + x^6 + x^3 + x + 1$ .

*Пример.* Каждый байт данных рассматривается как полином с коэффициентами 0 и 1, действия с коэффициентами производятся по модулю 2:

$$\begin{aligned} 100010011 &= x^7 + x^4 + x + 1, \\ 10010011 + 01010001 &= 11000010. \end{aligned}$$

*Пример.* Полиномы-байты умножаются по модулю примитивного полинома  $m(x) = x^8 + x^4 + x^3 + x + 1$ .

$$10010011 \cdot 00000010 \bmod m(x) = 000111101.$$

Каждое слово данных может быть представлено как полином с коэффициентами в  $F_{2^8}$  в шестнадцатеричной системе счисления

*Пример.*

$$0x7500a302 = (0x75)x^3 + (0xa3)x + 2,$$

где префикс 0x означает запись числа в шестнадцатеричной системе

Раундовые преобразования *AES* оперируют 32-разрядными словами. Четырехбайтовому слову может быть поставлен в соответствие многочлен  $a(x)$  с коэффициентами из  $GF(2^8)$  степени не более трех

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0,$$

где  $a_i \in GF(2^8)$ ,  $i = \overline{0,3}$ .

Сложение двух многочленов с коэффициентами из  $GF(2^8)$  суть операция сложения многочленов с приведением подобных членов в поле  $GF(2^8)$ , т. е.

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0).$$

Таким образом, сложение двух 4-байтовых слов суть операция поразрядного *XOR* (суммирования по модулю 2).

Операция умножения полиномов выполняется по модулю  $(x^4 + 1)$ , для которого справедливо соотношение  $x^i \bmod (x^4 + 1) = x^{i \bmod 4}$ . Результат умножения  $d(x) = a(x) \otimes b(x) \bmod (x^4 + 1)$  в этом случае представляется 4-байтовым словом. В матричной форме это может быть записано следующим образом

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Умножению на  $x$  многочлена  $b(x)$  с коэффициентами из  $GF(2^8)$  по модулю  $x^4 + 1$ , учитывая свойства последнего, соответствует циклический сдвиг байтов в пределах слова в сторону старшего байта.

## 2.2. Алгебраическая структура шифра

### 1. Операции

Структура конечного поля. Поле  $F = GF(2^8)$  определяется примитивным полиномом

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

Определим  $a$  - корень полинома  $m(x)$ , т.е.  $m(a) = 0$  в поле  $F$ .  
Операции в матричном кольце определяются через матрицу

$$M_4(F) = \{Y = (\overset{\bullet}{b}_{ij})_{4 \times 4} \mid \overset{\bullet}{b}_{ij} \in F\}.$$

Элемент каждой матрицы  $M_4(F)$  принадлежит полю  $F$ . Каждый элемент матрицы состоит из 8 бит (образует байт). Каждая строка или каждый столбец может рассматриваться как слово из 32 бит (4 байт).

### 2. Базисные операторы

Преобразование подстановки  $S$  ( $S$ -блок) *SubBytes* в кольце  $M_4(F)$ . Для  $Y = (\overset{\bullet}{b}_{ij}) \in M_4(F)$  преобразование  $S$  представляет собой побайтно ориентированную операцию

$$S(Y) = (S(\overset{\bullet}{b}_{ij}))_{4 \times 4}.$$

Определим вектор-байт  $\overset{\bullet}{b} = (b_0, b_1, \dots, b_7) \in F$ , как элемент поля  $F$ . В полиномиальном представлении используем обозначение  $b(x) \bmod m(x)$ .

Построим аффинное преобразование  $T$

$$T(\overset{\mathbf{r}}{b}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

или

$$T(\overset{\bullet}{b}) = A\overset{\bullet}{b} + \overset{\mathbf{v}}{c},$$



где  $A$  – матрица размером  $(8 \times 8)$  циклических сдвигов вектора  $(1,0,0,0,1,1,1,1)$ ,  
 $\vec{c} = (1,1,0,0,0,1,1,0)$  – постоянный вектор.

В полиномиальном виде это преобразование  $T(b(x))$  запишется как

$$T(b(x)) = b(x) \leftarrow (x^7 + x^6 + x^5 + x^4 + 1)b(x) + (x^7 + x^6 + x^2 + x) \bmod (x^8 + 1).$$

Обратное преобразование  $T^{-1}$  выполняется с применением обратной матрицы  $A^{-1}$  и в матричном виде имеет вид

$$T^{-1}(\vec{b}) = A^{-1}(\vec{r} - \vec{b}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \left( \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \right)$$

или в полиномиальной форме

$$T^{-1}(b(x)) = b(x) \leftarrow (b(x) - (x^7 + x^6 + x^2 + x))(x^7 + x^5 + x^2) \bmod (x^8 + 1),$$

где  $(x^7 + x^5 + x^2) = (x^7 + x^6 + x^5 + x^4 + 1)^{-1} \bmod (x^8 + 1)$ .

Преобразование  $SubBytes(Y)$  действует независимо на каждый байт  $b$  в  $Y$  и определяется как

$$Sb(Y) = T \circ \vec{b}^{-1},$$

в полиномиальном виде

$$b(x) \leftarrow b^{-1}(x) \bmod m(x),$$

$$b(x) \leftarrow T(b(x)).$$

Обратное преобразование  $SubBytes^{-1}(X)$  действует на каждый байт  $b$  выходного блока  $X$  и записывается как

$$Sb^{-1}(X) = (T^{-1} \circ \vec{r}(\vec{c} - \vec{b}))^{-1}$$

или с помощью полиномов

$$b(x) \leftarrow T^{-1}(b(x)),$$

$$b(x) \leftarrow b^{-1}(x) \bmod m(x)$$

Результаты прямого и обратного преобразований, заранее подсчитанные для каждого байта от 0 до 255, заносятся в соответствующие таблицы (S-боксы).

*Обоснование выбора структуры.* Функция  $SubBytes$  состоит в обеспечении требуемого уровня перемешивания. Критерии разработки таблиц замен  $S$  – боксов обусловлены учетом возможностей дифференциального и линейного крип-

тоанализа, а также возможных алгебраических манипуляций (атака методом интерполяции). Основные критерии :

- обратимость;
- минимизация наидлиннейшей нетривиальной корреляции между линейными комбинациями бит входных и выходных данных (защита от линейного криптоанализа);
- минимизация наибольшего нетривиального значения  $XOR$  таблицы, что характеризует устойчивость к дифференциальному криптоанализу;
- сложность алгебраического представления в поле  $GF(2^8)$  для обеспечения устойчивости к алгебраическим атакам;
- простота описания.

Для формирования таблиц замен выбрано отображение  $x \rightarrow x^{-1}$  (мультипликативная инверсия) в поле  $GF(2^8)$ . Для защиты от алгебраической атаки применяется дополнительное изменение, которое может быть представлено как перемножение многочленов по модулю  $(x^8 + 1)$  с последующим сложением. Причем многочлен  $(x^6 + x^5 + x + 1)$  выбран таким образом, чтобы таблица замен не имела точек симметрии ( $S(a) = a$ ) и точек обратной симметрии ( $S(a) = a^{-1}$ ).

*Преобразование сдвига строк ShiftRows( $Y$ )*. Последние 3 строки состояния циклически сдвигаются влево на различное число байтов. Строка 1 сдвигается на  $C_1$  байт, строка 2 – на  $C_2$  байт, и строка 3 – на  $C_3$  байт. Значения сдвигов  $C_1$ ,  $C_2$ , и  $C_3$  зависят от длины блока  $N_b$ . Их величины приведены в таблице.

Величины сдвигов для разных длин блоков

$N_b$	$C_1$	$C_2$	$C_3$
4	1	2	3
6	1	2	3
8	1	3	4

В стандарте  $AES$ , где определен единственный размер блока, равный 128 битам,  $C_1 = 1$ ,  $C_2 = 2$  и  $C_3 = 3$ . Сдвиг действует на каждую строку  $r_i$  в  $Y$ , т. е. на  $i$ -ю ( $i = 0, 1, 2, 3$ ) последовательность байт слов блока, по правилу

$$r_i \leftarrow r_i \ll i,$$

где  $\ll$  – операция циклического сдвига влево на указанное число байт.

