Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Инженерная математика»

Н.Н. Буснюк

КРИПТОГРАФИЧЕСКИЕ СРЕДСТВА ЗАЩИТЫ ИНФОРМАЦИИ

Учебно-методическое пособие
для студентов специальности 1-38 02 03
«Техническое обеспечение безопасности»
специализации 1-38 02 03 02 «Аппаратно-программные средства
защиты компьютерной информации»

В 2 частях

Часть 2

Учебное электронное пособие

Минск
БНТУ
2011

УДК 512.624.95:378.147.091.3

Рецензенты:

И.Е. Зуйков, заведующий кафедрой «Информационно-измерительная техника и технология» БНТУ, доктор физико-математических наук, профессор;

А.А. Черняк, профессор кафедры математики БГПУ, доктор физико-математических наук.

В учебн-методическом пособии размещены две лабораторные работы. Одна работа посвящена методам шифрования с симметричным ключом и рассматривает Стандарт криптографической защиты – *AES*. Во второй лабораторной работе рассмотрены аппаратные средства криптографической защиты информации серий Криптон и Шипка и квантовый метод шифрования.

Белорусский национальный технический университет
Пр-т Независимости, 65, г. Минск, Республика Беларусь
Тел (017) 292-77-52 факс (017) 292-91-37
Регистрационный № ЭИ БНТУ/ПСФ85-64.2012

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В часть 2 учебно-методического пособия включены две лабораторные работы. Каждая из них состоит из теоретической части и контрольные задания, которые приводятся в конце лабораторных работ.

Лабораторная работа № 8 посвящена методам шифрования с симметричным ключом и рассматривает Стандарт криптографической защиты – *AES* (Advanced Encryption Standard), который является стандартом шифрования США, принятым в 2000 году. К лабораторной работе прилагается программный комплекс «*Visual AES*», реализованный на платформе Windows и включающий:

– средства полноценной реализации алгоритмов шифрования/расшифровывания по стандарту *AES* с использованием основных блочных режимов: *ECB, CBC, CFB, OFB, CTR*;

– средства визуализации и протоколирования раундовых преобразований шифра с возможностями детальной настройки основных параметров;

– средства анализа алгоритма, состоящие из контроля за математическим аппаратом раундовых трансформаций и их пошагово управляемой трассировки.

В лабораторной работе № 9 рассмотрены аппаратные средства криптографической защиты информации серии Криптон и Шипка, а также такой относительно новый метод шифрования, как квантовый метод.

Лабораторная работа № 8

БЛОЧНОЕ ШИФРОВАНИЕ. АЛГОРИТМ *AES*

Цель работы: Изучение устройства и методов работы симметричных криптосистем на примере алгоритма блочного шифрования *AES*. Приобретение навыков шифрования с помощью программы шифрования *Visual AES*.

Необходимые математические сведения

Поле F есть множество, на котором определены операции сложения и умножения, удовлетворяющие требованиям: ассоциативности, коммутативности, дистрибутивности, существования аддитивного 0 и мультипликативной 1, аддитивных обратных и мультипликативных обратных для всех элементов за исключением 0.

Конечное поле $F(p)$ с конечным числом p элементов играет важную роль в криптографии. В общем случае число элементов $p = q^n$, где q – некоторое простое число и $n>1$. Такие конечные поля называют ***полями*** Галуа и обозначают $GF(q^n)$ или $GF(q)$ при $n=1$. (Эварист Галуа – французский математик начала XIX века.) Многие криптосистемы базируются на полях Галуа $GF(q)$, где q – большое простое число.

Если q – простое число, то число $a \in [1,\ q-1]$ является взаимно простым с q, и поэтому обратный элемент a^{-1} имеет единственное значение. Тем самым однозначно определяется операция деления.

Поля Галуа, используемые в алгоритме *AES*, основываются на арифметике по модулю неприводимых многочленов степени n, чьи коэффициенты – целые числа по модулю q, где q – простое. Эти поля Галуа обозначают как $GF\left(q^n\right)$.. Они имеют элементы, которые описываются многочленами степени не выше $(n-1)$ в форме

$$a(X) = a_{n-1}X^{n-1} + \ldots + a_1 X + a_0 .$$

Каждый элемент $a(X)$ является вычетом по модулю $p(X)$, где $p(X)$ – *неприводимый* многочлен степени n (т. е. $p(X)$ нельзя разложить на сомножители – многочлены степени меньше n).

Арифметические действия над коэффициентами a_i выполняются по модулю q, а наивысшая степень X равна $n-1$, так как выполняется приведение по модулю многочлена $p(X)$, имеющего старшую степень n.

Особый интерес представляют поля $GF(2^n)$. Здесь коэффициентами a_i являются 0 и 1. Поэтому многочлен $a(X)$ степени не выше $n-1$ можно представить как вектор из n двоичных цифр:

$$a_{n-1}a_{n-2}...a_1a_0 .$$

Каждый из n-битовых векторов соответствует конкретному элементу поля $GF(2^n)$.

Например, поле Галуа $GF(2^3)$ имеет элементы:

Многочлены	*Двоичная форма*
0	000
1	001
x	010
$x+1$	011
x^2	100
x^2+1	101
x^2+x	110
x^2+x+1	111

Организация вычислений в полях Галуа предполагает знание некоторых свойств многочленов и их корней в двоичном поле $GF(2)$. Кратко приведем некоторые из них:

***Свойство* 1.** Ненулевые элементы поля $GF\left(2n^2\right)$ являются корнями обобщенного многочлена $X^{2^n-1}+1$.

***Свойство* 2.** Каждый многочлен $p(X)$ степени n, неприводимый над полем $GF(2)$, является делителем двучлена $X^{2^n-1}+1$, и каждый делитель двучлена $X^{2^n-1}+1$, неприводимый над полем $GF(2)$, имеет степень, равную n и менее.

***Свойство* 3.** Все элементы поля $GF(2^n)$ можно получить как совокупность остатков от деления 100...00 на неприводимый многочлен $p(X)$, входящий в разложение двучлена $X^{2^n-1}+1$. Эти остатки – корни двучлена $X^{2^n-1}+1$, т.е. обращают его в нуль. Число остатков равно 2^n-1.