Если представить блок  $Y$  как

$$Y = \begin{bmatrix} y_{00} & y_{01} & y_{02} & y_{03} \\ y_{10} & y_{11} & y_{12} & y_{13} \\ y_{20} & y_{21} & y_{22} & y_{23} \\ y_{30} & y_{31} & y_{32} & y_{33} \end{bmatrix},$$

где  $y_{ij}$  – байт блока, то операцию  $R$  сдвига при шифровании можно записать как

$$R(Y) = \begin{bmatrix} y_{00} & y_{01} & y_{02} & y_{03} \\ y_{11} & y_{12} & y_{13} & y_{10} \\ y_{22} & y_{23} & y_{20} & y_{21} \\ y_{33} & y_{30} & y_{31} & y_{32} \end{bmatrix}.$$

Обратная операция над блоком  $X$  имеет вид

$$R^{-1}(X) = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{13} & x_{10} & x_{11} & x_{12} \\ x_{22} & x_{23} & x_{20} & x_{21} \\ x_{31} & x_{32} & x_{33} & x_{30} \end{bmatrix}.$$

Преобразование  $ShiftRow^{-1}(X)$  или  $R^{-1}(X)$  действует на каждую строку строку  $r_i$  в  $X$  по правилу

$$r_i \leftarrow r_i \gg i,$$

где  $\gg$  - операция циклического сдвига вправо на указанное число байт.

*Обоснование выбора структуры.* Выбор из возможных комбинаций смещений в строках преобразуемого блока был сделан на основании следующих критериев:

- 1) все четыре смещения должны быть различны для каждой строки и  $C_0 = 0$ ;
- 2) стойкость к атакам, использующим сокращенные дифференциалы;
- 3) стойкость к атаке «Квадрат»;
- 4) простота.

Операция сдвига последних трех строк состояния обозначена как  $ShiftRows(State)$ .

2.3. Преобразование  $MixColumns(Y)$  действует на каждый столбец  $c_i$  в блоке  $Y$ , т. е. на каждое машинное слово,  $i = 0, 1, 2, 3$ , по правилу

$$c_i(x) \leftarrow a(x)c_i(x) \bmod(x^4 + 1),$$

где

$$a(x) = 3x^3 + x^2 + x + 2.$$

Как линейное преобразование  $L$  в поле  $F$  преобразование  $MixColumns(Y)$  запишется следующим образом

$$L(Y) = L \cdot Y,$$

где

$$L = \begin{bmatrix} a & 1+a & 1 & 1 \\ 1 & a & 1+a & 1 \\ 1 & 1 & a & 1+a \\ 1+a & 1 & 1 & a \end{bmatrix}.$$

Обратное преобразование  $L^{-1}$  имеет вид

$$L^{-1} = \begin{bmatrix} b_0 & b_3 & b_2 & b_1 \\ b_1 & b_0 & b_3 & b_2 \\ b_2 & b_1 & b_0 & b_3 \\ b_3 & b_2 & b_1 & b_0 \end{bmatrix},$$

где  $b_0 = a^3 + a^2 + a$ ,  $b_1 = a^3 + 1$ ,  $b_2 = a^3 + a^2 + 1$ ,  $b_3 = a^3 + a + 1$ .

Преобразование  $MixColumns^{-1}(X)$  действует на каждый столбец  $c_i$  в блоке  $X$  по правилу

$$c_i(x) \leftarrow a^{-1}(x)c_i(x) \bmod(x^4 + 1),$$

где  $a^{-1}(x) = 11x^3 + 13x^2 + 9x + 14$ .

*Обоснование выбора структуры.* Предварительно определим критерий распространения линейного преобразования

Пусть  $F$  будет линейным преобразованием, выполняемым над вектором байт, и пусть байтовым весом будет количество ненулевых байт в нем.

Коэффициентом распространения обратимого линейного преобразования, характеризующим его силу криптографического рассеивания назовем величину

$$K(F) = \min_{\mathbf{a} \neq 0} (wt(\mathbf{a}) + wt(F(\mathbf{a}))),$$

где  $wt(\mathbf{a})$  – вес Хэмминга вектора  $\mathbf{a}$ , т.е. количество ненулевых компонент вектора.

Байт не равный нулю будем называть активным байтом.

Для функции *MixColumns* при одном активном байте входного 32-разрядного массива *State* на выходе будет максимум 4 активных байта, так как эта функция преобразует столбцы по отдельности. Таким образом, верхняя граница коэффициента распространения равна 5. Это говорит о том, что разница во входных данных в 1 байт рассеивается (распространяется) на 4 байта выходных данных. Разница в 2 байта входных данных отразится как минимум, на 3 байтах выходных данных. Можно сказать, что линейная зависимость между битами во входных и выходных данных всегда затрагивает биты как минимум пяти различных байт на входе и выходе функции *MixColumns*.

Функция рассеивания *MixColumns* была выбрана из возможных линейных преобразований  $4 \text{ bytes} \rightarrow 4 \text{ bytes}$  исходя из следующих критериев выбора:

- обратимость;
- линейность в поле  $GF(2^8)$ ;
- достаточно сильное рассеивание;

- симметричность, т. е. функция должна работать со всеми данными единообразно;
- скорость выполнения на 8-разрядных процессорах;
- простота описания.

3. *Алгоритм формирования ключа.* Формирование ключа проходит в пословном режиме. Раундовые ключи получаются из секретного ключа шифрования  $\mathbf{K}_c$  посредством алгоритма выработки ключей. Он содержит два компонента: *расширение ключа* (Key Expansion) и *выбор раундового ключа* (Round Key Selection). Основополагающие принципы алгоритма выглядят следующим образом:

- общее число раундовых ключей равно длине блока, умноженной на число раундов, плюс 1 (например, для длины блока 128 бит и 10 раундов требуется 1408 раундовых ключей);
- ключ шифрования расширяется в *расширенный ключ* (Expanded Key);
- раундовые ключи берутся из расширенного ключа следующим образом: первый раундовый ключ содержит первые  $N_b$  слов, второй - следующие  $N_b$  слов и т. д.

3.1. *Расширение ключа (Key Expansion).* Расширенный ключ представляет собой линейный массив  $\{K[i] = (k_{i,0}, k_{i,1}, k_{i,2}, k_{i,3}) \mid i = \overline{0, N_b(N_r + 1)}\}$  из  $N_b(N_r + 1)$  4-байтовых слов. В AES-128 массив  $\{K[i] \mid i = \overline{0, 4(N_r + 1)}\}$  состоит из  $4(N_r + 1)$  4-байтовых слов.

**Алгоритм.** Формирование рабочего ключа

ВХОД: Секретный ключ  $\mathbf{K}_c$  из  $N_k$  слов.

ВЫХОД: Расширенный ключ  $\mathbf{K} = \{K[i] \mid i = \overline{0, N_b(N_r + 1)}\}$  из  $N_b(N_r + 1)$  4-байтовых слов  $K[i]$ .

1.  $\mathbf{K} \leftarrow \mathbf{K}_c$  из  $N_k$  слов;
2. FOR  $i = N_k, N_k + 1, \dots, N_b(N_r + 1) - 1$  DO
3.    $T \leftarrow K[i - 1]$ ;
4.   IF  $i \bmod N_k = 0$  THEN
5.      $T \leftarrow \text{SubWord}(\text{RotWord}(T)) \oplus \text{Rcon}(T) \ (i \div N_k)$ ;
6.   ELSE IF  $N_k = 8$  AND  $i \bmod N_k = 4$  THEN
7.      $T \leftarrow \text{SubWord}(T)$ ;
8.    $K[i] = K[i - N_k] \oplus T$ ;
9.   RETURN  $K[0], \dots, K[N_b(N_r + 1)]$ .

В данном алгоритме *SubWord* – функция, применяющая *S*-бокс шифра к каждому байту слова  $T = [t_0, t_1, t_2, t_3]$

$$SubWord(T) \rightarrow [S[t_1], S[t_2], S[t_3], S[t_0]].$$

Преобразование  $RotWord(T)$  осуществляет циклический сдвиг слова  $T$  на один байт влево

$$RotWord(T) \rightarrow [t_1, t_2, t_3, t_0].$$

Массив раундовых констант  $Rcon$  состоит из слов  $Rcon[i] = [z_i, 0, 0, 0]$ , где  $z_i = a^{i-1} \bmod m(x)$ .

*Пример.* Рассмотрим формирование ключа для 128-битной версии из 10 раундов. В этом случае требуется раундовый ключ из 44 слов или 1408 бит. Пусть  $\{K[i], i = 0, 1, \dots, 43\}$  последовательность слов,  $K[i] \in F_4$  и состоит из 4 байт

$$K[i] = (k_{i,0}, k_{i,1}, k_{i,2}, k_{i,3}).$$

Матрица ключей  $\mathbf{K} = [K_{i,j} = K[4i + j]]$ ,  $0 \leq i \leq 10$ ,  $0 \leq j \leq 3$  имеет размерность  $(11 \times 4)$ .

Формирование ключа происходит следующим образом.

*Начальное состояние.* Секретный ключ из 128 бит обозначим как  $K_c = \{K[0], K[1], K[2], K[3]\}$  и примем за начальное состояние.

*Шаги расширения:*

Для  $i = 1, 2, \dots, 10$  вычисляются

$$K[4i] = K[4(i-1)] + (S(k_{4i-1,3}), S(k_{4i-1,0}), S(k_{4i-1,1}), S(k_{4i-1,2})) + (a^i, 0, 0, 0);$$

$$K[4i + j] = K[4i + j - 1] + K[4(i-1) + j], \in F_4, j = 1, 2, 3.$$

Так для  $i = 1$  получаем

$$K[4] = K[0] + (S(k_{3,3}), S(k_{3,0}), S(k_{3,1}), S(k_{3,2})) + (a, 0, 0, 0), \quad K[5] = K[4] + K[1], \\ K[6] = K[5] + K[2], \quad K[7] = K[6] + K[3].$$

Первая вектор-строка матрицы ключей равна  $\mathbf{K}_1 = (K[4], K[5], K[6], K[7])$ .

Массив раундовых констант представляет собой элементы поля  $GF(2^8)$  и для  $m(x) = x^8 + x^4 + x^3 + x + 1$  имеет вид  $z_1 = 1 \rightarrow 00000001$ ,  $z_2 = a \rightarrow 00000010$ ,

$$z_3 = a^2 \rightarrow 00000100, \quad z_4 = a^3 \rightarrow 00001000, \quad z_5 = a^4 \rightarrow 00010000,$$

$$z_6 = a^5 \rightarrow 00100000, \quad z_7 = a^6 \rightarrow 01000000, \quad z_8 = a^7 \rightarrow 10000000,$$

$$z_9 = a^4 + a^3 + a + 1 \rightarrow 00011011, \quad z_{10} = a^5 + a^4 + a^2 + a \rightarrow 00110110.$$

*Пример.* Сформируем полный ключ для 10 раундового шифрования. Воспользуемся оператором *randKeyGenerator*, который формирует секретный ключ случайным образом  $\mathbf{K}_c = randKeyGenerator()$ . Получим

$$\mathbf{K}_c := ["11100100", "01000011", "10001010", "11010001", "10011100", "01000100", \\ "11000101", "10110111", "00111011", "11010010", "11000101", "10000000", \\ "00011110", "01001000", "10000101", "01101101"].$$

Оператор *keyExpander* реализует алгоритм формирования расширенного ключа. Применяя его к выбранному секретному ключу, получим матрицу, состоящую из 4 строк и 44 столбцов, элементы которой имеют представление десятичных чисел:

$KeyExpander(K_s) := matrix([ [228, 156, 59, 30, 183, 43, 16, 14, 210, 249, 233, 231, 92, 165, 76, 171, 45, 136, 196, 111, 246, 126, 186, 213, 11, 117, 207, 26, 121, 12, 195, 217, 34, 46, 237, 52, 212, 250, 23, 35, 75, 177, 166, 133], [67, 68, 210, 72, 212, 144, 66, 10, 23, 135, 197, 207, 34, 165, 96, 175, 51, 150, 246, 89, 240, 102, 144, 201, 158, 248, 104, 161, 174, 86, 62, 159, 164, 242, 204, 83, 218, 40, 228, 183, 124, 84, 176, 7], [138, 197, 197, 133, 182, 115, 182, 51, 47, 92, 234, 217, 140, 208, 58, 227, 58, 234, 208, 51, 76, 166, 118, 69, 157, 59, 77, 8, 221, 230, 171, 163, 100, 130, 41, 138, 228, 102, 79, 197, 223, 185, 246, 51], [209, 183, 128, 109, 163, 20, 148, 249, 8, 28, 136, 113, 156, 128, 8, 121, 254, 126, 118, 15, 86, 40, 94, 81, 85, 125, 35, 114, 247, 138, 169, 219, 194, 72, 225, 58, 218, 146, 115, 73, 252, 110, 29, 84] ]).$

*Обоснование выбора алгоритма.* Метод построения рабочего ключа должен способствовать решению следующих задач:

- 1) затруднить атаки на шифр при частично известном секретном ключе или при использовании зависимых ключей;
- 2) устранить имеющиеся симметрии внутри раунда шифра и между раундами (использование массива раундовых констант).

Чтобы получить рабочий ключ для расшифрования, необходимо подвергнуть блоки рабочего ключа с первого по предпоследний блок преобразованию  $MixColumns^{-1}$ .

4. *Алгоритмы шифрования и расшифрования.* Шифр начинается и заканчивается сложением с ключом. Это позволяет закрыть вход первого раунда при атаке по известному тексту и сделать криптографически значимым результат последнего раунда

*Алгоритм. Шифрование блока*

ВХОД: Блок  $X$ , рабочий ключ  $K$

ВЫХОД: Блок  $Y$ .

1.  $Y \leftarrow X \oplus K_0$ ; *AddRoundKey*
2. FOR  $i = 1, 2, \dots, r-1$  DO
3.      $Y \leftarrow SubBytes(Y)$ ,
4.      $Y \leftarrow ShiftRows(Y)$ ,
5.      $Y \leftarrow MixColumns(Y)$ ,
6.      $Y \leftarrow Y \oplus K_i$ ; *AddRoundKey*; END DO;
7.      $Y \leftarrow SubBytes(Y)$ ,
8.      $Y \leftarrow ShiftRows(Y)$ ,
9.      $Y \leftarrow Y \oplus K_r$ ; *AddRoundKey*;
10. RETURN  $Y$ .