***Свойство* 4.** В поле $GF(2^n)$ существует примитивный элемент a, такой что каждый ненулевой элемент поля $GF(2^n)$ может быть представлен как некоторая степень a, т.е. мультипликативная группа $GF(2^n)$ является циклической.

В поле Галуа $GF(2^n)$ определены четыре алгебраические операции. Операции сложения и вычитания выполняются как операции поразрядного сложения по модулю 2 (обозначается $\oplus$ или *XOR*). Операция умножения элементов поля выполняется как умножение соответствующих многочленов с приведением по модулю неприводимого многочлена $p(X)$, т. е. многочлена, по модулю которого построены элементы поля $GF(2^n)$.

Чтобы выполнить деление элемента b на элемент a в поле $GF(2^n)$ по модулю $p(X)$, сначала находят обратный элемент $a^{-1} \pmod{p(X)}$, а затем вычисляют $b \cdot a^{-1} \pmod{p(X)}$.

Каждый двоичный вектор длиной n, исключая 0, является взаимно простым с неприводимым многочленом $p(X)$ независимо от значения $p(X)$. Поэтому число вычетов, взаимно простых с $p(X)$, равно $\varphi(p(X))=2^n-1$ (расширение функции Эйлера для многочленов). Поэтому

$$a^{-1} = a^{\varphi(p(X))-1} \bmod p(X) = a^{2^n-2} \bmod p(X).$$

Достоинства вычислений в поле $GF(2^n)$

1. Все элементы поля Галуа имеют конечный размер, деление элементов не имеет каких-либо ошибок округления.

2. Сложение и вычитание элементов поля $GF(2^n)$ не требует деления на модуль.

3. Алгоритмы вычислений в поле $GF(2^n)$ допускают параллельную реализацию.

4. Для поля $GF(2^n)$ обычно применяют в качестве модуля трехчлен $p(X^n) = X^n + X + 1$.

5. Длинная строка нулей между коэффициентами при X^n и X обеспечивает более простую реализацию быстрого умножения (с приведением по модулю). Трехчлен $p(X^n)$ должен быть неприводимым и примитивным.

6. Трехчлен $p(X^n) = X^n + X + 1$ является примитивным для следующих значений $n\ (n < 1000)$:

1, 3, 4, 6, 9, 15, 22, 28, 30, 46, 60, 63,

127, 153, 172, 303, 471, 532, 865, 900.

Структура шифра *AES*

Формат блоков данных и число раундов

AES – это симметричный блочный шифр, оперирующий блоками данных размером 128 и длиной ключа 128, 192 или 256 битов; название стандарта – соответственно «*AES*-128», «*AES*-192», и «*AES*-256».

Введем следующие обозначения:

Nb – число 32-битовых слов, содержащихся во входном блоке, *Nb* = 4;

Nk – число 32-битовых слов, содержащихся в ключе шифрования, *Nk* = = 4, 6, 8;

Nr – число раундов шифрования – функция от Nb и Nk, $Nr = 10, 12, 14$.

Входные (*input*), промежуточные (*state*) и выходные (*output*) результаты преобразований, выполняемых в рамках криптоалгоритма, называются *состояниями* (*State*). Состояние можно представить в виде матрицы 4×Nb, элементами которой являются байты (четыре строки по Nb байтов) в порядке S_{00}, S_{10}, S_{20}, S_{30}, S_{01}, S_{11}, S_{21}, S_{31}, … . Рис. 1.1 демонстрирует такое представление, носящее название архитектуры «Квадрат».

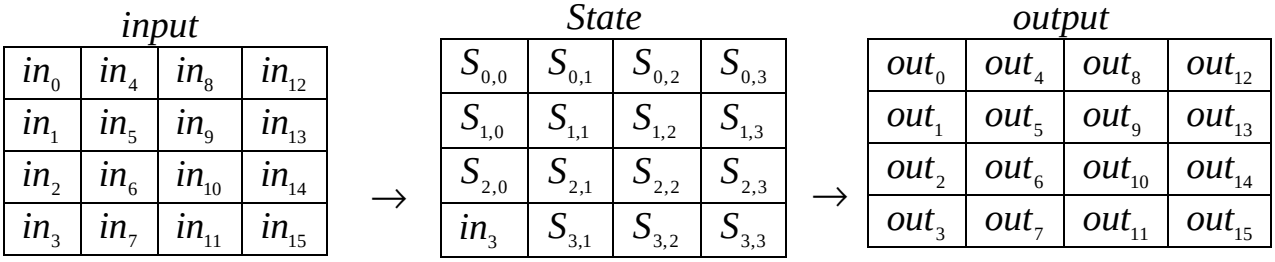


Рис. 1.1. Пример представления блока в виде матрицы 4×Nb

На старте процессов шифрования и дешифрования массив входных данных $in_0, in_1, ..., in_{15}$ преобразуется в массив *State* по правилу

$$s[r, c] = in[r + 4c],$$

где $0 \le r < 4$ и $0 \le c < Nb$.

В конце действия алгоритма выполняется обратное преобразование

$$out[r + 4c] = s[r, c],$$

где $0 \le r < 4$ и $0 \le c < Nb$ – выходные данные, получаются из байтов состояния в том же порядке.

Четыре байта в каждом столбце состояния представляют собой 32-битное слово; если r – номер строки в состоянии, то одновременно он является индексом каждого байта в этом слове. Следовательно, состояние можно представить как одномерный массив 32-битных слов $w_0, ..., w_{Nb}$, где номер столбца состояния c есть индекс в этом массиве. Тогда состояние можно представить так:

$w_0 = S_{0,0}, S_{1,0}, S_{2,0}, S_{3,0}$	$w_2 = S_{0,2}, S_{1,2}, S_{2,2}, S_{3,2}$
$w_1 = S_{0,1}, S_{1,1}, S_{2,1}, S_{3,1}$	$w_3 = S_{0,3}, S_{1,3}, S_{2,3}, S_{3,3}$

Ключ шифрования так же, как и массив *State*, представляется в виде прямоугольного массива с четырьмя строками. Число столбцов этого массива равно *Nk*. Для алгоритма *AES* число раундов *Nr* определяется на старте в зависимости от значения *Nk* (табл. 1.1).