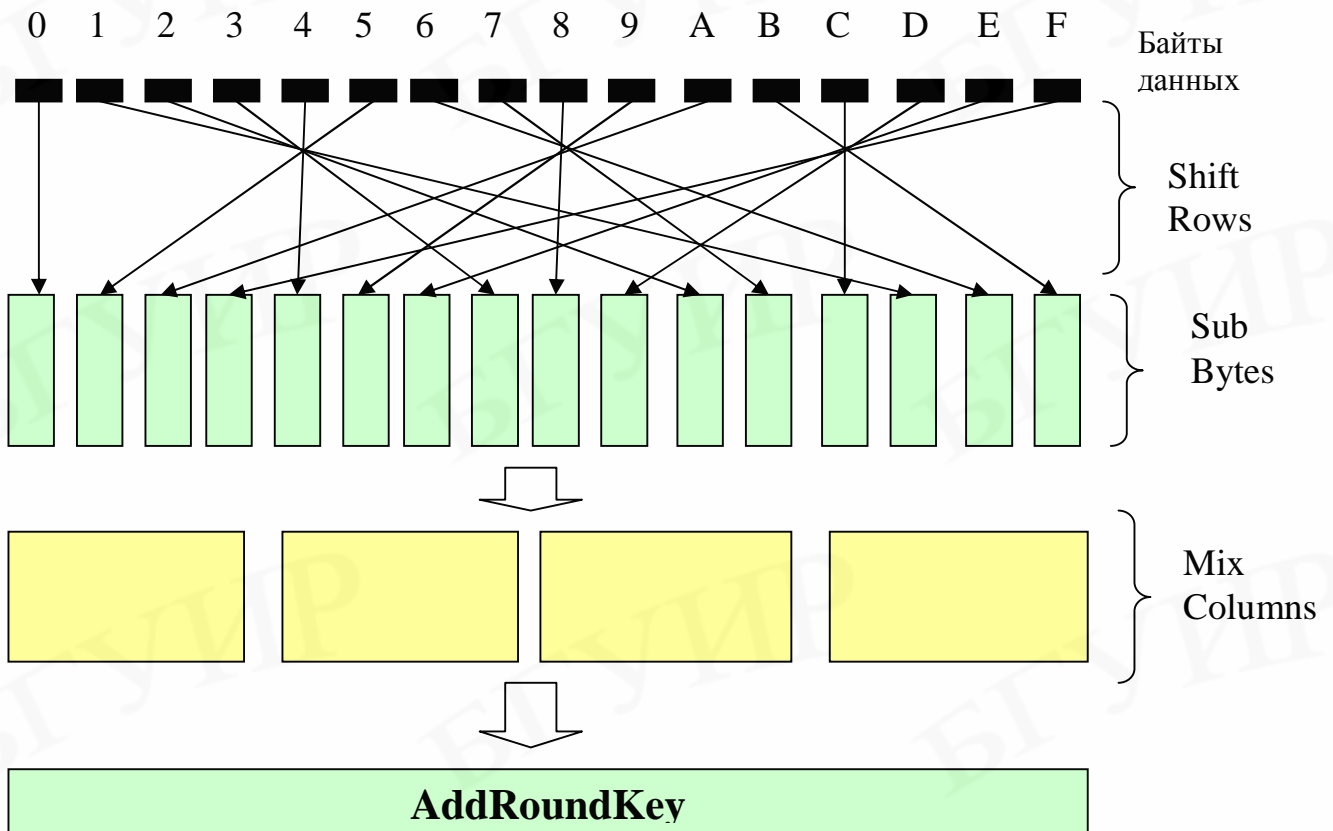


Рис. 2.1. Структура раунда

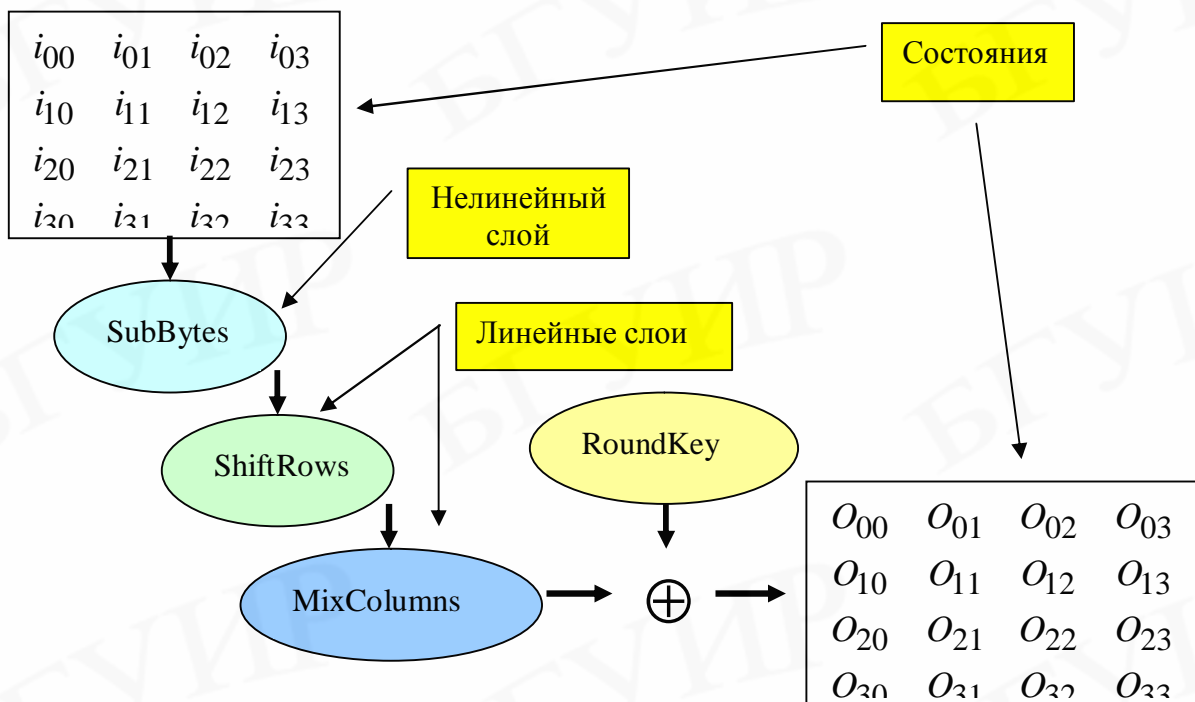


Рис 2.2. Классификация базовых операторов



Алгоритм. Расшифрование блока

ВХОД: Блок  $Y$ , рабочий ключ  $K$

ВЫХОД: Блок  $X$ .

1.  $Y \leftarrow X \oplus K_r$ ; *AddRoundKey*
2. FOR  $i = r - 1, r - 2, \dots, 1$  DO
3.      $X \leftarrow \text{SubBytes}^{-1}(X)$ ,
4.      $X \leftarrow \text{ShiftRows}^{-1}(X)$ ,
5.      $X \leftarrow \text{MixColumns}^{-1}(Y)$ ,
6.      $X \leftarrow X \oplus K_i$ ; *AddRoundKey*; END DO;
7.      $X \leftarrow \text{SubBytes}^{-1}(Y)$ ,
8.      $X \leftarrow \text{ShiftRows}^{-1}(Y)$ ,
9.      $X \leftarrow X \oplus K_0$ ; *AddRoundKey*;
10.    RETURN  $X$ .

Рис. 2.1 и 2.2 иллюстрируют процесс криптопреобразований.

### 3. СТОЙКОСТЬ К АТАКАМ КРИПТОАНАЛИЗА

#### 3.1. Дифференциальный криптоанализ

Дифференциальный криптоанализ (ДК) – это атака на шифр, базирующаяся на возможности свободного подбора открытых текстов для последующего их зашифрования. Анализуются зависимости между парами блоков данных до и после применения шифра. ДК-атаки становятся возможны тогда, когда можно установить проникновение таких зависимостей сквозь почти все (за исключением 2-3) раунды криптопреобразования.

Коэффициент проникновения определяется как относительное количество фиксированных пар бит данных, для которых можно установить зависимость различий на выходе от различий на входе при зашифровании.

Если при криптоанализе обнаруживается, что в выходных данных существует пара бит такая, что различия между этими битами можно предсказать, исходя из различий между битами выбранной пары во входных данных, и это событие не случайно (т.е. его вероятность больше чем  $1/2^{n-1}$ , где  $n$  – длина входных данных в битах), то можно считать, что криптопреобразование имеет явную криптографическую слабость.

Раунд зашифрования при ДК удобно рассматривать как отдельную разностную сессию. Полная разностная сессия состоит из отдельных сессий – раундов шифрования, а полный коэффициент проникновения является произведением коэффициентов составляющих сессий.

Для шифра AES уже после четырех раундов коэффициент проникновения становится не больше, чем  $2^{-150}$ , а после 8 раундов –  $2^{-300}$ , что доказывает стойкость шифра.

### 3.2. Линейный криптоанализ

Линейный криптоанализ (ЛК) основан на выявлении линейных зависимостей между битами входных данных, которые обуславливают вполне определенные зависимости между битами выходных данных.

Пусть

$$A[i_1, i_2, \dots, i_a]$$

сумма по модулю 2 бит массива данных  $A$ , индексами которых являются значения  $i_1, i_2, \dots, i_a$ , то есть

$$A[i_1, i_2, \dots, i_a] = A[i_1] \oplus A[i_2] \oplus \dots \oplus A[i_a].$$

Тогда линейной зависимостью будем называть выражение вида

$$P[i_1, i_2, \dots, i_a] = C[j_1, j_2, \dots, j_b] \oplus K[k_1, k_2, \dots, k_c],$$

где  $P, C$  и  $K$  означают, соответственно, массивы бит входных данных, выходных данных и ключа.

Если для достаточно большого количества различных  $P$  и  $C$  удастся подобрать  $K$ , удовлетворяющее данному выражению, то можно говорить о раскрытии одного бита информации о ключе на основании существования линейной зависимости (иначе – *корреляции*) входных и выходных данных.

ЛК-атаки становятся эффективными, когда можно установить существование таких корреляций после почти всех раундов шифрования. При этом относительное число корреляций в данных должно быть значительно выше, чем  $2^{-n/2}$ . Достаточным условием стойкости к ЛК-атакам является отсутствие каких-либо линейных сессий с коэффициентом общей корреляции выше, чем  $2^{-n/2}$ .

### 3.3. Метод проникновения образов активности

#### *Активные S-боксы*

Для ДК определим количество активных  $S$ -боксов в одном раунде как количество ненулевых байт, задающих различия во входных данных раунда (количество ненулевых байт после применения операции XOR к двум массивам входных данных).

Образ массива  $State$ , определяющий позиции активных  $S$ -боксов, обозначается термином разностный образ активности. Разностный байтовый вес обозначает число активных байт в образе.

Для ЛК количество активных  $S$ -боксов определяется количеством ненулевых байт в массиве входных данных. Образ, определяющий позиции активных  $S$ -боксов, обозначается термином корреляционный образ активности. Корреляционный байтовый вес  $W(a)$  определяется количеством ненулевых байт в образе  $a$ . Обозначим количество активных столбцов в  $a$  как  $W_c(a)$ . Байтовый вес столбца  $j$  образа  $a$  обозначим как  $W_j(a)$ . Общий вес сессии равен сумме весов образов активности входных данных для каждого раунда, входящего в состав сессии.

Можно рассматривать проникновение разностных (линейных) образов активности сквозь раунды преобразования как проникновение зависимостей входных/выходных данных в разностной (линейной) сессии.

Функции, входящие в состав одного раунда преобразования, оказывают следующее влияние на образы активности.

*SubBytes* и *AddRoundKey*. Образы активности, байтовый столбцовый вес не меняется.

*ShiftRows*. Байтовый вес не изменяется. Столбцовый вес выходных данных ограничен снизу максимальным байтовым весом столбца входных данных. Столбцовый вес входных данных ограничен снизу максимальным байтовым весом столбца из выходных данных.

*MixColumns*. Столбцовый вес не меняется. Функция имеет коэффициент распространения, равный 5. Это означает, что для любого активного столбца из образа входных (или выходных) данных сумма байтовых весов на входе и выходе раунда будет ограничена снизу значением 5.

**Свойство 1.** Общий байтовый вес двухраундовой сессии с  $Q$  активными столбцами на входе второго раунда ограничен снизу значением  $5Q$ .

**Свойство 2.** В двух раундовой сессии сумма активных столбцов входных данных и активных столбцов в выходных данных не меньше 5.

**Свойство 3.** Любая сессия, состоящая из 4 раундов, имеет как минимум 25 активных байт.

На рис. 3.1, 3.2 и 3.3 показаны примеры проникновения образов активности.

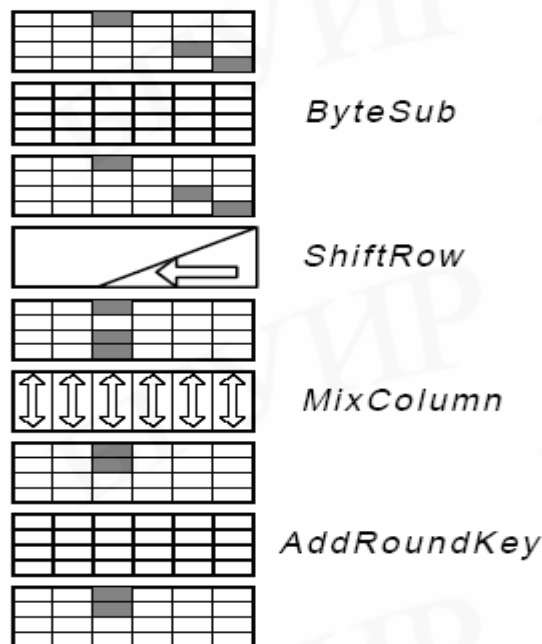


Рис. 3.1. Проникновение образов активности сквозь однораундовое преобразование

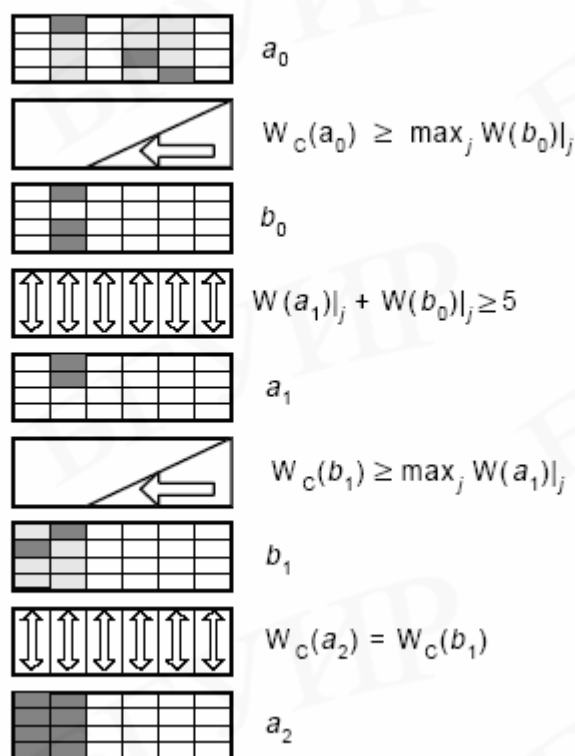


Рис. 3.2. Иллюстрация свойства 2 с одним активным столбцом в образе

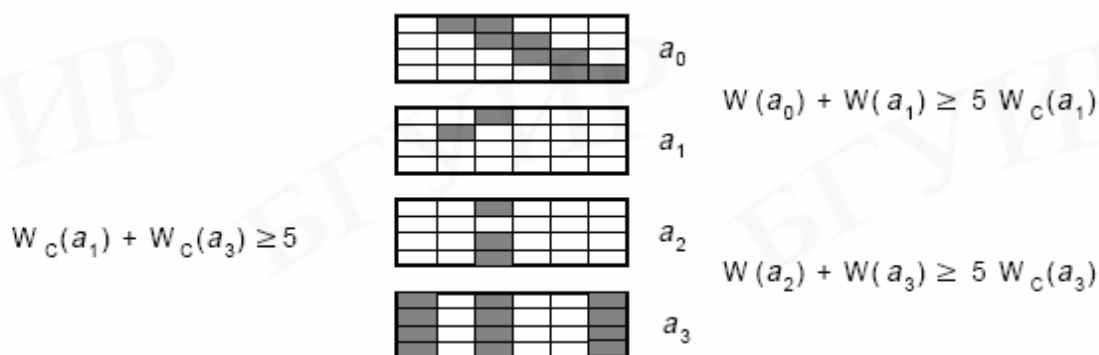


Рис. 3.3. Иллюстрация свойства 3

### 3.4. Атака «Квадрат»

Атака использует при своем проведении байт-ориентированную структуру шифра и основана на свободном подборе атакующим некоторого набора открытых текстов для последующего их зашифрования. Атака считается эффективной, если она дает результат быстрее, чем полный перебор всех возможных значений ключа.