Таблица 1.1

Зависимость значения *Nr* от *Nk* и *Nb*

	Nk	*Nb*	*Nr*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Функция зашифровки

Введем следующие обозначения:

SubBytes( )– замена байтов – побайтовая нелинейная подстановка в *State*-блоках (*SBox*) с использованием фиксированной таблицы замен размерностью 8×256;

ShiftRows( ) – сдвиг строк – циклический сдвиг строк массива *State* на различное количество байтов;

MixColumns( ) – перемешивание столбцов – умножение столбцов состояния, рассматриваемых как многочлены в $GF(2^8)$;

AddRoundKey( ) – сложение с раундовым ключом – поразрядное *XOR* содержимого *State* с текущим фрагментом развернутого ключа.

После заполнения массива *State* элементами входных данных к нему применяется преобразование *AddRoundKey*( ), далее, в зависимости от величины *Nk*, массив *State* подвергается раундовой трансформации 10, 12 или 14 раз, причем в финальный раунд является несколько укороченным – в нем отсутствует преобразование *MixColumns*( ). Выходными данными описанной последовательности операций является шифротекст – результат действия функции зашифровки *AES*.

Функции расшифровки

В спецификации алгоритма *AES* предлагается два вида реализации функции расшифровки, отличающиеся друг от друга последовательностью приложения преобразований, обратных преобразованиям функции зашифровки, и последовательностью планирования ключей.

Введем следующие обозначения:

InvSubBytes( ) – обратная *SubBytes*( ) замена байтов – побайтовая нелинейная подстановка в *State*-блоках с использованием фиксированной таблицы замен размерностью 8×256 (*inverse affain map*);

InvShiftRows( ) – обратный сдвиг строк *ShiftRows*( )– циклический сдвиг строк массива *State* на различное количество байтов;

InvMixColumns( ) – восстановление значений столбцов – умножение столбцов состояния, рассматриваемых как многочлены в $GF(2^8)$.

Функция обратного расшифрования

Если вместо *SubBytes*( ), *ShiftRows*( ), *MixColumns*( ) и *AddRoundKey*( ) в обратной последовательности выполнить инверсные им преобразования, можно построить функцию обратной расшифровки. При этом порядок использования раундовых ключей является обратным по отношению к тому, который используется при зашифровке.

Алгоритм обратной расшифровки, описанный выше, имеет порядок операций-функций, обратный порядку операций в алгоритме прямой зашифровки, но использует те же параметры развернутого ключа. Изменив определенным образом последовательность планирования ключа, можно построить еще один алгоритм – алгоритм прямой расшифровки.

Два следующих свойства позволяют сделать это:

• порядок использования функций *SubBytes*( ) и *ShiftRows*( ) не играет роли. То же самое верно и для операций *InvSubBytes*( )и *InvShiftRows*( ). Это происходит потому, что функции *SubBytes*( ) и *InvSubBytes*( ) работают с байтами, а операции *ShiftRows*( ) и *InvShiftRows*( ) сдвигают целые байты, не затрагивая их значений;

• операция *MixColumns*( ) является линейной относительно входных данных, что означает *InvMixColumns*(*State XOR RoundKey*) = *InvMixColumns*(*State*) *XOR InvMixColumns*(*RoundKey*).

Эти свойства функций алгоритма шифрования позволяют изменить порядок применения функций *InvSubBytes*( ) и *InvShiftRows*( ). Функции *AddRoundKey*( ) и *InvMixColumns*( ) также могут быть применены в обратном порядке, но при условии, что столбцы (32-битовые слова) развернутого ключа расшифровки предварительно пропущены через функцию *InvMixColumns*( ).

Таким образом, можно реализовать более эффективный способ расшифровки с тем же порядком приложения функций, что и в алгоритме зашифровки.

Алгоритм выработки ключей (*Key Schedule*)

Введем следующие обозначения:

Rcon[ ] – массив 32-битовых раундовых констант;

RotWord( ) – операция циклической перестановки входного 4-байтового слова в выходное по следующему правилу $[a_0, a_1, a_2, a_3] \to [a_1, a_2, a_3, a_0]$;

SubWord( ) – операция замены в 4-байтном слове с помощью *S-Box* каждого байта;

$\oplus$ – операция поразрядного сложения по модулю 2 (или *XOR*).

Раундовые ключи получаются из ключа шифрования 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 битов раундовых ключей равно длине блока, умноженной на число раундов, плюс 1;

• ключ шифрования расширяется в *расширенный ключ* (*Expanded Key*);

• раундовые ключи берутся из расширенного ключа следующим образом: первый раундовый ключ содержит первые *Nb* слов, второй – следующие *Nb* слов и т. д.

Расширение (планирование) ключа

Расширенный ключ (рис. 1.2) представляет собой линейный массив $w[i]$, состоящий из $4(Nr+1)$ 4-байтовых слов, $i=\overline{0,4(Nr+1)}$.

w_0 w_1 w_2 w_3	w_4	w_5 w_6 w_7	w_8	w_9 w_{10} w_{11}	w_{12}	w_{13} w_{14} …
Раундовый ключ 0	Раундовый ключ 1		Раундовый ключ 2		…	

Рис. 1.2. Процедуры расширения и выборки раундового ключа для *Nk* = 4

Светло-серым цветом выделены слова расширенного ключа, которые формируются без использования функций *SubWord*( ) и *RotWord*( ). Темно-

серым цветом выделены слова расширенного ключа, при вычислении которых используются преобразования $SubWord(\ )$ и $RotWord(\ )$.

Первые Nk слов содержат ключ шифрования. Каждое последующее слово $w[i]$ получается посредством XOR предыдущего слова $w[i-1]$ и слова на Nk позиций ранее

$$w[i-Nk]: w[i] = w[i-1] \oplus w[i-Nk].$$

Для слов, позиция которых кратна Nk, перед XOR применяется преобразование к $w[i-1]$, а затем еще прибавляется раундовая константа $Rcon[i]$. Преобразование реализуется с помощью двух дополнительных функций: $RotWord(\ )$ и $SubWord(\ )$.

Значение $Rcon[j]$ равно $2j - 1$. Значение $w[i]$ в этом случае определяется выражением $w[i] = SubWord(RotWord(w[i-1])) \oplus Rcon[i/Nk] \oplus w[i-Nk]$.

Выбор раундового ключа

i-тый раундовый ключ выбирается из слов массива расширенного ключа в промежутке от $W[Nb \cdot i]$ до $W[Nb \cdot (i+1)]$.

Для функции зашифровки расширенный ключ генерируется уже упоминавшимся выше алгоритмом. Для функции обратной расшифровки используется этот же ключ, но в обратной последовательности, начиная с последнего раундового подключа зашифровки.

Для функции прямой расшифровки используется несколько модифицированный алгоритм планирования ключа. При формировании развернутого ключа в процедуру планирования необходимо добавить в конце дополнительное преобразование, причем расширенный ключ используется при прямой расшифровке в той же последовательности, что и при зашифровке.

Раундовое преобразование

Раундовое преобразование состоит из последовательного применения к массиву *State* следующих математических трансформаций.

Замена байтов. Нелинейная замена байтов массива состояния посредством трансформации $SubBytes(\ )$ имеет вид:

$$\lambda(f) \equiv \left((X^4 + X^3 + X^2 + X + 1)\cdot f + X^6 + X^5 + X + 1\right) \bmod (X^8 + 1).$$

Многократное вычисление в процессе зашифровки данного выражения оказывало бы неоправданную вычислительную нагрузку на исполняющую систему, поэтому для практической реализации наиболее приемлемым решением является использование предварительно вычисленной таблицы замены *S-Box*. Логика работы *S-Box* при преобразовании байта {*xy*} отражена в шестнадцатеричном виде на рис. 1.3.

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

Рис. 1.3. Таблица *S-Box* замены байтов

Использование данной таблицы сводит операцию $SubBytes(\ )$ к простейшей выборке байта из массива $\lambda(f) = Sbox[f]$.

В функциях расшифровки применяется операция, обратная *SubBytes*( ), – *InvSubBytes*( ). Математическая интерпретация ее выражается следующей формулой:

$$\lambda^{-1}(f) \equiv \left(\left(X^6 + X^3 + X\right) \cdot f + X^2 + 1\right) \bmod \left(X^8 + 1\right).$$

Она реализуется так же просто, как и предыдущая, посредством инверсной таблицы $Sbox - \lambda^{-1}(f) = InvSbox[f]$. Ее логика работы при преобразовании байта $\{xy\}$ отражена в шестнадцатеричном виде на рис. 1.4.

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

Рис. 1.4. Таблица *S-Box* инверсной замены байтов

Рис. 1.5 иллюстрирует применение преобразования *замены байт* к состоянию в функциях зашифровки и расшифровки:

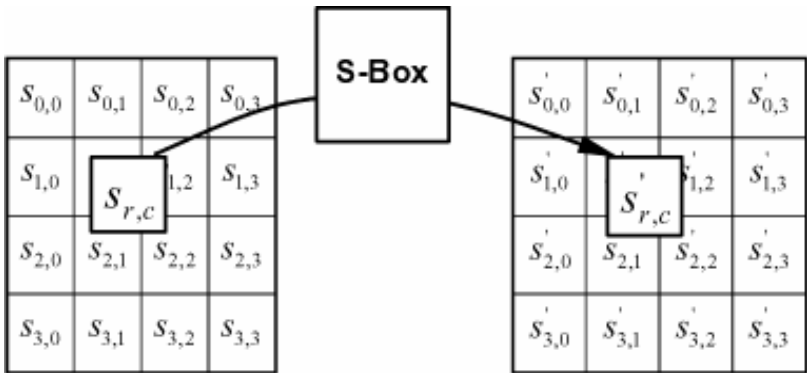


Рис. 1.5. Преобразование состояния посредством таблицы замены *S-Box*

Сдвиг строк. В преобразовании сдвига строк (*ShiftRows*) последние 3 строки состояния циклически сдвигаются влево на различное число байтов. Строка 1 сдвигается на C_1 байтов, строка 2 – на C_2 байт, и строка 3 – на C_3 байтов. Значения сдвигов C_1, C_2 и C_3 в *AES* зависят от длины блока *Nb*. Их величины приведены в табл. 1.2.

Таблица 1.2

N_b	$shift(0, Nb) = C_0$	$shift(1, Nb) = C_1$	$shift(2, Nb) = C_2$	$shift(3, Nb) = C_3$	
4	0	1	2	3	*AES*
6	0	1	2	3	
	0	1	3	4	

В стандарте *AES*, где определен единственный размер блока, равный 128 битам, $C_1 = 1$, $C_2 = 2$ и $C_3 = 3$.

Операция сдвига содержимого состояния может быть представлена следующим выражением: $S_{r,c} = S_{r,(c+shift(r,Nb)) \bmod Nb}$ для $0 < r < 4$ и $0 \le c < Nb$, где значение *shift*(*r*, *Nb*) зависит от номера строки *r*.

Рис. 1.6 показывает влияние преобразования *ShiftRows*( ) на состояние.