Пусть  $\Lambda$ -набор – такой набор из 256 входных блоков (массивов *State*), каждый из которых имеет байты (активные), значения которых различны для всех 256 блоков. Остальные байты (пассивные), остаются одинаковыми для всех 256 блоков из  $\Lambda$ -набора. Будучи подвергнуты обработке функциями *SubBytes* и *Ad-*

*dRoundKey* блоки  $\Lambda$ -набора дадут в результате другой  $\Lambda$ -набор с активными байтами в тех же позициях, что и у исходного. Функция *ShifrRows* сместит эти байты соответственно заданным в ней смещениям в строках массива *State*. После функции *MixColumns*  $\Lambda$ -набор в общем случае не обязательно останется  $\Lambda$ -набором. Но поскольку каждый байт результата функции *MixColumns* является линейной комбинацией четырех входных байт того же столбца

$$b_{ij} = 2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j},$$

столбец с единственным активным байтом на входе даст в результате на выходе столбец со всеми четырьмя байтами – активными.

Рассмотрим  $\Lambda$ -набор, во всех блоках которого активен только один байт. Значение этого байта различно во всех 256 блоков, а остальные байты одинаковы. Проследим действие операторов шифра на такой  $\Lambda$ -набор.

В первом раунде функция *MixColumns* преобразует один активный байт в столбец из 4 активных байтов. Во втором раунде эти 4 байта разойдутся по 4 различным столбцам в результате преобразования функцией *ShifrRows*. Функция же *MixColumns* следующего третьего раунда преобразует эти байты в 4 столбца, содержащие активные байты. Этот набор все еще остается  $\Lambda$ -набором до того самого момента, когда он поступает на вход функции *MixColumns* третьего раунда.

Основное свойство  $\Lambda$ -набора состоит в том, что поразрядная сумма по модулю два всех блоков такого набора всегда равна нулю. Таким образом, все данные на входе четвертого раунда сбалансированы (т.е. их полная сумма равна нулю). Этот баланс в общем случае нарушается последующим преобразованием данных функцией *SubBytes*.

Если предположить, что четвертый раунд является последним, то есть в нем нет функции *MixColumns*, тогда каждый байт выходных данных этого раунда зависит только от одного байта входных данных. Если обозначить через  $a$  байт выходных данных четвертого раунда, через  $b$  байт входных данных и через  $k$  – соответствующий байт раундового ключа, то можно записать

$$a_{ij} = \text{SubBytes}(b_{ij}) \oplus k_{ij}.$$

Отсюда, предполагая значение  $k_{ij}$ , можно по известному  $a_{ij}$  вычислить  $b_{ij}$ , а затем проверить правильность догадки о значении  $k_{ij}$ : если значения байта  $b_{ij}$ , полученные при данном  $k_{ij}$  не будут сбалансированы по всем блокам (то есть не дадут при поразрядном суммировании нулевой результат), значит догадка неверна. Перебрав максимум  $2^8$  вариантов байт раундового ключа, можно найти его истинное значение. По такому же принципу могут быть определены и другие байты раундового ключа. А по значению полного раундового ключа, при известном алгоритме его развертывания, не составляет труда восстановить сам начальный ключ.

Известны расширенные атаки типа «Квадрат» с добавлением пятого раунда в конец атаки и шестого раунда в начало атаки. Однако уже для 7 раундов атака «Квадрат» становится менее эффективна, чем полные перебор.

### 3.5. Спектральная оценка нелинейности криптографической функции

Линейность той или иной математической задачи предопределяет эффективность методов и алгоритмов ее решения. При разработке и исследовании криптографических систем «нелинейность» (как общее понятие) используемых преобразований информации является их фундаментальным и неотъемлемым качеством.

*Спектральные преобразования.* Определим поле  $GF(2^n)$ , состоящее из  $2^n$  элементов, примитивный элемент  $\alpha$  и отображение  $f: GF(2^n) \rightarrow GF(2)$ . Булева функция (БФ)  $f(x)$  является представлением следа бинарной последовательности  $\mathbf{a} = \{a_i\}_{i=0}^{2^n-1}$ , если  $a_i = f(\alpha^i)$  для всех  $i$ . Спектральное преобразование Уолша – Адамара (ПУ – А) функции  $f: GF(2^n) \rightarrow GF(2)$  определяется через выражение

$$\hat{f}(I) = \sum_{x \in GF(2^n)} (-1)^{\Psi(Ix) + f(x)}, I \in GF(2^n).$$

Если функция  $\Psi(Ix)$  представляет собой след бинарной последовательности  $Tr(Ix)$ , спектральное преобразование носит название Адамара. В том случае, если  $\Psi(Ix) = \langle \lambda, x \rangle$ , функцию  $\hat{f}(I)$  называют преобразованием Уолша [2 - 4].

*Оценка нелинейности булевых функций.* Нелинейность  $N_f$  БФ  $f$  определяется через расстояние Хэмминга  $d(f, a)$  как  $N_f = \min\{d(f, a), a - \text{аффинная функция}\}$ , где  $d(f, a)$  – число  $x$  для которых  $f(x) \neq a(x)$ .

Уровень нелинейности может быть оценен через ПУ – А как

$$N_f = 2^{n-1} - 1/2 \max_I |\hat{f}(I)|.$$

Булева функция удовлетворяет условию корреляционной иммунности порядка  $k$  если  $\hat{f}(w) = 0$ , для все весов Хэмминга  $1 \leq wt(w) \leq k$ . Если функция  $f$  является к тому же балансной, то говорят, что она является  $k$ -резилентной функцией [4].

*Пример.* Оценим уровень нелинейности выходной функции шифра AES. В качестве открытого текста выберем предложение, состоящее из 16 знаков

mess := "Have a nice day."

Сообщению соответствует матрица массива *State*

$$messMatrix := \begin{bmatrix} "01001000" & "00100000" & "01101001" & "01100100" \\ "01100001" & "01100001" & "01100011" & "01100001" \\ "01110110" & "00100000" & "01100101" & "01111001" \\ "01100101" & "01101110" & "00100000" & "00101110" \end{bmatrix}$$

С помощью оператора *randKeyGenerator* сформируем начальный, секретный ключ

testKey := ["11001000", "01011000", "11101001", "11001011", "10110100",  
 "00001110", "01010110", "01101010", "00100110", "10101111", "00000000",  
 "11000101", "00111010", "01100001", "01010011", "01001110"] = [200, 88, 233,  
 203, 180, 14, 86, 106, 38, 175, 0, 197, 58, 97, 83, 78].

Используя оператор *encryptAESascii(mess, testKey)*, проведем зашифрование сообщения. Шифрограмма «cipherText» имеет вид

cipherText := "A891B7EDCA05A28E1518ECCFDE8DE03D".

Шифрограмме соответствует числовая и битовая последовательности

z := [10, 8, 9, 1, 11, 7, 14, 13, 12, 10, 0, 5, 10, 2, 8, 14, 1, 5, 1, 8, 14, 12, 12, 15,  
 13, 14, 8, 13, 14, 0, 3, 13] = [ 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,  
 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,  
 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,  
 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,  
 1, 1, 0, 1].

Проведя в битовой последовательности подстановку  $0 \rightarrow 1, 1 \rightarrow (-1)$  и выполнив преобразование Уолша-Адамара (оператор *Walsh\_N*) получим спектр, показанный на рис.3.4.