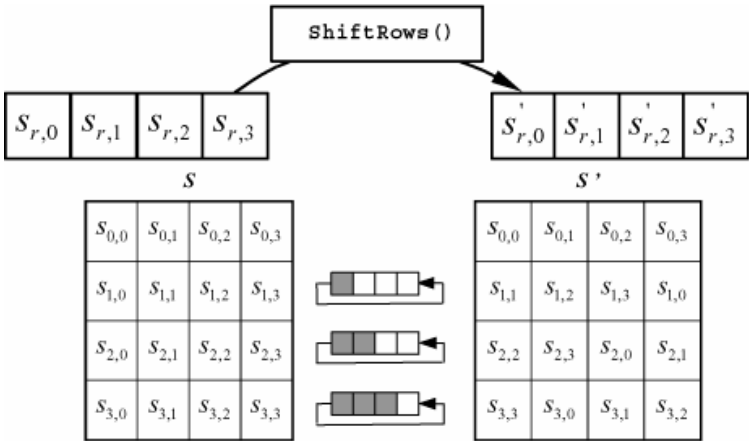


Рис. 1.6. Преобразование сдвига строк в функции зашифровки

В преобразовании обратного сдвига строк *InvShiftRows* последние 3 строки состояния циклически сдвигаются вправо на различное число байтов. Строка 1 сдвигается на C_1 байтов, строка 2 – на C_2 байтов, и строка 3 – на C_3 байтов. Значения сдвигов C_1, C_2 и C_3 зависят от длины блока *Nb*. Их величины естественно соответствуют значениям, приведенным в таб. 1.2. Операция обратного сдвига $InvShiftRows(\ )$ содержимого состояния представлена выражением

$$S'_{r,(c+shift(r,Nb))\bmod Nb} = S_{r,c}$$

для $0 < r < 4$ и $0 \le c < Nb$, где значение $shift(r, Nb)$ зависит от номера строки r. Рис. 1.7 иллюстрирует преобразования обратного сдвига строк состояния.

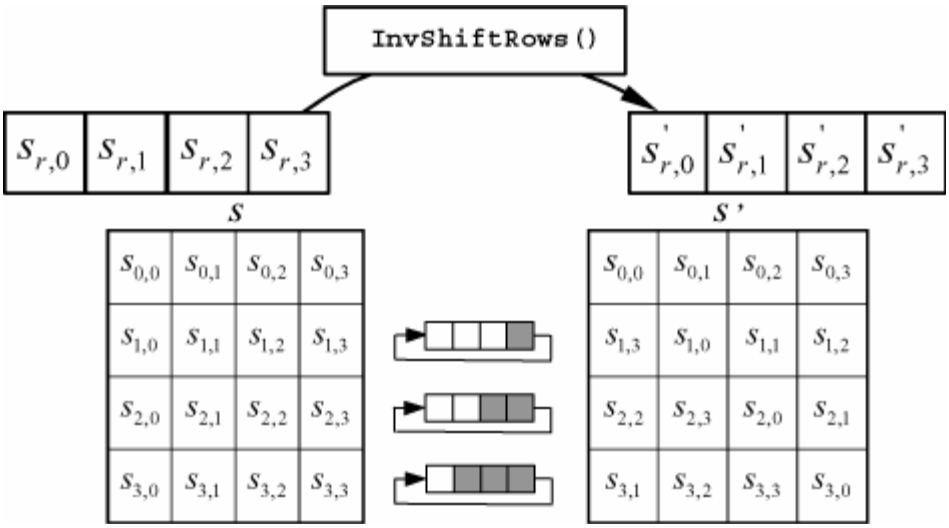


Рис. 1.7. Обратный сдвиг строк в функции расшифровки

Перемешивание столбцов. В преобразовании перемешивания столбцов (*MixColumns*) столбцы состояния рассматриваются как многочлены в $GF(2^8)$ и подвергаются преобразованию $\mu(g) \equiv c \cdot g \bmod(Y^4 + 1)$, где $c = (X, 1, 1, X + 1)$, т. е умножаются по модулю $x^4 + 1$ на многочлен $c(x)$, выглядящий как

$$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

Это преобразование может быть представлено в матричном виде следующим образом:

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

где c – номер столбца массива $State$.

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

$$S'_{0c} = (\{02\} \bullet S_{0c}) \oplus (\{03\} \bullet S_{1c}) \oplus S_{2c} \oplus S_{3c},$$
$$S'_{1c} = S_{0c} \oplus (\{02\} \bullet S_{1c}) \oplus (\{03\} \bullet S_{2c}) \oplus S_{3c},$$
$$S'_{2c} = S_{0c} \oplus S_{1c} \oplus (\{02\} \bullet S_{2c}) \oplus (\{03\} \bullet S_{3c}),$$
$$S'_{3c} = (\{03\} \bullet S_{0c}) \oplus S_{1c} \oplus S_{2c} \oplus (\{02\} \bullet S_{3c}).$$

Применение этой операции ко всем четырем столбцам состояния обозначается как *MixColumns*(*State*). Рис. 1.8 демонстрирует применение преобразования $MixColumns(\ )$ к столбцу состояния.

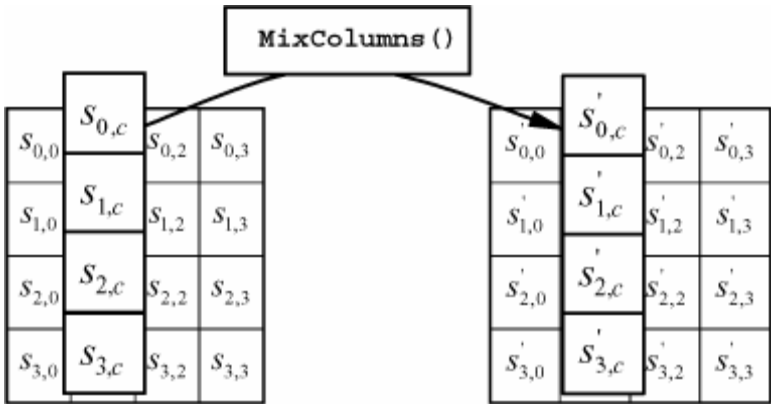


Рис. 1.8. Действие операции перемешивания на столбцы состояния

В обратном преобразовании $InvMixColumns(\ )$ столбцы состояния рассматриваются как многочлены в $GF(2^8)$, но, естественно, подвергаются обратному преобразованию $v(g) = \mu^{-1}(g) \equiv d \cdot g \bmod (Y^4 + 1)$, где

$d = \left(X^3 + X^2 + X,\ X^3 + 1,\ X^3 + X^2 + 1,\ X^3 + X + 1\right)$, т. е. умножаются по модулю $x^4 + 1$ на многочлен $d(x)$, выглядящий следующим образом:

$$d(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}.$$

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} S'_{0c} \\ S'_{1c} \\ S'_{2c} \\ S'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0c} \\ S_{1c} \\ S_{2c} \\ S_{3c} \end{bmatrix}, \quad 0 \le c \le 3,$$

где c – номер столбца массива *State*.

В результате на выходе получаются следующие байты:

$$S'_{0c} = (\{0e\} \bullet S_{0c}) \oplus (\{0b\} \bullet S_{1c}) \oplus (\{0d\} \bullet S_{2c}) \oplus (\{09\} \bullet S_{3c}),$$
$$S'_{1c} = (\{09\} \bullet S_{0c}) \oplus (\{0e\} \bullet S_{1c}) \oplus (\{0b\} \bullet S_{2c}) \oplus (\{0d\} \bullet S_{3c}),$$
$$S'_{2c} = (\{0d\} \bullet S_{0c}) \oplus (\{09\} \bullet S_{1c}) \oplus (\{0e\} \bullet S_{2c}) \oplus (\{0b\} \bullet S_{3c}),$$
$$S'_{3c} = (\{0b\} \bullet S_{0c}) \oplus (\{0d\} \bullet S_{1c}) \oplus (\{09\} \bullet S_{2c}) \oplus (\{0e\} \bullet S_{3c}).$$

Рис. 1.8 может также являться демонстрацией применения обратного преобразования *InvMixColumns*( ) к столбцу состояния в функции расшифрования.

Добавление раундового ключа *AddRoundKey*( ). В данной операции раундовый ключ добавляется к состоянию посредством поразрядного *XOR*. Длина ключа (в 32-разрядных словах) равна длине блока *Nb*. Преобразование, содержащее добавление раундового ключа к состоянию, представляется выражением

$$[S_{0c}, S_{1c}, S_{2c}, S_{3c}] = [S_{0c}, S_{1c}, S_{2c}, S_{3c}] \oplus [w_{r(Nb+c)}]$$

где $0 \le c \le Nb$ и $0 \le r \le Nr$.

Рис. 1.9 иллюстрирует действие данной операции на состояние. Это преобразование обозначается как *AddRoundKey*(*State*, *RoundKey*).

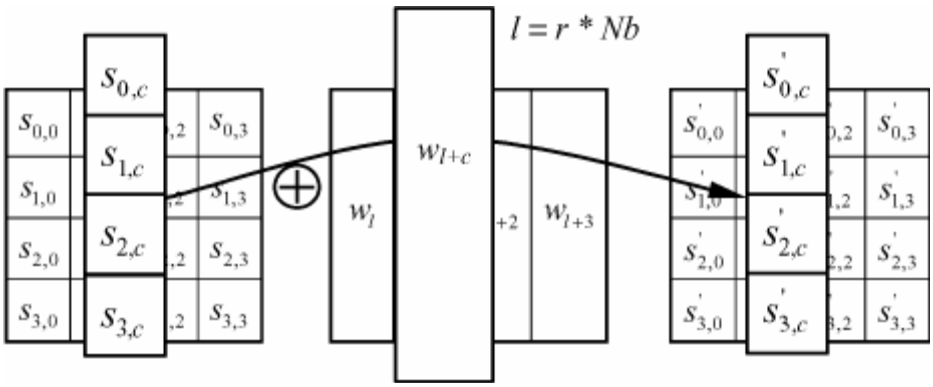


Рис. 1.9. Действие *AddRoundKey*(*State*, *RoundKey*) на столбцы состояния

Основные особенности AES

• Новая архитектура «Квадрат», обеспечивающая быстрое рассеивание и перемешивание информации, при этом за один раунд преобразованию подвергается весь входной блок;

• Байт-ориентированная структура, удобная для реализация на 8- разрядных микроконтроллерах;

• Все раундовые преобразования являются операциями в конечных полях, допускающими эффективную аппаратную и программную реализацию на различных платформах.

Система *Visual AES*

Программный комплекс запроектирован, как многофункциональная система визуализации, анализа и реализации шифрования данных с использованием алгоритма *AES*. Основными компонентами системы *Visual AES* (рис. 1.10) являются:

• подсистема шифрования по стандарту *AES* – модули реализующие зашифровку/расшифровку с поддержкой большинства существующих режимов шифрования с обратной связью (*ECB*, *CBC*, *CFB*, *OFB* и *CTR*);

• подсистема визуализации с помощью соответствующего графического материала, а также специально встроенного инструментария («Журнал»,

«Эксперт»), раскрывающая суть алгоритма и принципов преобразований, реализованных в нем;

• подсистема анализа, включающая в себя средства управления математическим аппаратом преобразований, проводимых в процессе шифрования, и визуальная реализация известной атаки «Квадрат» на криптоалгоритм *AES*.