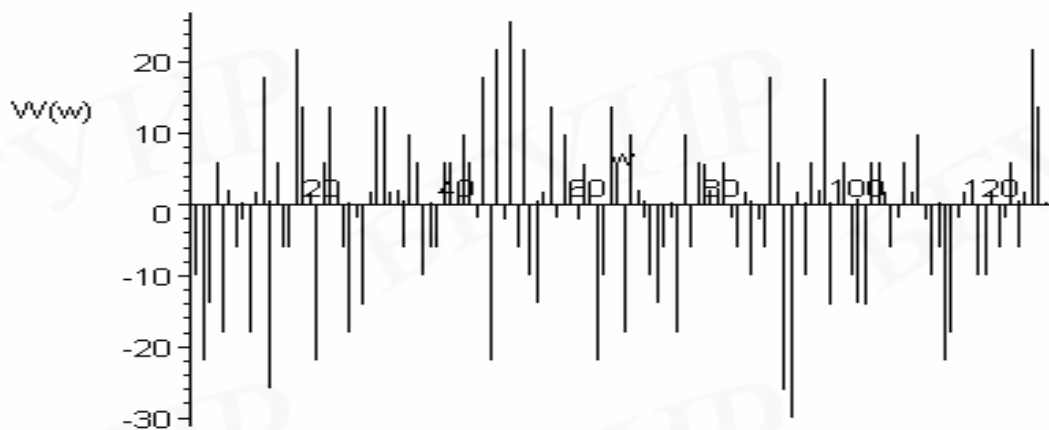


Рис. 3.6. Спектр П У-А шифрограммы

Максимальное абсолютное значение спектральных коэффициентов не превышает 30, что характеризует уровень нелинейности исследуемой дискретной функции.

## 4. МОДЕЛИРОВАНИЕ КРИПТОАЛГОРИТМОВ В СРЕДЕ MAPLE

### 4.1. Список основных операторов моделирования

1. Операторы преобразования формата данных:

- *intToBits* – целых чисел в двоичный код;
- *bitToList* – двоичного кода в последовательность бит:  
    > *bitToList(intToBits(18));*  
        [0, 0, 0, 1, 0, 0, 1, 0];
- *listToPoly* – последовательности бит в полином:  
    > *listToPoly([0, 0, 0, 1, 0, 0, 1, 0]);*  
         $a^4 + a$ ;
- *polyToInt* – преобразование полинома в целое число:  
    > *polyToInt(alpha^4+alpha);*  
        18;
- *hexTo8Bits* – 16-ичного кода в байт;
- *listToBits* – последовательности чисел в двоичный код;
- *bitToInt* – двоичный код в целое число;
- *listToInt* – последовательность бит в число;
- *polyToList* – полином в последовательность бит;
- *intToHex* – числа десятичной системы в 16-ичную;
- *polyToBits* – полином в биты;
- *bitToPoly* биты в полином;
- *intToPoly* – десятичное число в полином;
- *listToMatrix* – последовательность чисел в матрицу размером (4×4);
- *listToMatrix2* – последовательность чисел в транспонированную матрицу размером (4×4);
- *matrixToList* – матрицу в последовательность десятичных чисел;
- *matrixToHex* – матрицу в последовательность 16-ичных чисел.

2. Операторы суммирования по модулю два (XOR):

- *XOR* – операция суммирования по модулю два;
- *xorNbits* – операция суммирования последовательности из  $N$  бит;
- *xor8* – операция суммирования байт.

3. Базовые операторы:

- *SboxTable* –  $S$ -бокса;
- *InvSBoxTable* – инверсного  $S$ -бокса;
- *randKeyGenerator* – генератора случайного секретного ключа;
- *keyExpander* – алгоритма расширения ключа;



- *BS* – функции *SubBytes*;
- *InvBS* – инверсной функции *SubBytes*<sup>-1</sup>;
- *SR* – функции *ShiftPows*;
- *InvSR* – инверсной функции *ShiftPows*<sup>-1</sup>;
- *MC* – функции *MixColumns*;
- *InvMC* – инверсной функции *MixColumns*<sup>-1</sup> ;
- *ARK* – функции *AddRoundKey*

#### 4. Алгоритмы шифрования и расшифрования

- *EncryptAESascii* – шифрования данных в коде ASC II;
- *DecryptAESascii* – расшифрования данных в коде ASC II;
- *encryptAEShex* – шифрования данных в 16-системе представления чисел;
- *decryptAEShex* – расшифрования данных в 16-системе представления чисел.

### 4.2. Моделирование работы алгоритма шифрования

Разработчики шифра выбрали в качестве теста начальные условия следующего вида: данных и ключ шифрования имели вид

"000102030405060708090A0B0C0D0E0F".

Правильно работающий шифр должен давать следующие состояния:

PT = 000102030405060708090A0B0C0D0E0F  
 CT1 = B5C9179EB1CC1199B9C51B92B5C8159D  
 CT2 = 2B65F6374C427C5B2FE3A9256896755B  
 CT3 = D1015FCBB4EF65679688462076B9D6AD  
 CT4 = 8E17064A2A35A183729FE59FF3A591F1  
 CT5 = D7557DD55999DB3259E2183D558DCDD2  
 CT6 = 73A96A5D7799A5F3111D2B63684B1F7F  
 CT7 = 1B6B853069EEFC749AFEFD7B57A04CD1  
 CT8 = 107EEADFB6F77933B5457A6F08F046B2  
 CT9 = 8EC166481A677AA96A14FF6ECE88C010  
 CT = 0A940BB5416EF045F1C39458C653EA5A.

Составим программу следующего вида

```
> testString := "000102030405060708090A0B0C0D0E0F":
testHexList := [seq(substring(testString, 2*i-1..2*i), i=1..16)];
```

```
testHexList := ["00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "0A", "0B",
"0C", "0D", "0E", "0F"]
```

```
> testString2 := "00000000000000000000000000000001":
> testHexList2 := [seq(substring(testString2, 2*i-1..2*i), i=1..16)];
```

```
testHexList2 := ["00", "00", "00", "00", "00", "00", "00", "00", "00", "00", "00",
"00", "00", "00", "00", "01"]
```

```
> testByteList := map(hexTo8Bits, testHexList2):
```

```
testKey := testByteList;
```

```
messMatrix := listToMatrix2(testByteList);
```

```
testKey := ["00000000", "00000000", "00000000", "00000000", "00000000",
"00000000", "00000000", "00000000", "00000000", "00000000", "00000000",
"00000000", "00000000", "00000000", "00000000", "00000001"]
```

```
messMatrix := [
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000001"]
]
```

```
> testKeyExpanded := keyExpander(testKey):
```

```
> cipher := ARK(messMatrix, testKeyExpanded, 0);
```

```
hexState := matrixToHex(cipher);
```

```
for i from 1 to 9 do
```

```
  cipher := ARK(MC(SR(BS(cipher))), testKeyExpanded, i);
```

```
  hexState := matrixToHex(cipher);
```

```
end do;
```

```
cipher := ARK(SR(BS(cipher)), testKeyExpanded, 10);
```

```
cipherHex := matrixToHex(cipher);
```

Программа позволяет моделировать состояния *State* системы шифрования.

Первые два раунда имеют следующие состояния:

```
cipher := [
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000000"]
  ["00000000" "00000000" "00000000" "00000000"]
]
```

```
hexState := "00000000000000000000000000000000"
```

```
cipher := [
  ["00000001" "00000001" "00000001" "00000001"]
  ["00000000" "00000000" "00000000" "00000000"]
  ["00011111" "00011111" "00011111" "00011111"]
  ["00000000" "00000000" "00000000" "00000001"]
]
```

```
hexState := "01001F0001001F0001001F0001001F01"
```

Последний раунд шифрования имеет вид

$$cipher := \begin{bmatrix} "10100001" & "11100100" & "10000110" & "01111000" \\ "01111110" & "11110010" & "00100000" & "11101110" \\ "10011111" & "01011010" & "10110100" & "11111101" \\ "01101001" & "10001011" & "10101111" & "01101111" \end{bmatrix}$$

cipherHex := "A17E9F69E4F25A8B8620B4AF78EEFD6F"

*Моделирование процесса расшифрования.* Превратим шифротекст в список векторов и конвертируем этот список в матрицу байт

```
> ListCipher := [seq(substring(cipherHex, 2*i-1..2*i), i = 1..16)];
cipherByteMatrix := map(hexTo8Bits, listToMatrix2(ListCipher));
```

ListCipher := ["A1", "7E", "9F", "69", "E4", "F2", "5A", "8B", "86", "20", "B4",  
"AF", "78", "EE", "FD", "6F"]

$$cipherByteMatrix := \begin{bmatrix} "10100001" & "11100100" & "10000110" & "01111000" \\ "01111110" & "11110010" & "00100000" & "11101110" \\ "10011111" & "01011010" & "10110100" & "11111101" \\ "01101001" & "10001011" & "10101111" & "01101111" \end{bmatrix}$$

Программа 10 раундов расшифрования:

```
> plain := cipherByteMatrix;
plain := InvBS(InvSR(ARK(plain, testKeyExpanded, 10)));
for i from 1 to 9 do
plain := InvBS(InvSR(InvMC(ARK(plain, testKeyExpanded, 10-i))));
end do;
decryptMatrix := ARK(plain, testKeyExpanded, 0);
```

Состояния предпоследнего и последнего раунда расшифрования равны

$$plain := \begin{bmatrix} "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000000" \end{bmatrix}$$

$$decryptMatrix := \begin{bmatrix} "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000000" \\ "00000000" & "00000000" & "00000000" & "00000001" \end{bmatrix}$$

Возврат к исходному тексту. Выполним преобразование сообщения в вектор 16-ичного формата данных

```
> matrixToHex(decryptMatrix);
```

"00000000000000000000000000000000000001"

Проделанные вычисления показывают, что криптографические алгоритмы работают верно.

## 5. ИССЛЕДОВАНИЕ СВОЙСТВ БЛОЧНЫХ КРИПТОСИСТЕМ

## 5.1. Цели работы

1. Изучить принцип построения, свойства и алгоритмы AES -шифра.
2. Изучить правила пользования программного обеспечения к лабораторной работе и программного пакета Maple.
3. Провести моделирование процесса образования информационного блока в различных формах представления.
4. Провести моделирование алгоритмов базовых криптопреобразований.
5. Провести моделирование с помощью программного обеспечения алгоритмов шифрования и расшифрования.
6. Провести моделирование и оценить устойчивость шифра относительно атак криптографического анализа.
7. Провести моделирование и оценить уровень нелинейности дискретной функции на выходе алгоритма AES.

## 5.2. Порядок выполнения работы

Составить и отладить моделирующие алгоритмы программного обеспечения.

### 5.2.1. Моделирование алгоритмов формирования информационного сообщения

1. Получить от преподавателя исходные данные, построить массивы информационного сообщения в различных формах представления.

### 5.2.2. Моделирование алгоритмов базовых преобразований

1. Используя библиотеку «AES.m» инициализировать операторы базовых преобразований и проверить их работоспособность.
2. Составить программы формирования ключевого пространства в различных формах представления информации.

### 5.2.3. Моделирование процесса шифрования и расшифрования

1. Составить программу шифрования и расшифрования по алгоритму AES с возможностью управления количеством раундов и наблюдением состояний.
2. Получить от преподавателя исходные тексты и провести их шифрование и расшифрование на разных секретных ключах.

#### 5.2.4. Моделирование и оценка устойчивости шифра относительно атак криптографического анализа

1. Используя метод проникновения образов активности провести моделирование раундов шифра и оценить их устойчивость по отношению дифференциальному и линейному анализу.

2. Составить программу, моделирующую атаку типа «Квадрат» на последовательности из 256 информационных блоков. Провести анализ свойства сбалансированности, оценить возможность взлома ключа.

#### 5.2.5. Моделирование и оценка уровня нелинейности криптограмм шифра AES

1. Составить программу формирования криптограмм с разными случайными ключами.

2. Используя операторы преобразования Уолша-Адамара вычислить спектр полученных в предыдущем пункте криптограмм и оценить по максимальным значениям спектров уровень нелинейности.

### 5.3. Расчетная часть

#### 5.3.1. Изучение операций базовых криптопреобразований

1. Получить от преподавателя исходные данные в различных формах представления и вычислить функции SubBytes, MixColumns, ShiftRows.

## 6. СОДЕРЖАНИЕ ОТЧЕТА

1. Формулировка цели работы.
2. Схемы алгоритмов шифрования и расшифрования.
3. Результаты моделирования.
4. Анализ свойств шифра AES и их влияние на устойчивость по отношению к атакам криптоанализа.
5. Спектральная оценка уровня нелинейности криптограмм шифра AES
6. Выводы.

## 7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Пояснить сущность выполнения базовых функций с помощью математики конечного поля.
2. Каким образом в шифре AES устраняется симметричность криптографического преобразования ?
3. Чем обоснован выбор минимального полинома  $m(x)$  ?
4. Каким образом базовые функции удовлетворяют критериям перемешивания и рассеивания?
5. Сформулируйте основные атаки криптоанализа блочных, симметричных шифров.
6. Как оценить уровень нелинейности булевой функции?
7. Почему последний раунд шифрования не использует одну из базовых функций?

## ЛИТЕРАТУРА

1. Рябко Б.Я., Фионов А.Н. Основы современной криптографии для специалистов в информационных технологиях. – М.: Научный мир, 2004.-173 с.
2. Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля / Под ред. М.А. Иванова. – М.: КУДИЦ-ОБРАЗ, 2003.– 176 с.
3. Daemen J., Rijmen V. The Rijndael block cipher ([http://csrc.nist.gov/ encryption/aes/ rijndael](http://csrc.nist.gov/encryption/aes/rijndael)).

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. БЛОЧНЫЕ ШИФРЫ .....	4
2. СТАНДАРТ БЛОЧНОЙ СИММЕТРИЧНОЙ КРИПТОСИСТЕМЫ AES.....	6
2.1. Математические предпосылки.....	6
2.2. Алгебраическая структура шифра.....	8
3. СТОЙКОСТЬ К АТАКАМ КРИПТОАНАЛИЗА.....	16
3.1. Дифференциальный криптоанализ.....	16
3.2. Линейный криптоанализ.....	18
3.3. Метод проникновения образов активности.....	18
3.4. Атака «Квадрат».....	20
3.5. Спектральная оценка нелинейности криптографической функции.....	22
4. МОДЕЛИРОВАНИЕ КРИПТОАЛГОРИТМОВ В СРЕДЕ MAPLE.....	23
4.1. Список основных операторов моделирования .....	23
4.2. Моделирование работы алгоритма шифрования.....	25
5. СОДЕРЖАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ.....	28
5.1. Содержание работы.....	28
5.2. Порядок выполнения работы .....	22
5.3. Расчетная часть.....	29
6. СОДЕРЖАНИЕ ОТЧЕТА.....	29
7. КОНТРОЛЬНЫЕ ВОПРОСЫ.....	29
ЛИТЕРАТУРА.....	29

Учебное издание

**Саломатин Сергей Борисович**

## **ИССЛЕДОВАНИЕ СВОЙСТВ БЛОЧНЫХ КРИПТОСИСТЕМ**

Учебно-методическое пособие  
по курсу «Кодирование и защита информации»  
для студентов специальностей  
I-39 01 02 «Радиоэлектронные системы»,  
I-39 01 03 «Радиоинформатика»,  
I-39 01 04 «Радиоэлектронная защита информации»  
дневной формы обучения

В авторской редакции  
Отв. за выпуск С.Н. Воробьева

Подписано в печать 23.06.2006.  
Гарнитура «Таймс».  
Уч.-изд. л. 1,5.

Формат 60x84 1/16.  
Печать ризографическая.  
Тираж 150 экз.

Бумага офсетная.  
Усл. печ. л. 1,98.  
Заказ 362.

---

Издатель и полиграфическое исполнение: Учреждение образования  
«Белорусский государственный университет информатики и радиоэлектроники»  
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.  
220013, Минск, П. Бровки, 6