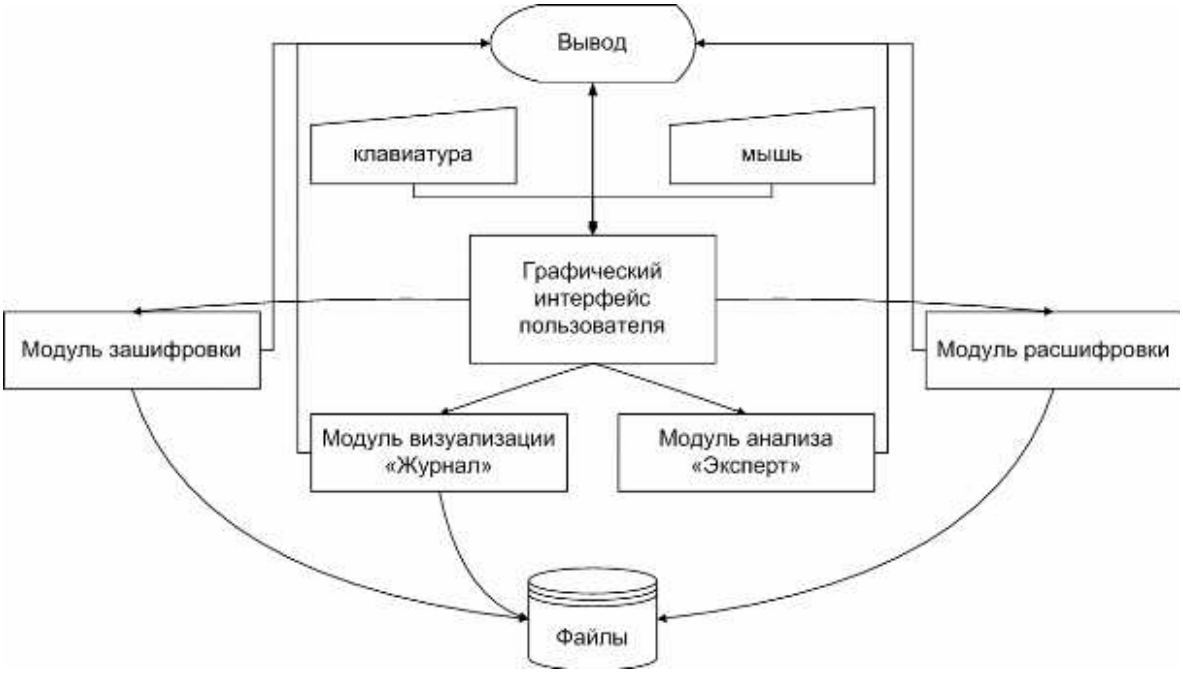


Рис. 1.10. Схема взаимодействия модулей программы *Visual AES*

Для начала работы необходимо произвести первичную подготовку данных, сюда входит загрузка проекций файлов открытого текста и шифротекста (вне зависимости от выбираемого направления зашифровки/расшифровки их размеры должны быть синхронизированы, что обусловливается условиями блочного криптоалгоритма) и ввод ключа, остальные настройки выбираются в зависимости от поставленной задачи.

В зависимости от дальнейшего выбора пользователем программы возможны следующие операции:

• зашифровка/расшифровка;

• подстройка математического аппарата преобразований для получения новых мутаций алгоритма, их исследования и, возможно, применения в обмене шифрованной информацией (файлами) в рамках системы *Visual AES*;

• наблюдение за промежуточными результатами раундовых преобразований и возможная их передача в другие системы, например, статистические или математические, для дальнейшего анализа с привлечением дополнительных методов.

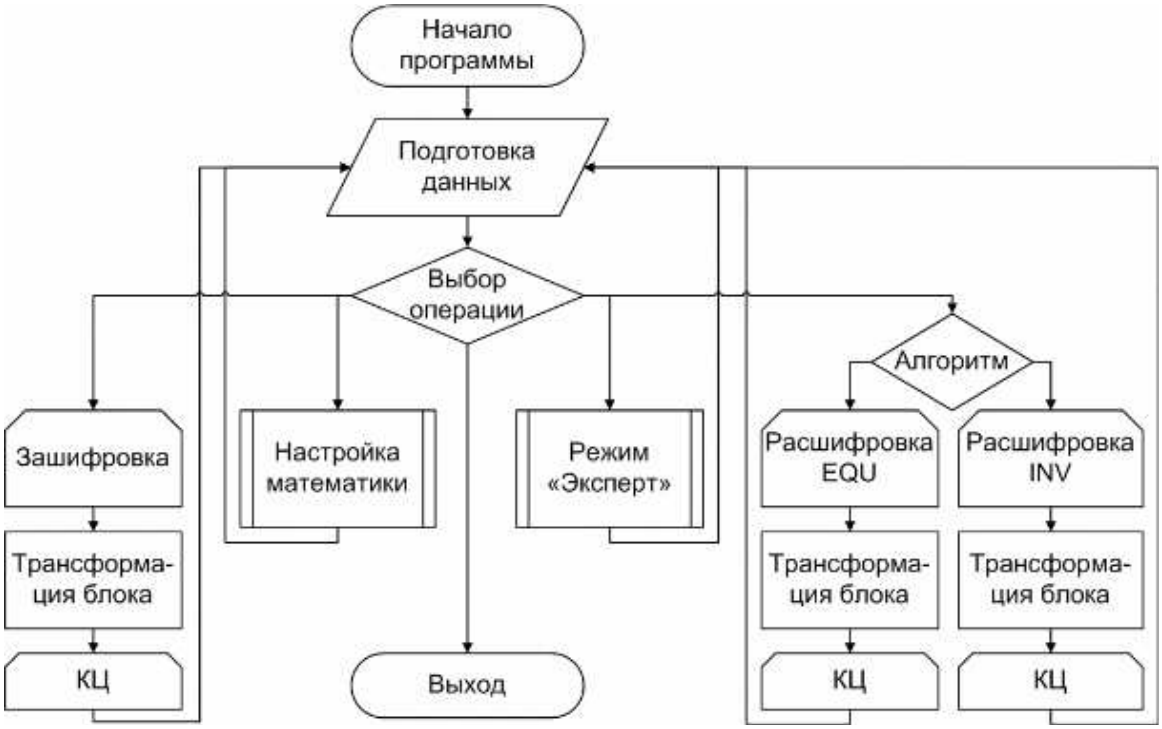


Рис. 1.11. Схема программы *Visual AES*

Модель данных, представленная на рис. 1.12, демонстрирует информационное взаимодействие модулей системы *Visual AES*:

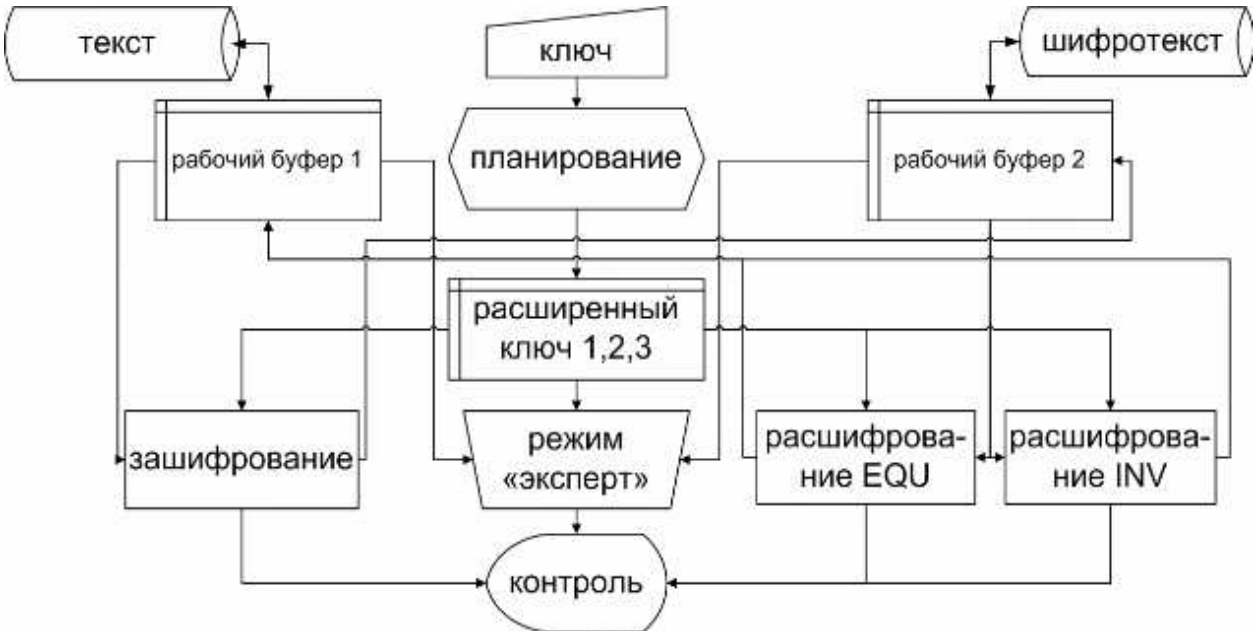


Рис. 1.12. Схема данных программы *Visual AES*

Основными информационными блоками в работе программы являются буферы файлов открытого и шифротекста. На базе вводимого пользователем ключа шифрования система автоматически генерирует три ключевых последовательности: для зашифровки, обратной и инверсной расшифровки (планирование подключей). Операции зашифровки ($РБ1 \Rightarrow РБ2$) и расшифровки ($РБ2 \Rightarrow РБ1$) перемещают текст и шифротекст между буферами, что позволяет произвести дополнительную проверку и убедиться в правильности результатов обращения операции.

Режим «Эксперт» обладает собственными рабочими буферами, что позволяет оперировать данными из исходных файлов, не внося в них изменений. При анализе атаки «Квадрат» создается закрытое множество λ-*set* в динамической памяти, а ключевая информация доступна только для чтения, что также не оказывает разрушающих действий на исходные файлы.

Рис. 1.13. иллюстрирует работу системы *Visual AES*. Стрелками показаны направления информационного взаимодействия между инструментами системы.

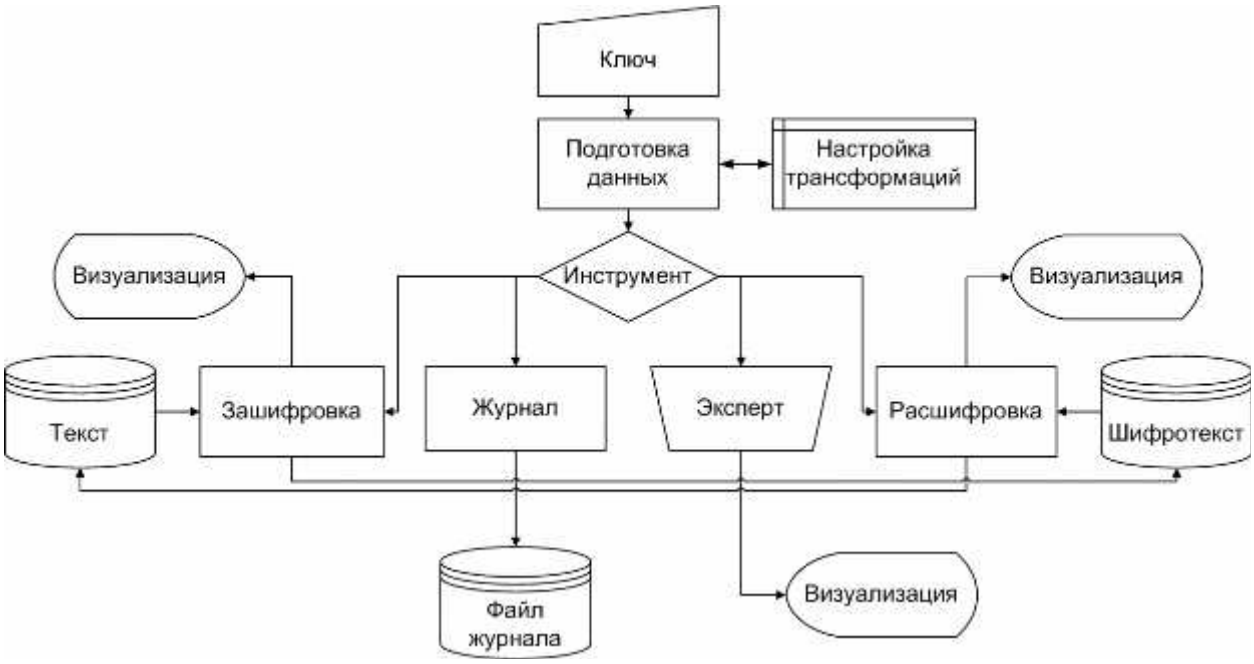


Рис. 1.13. Схема работы системы *Visual AES*

Visual AES как средство шифрования

Как видно из рис. 1.14, система *Visual AES* кроме реализации блочного преобразования по стандарту *AES* обладает всеми необходимыми средствами для выполнения операций зашифровки/расшифровки (128-, 192-, 256-битовых ключей и блоков данных независимо друг от друга и, соответственно, 10-, 12- и 14-раундовых преобразований) с применением основных режимов блочного шифрования с обратной связью. Ограничения на размер шифруемых данных зависят только от количества адресуемой виртуальной памяти, доступной системе.

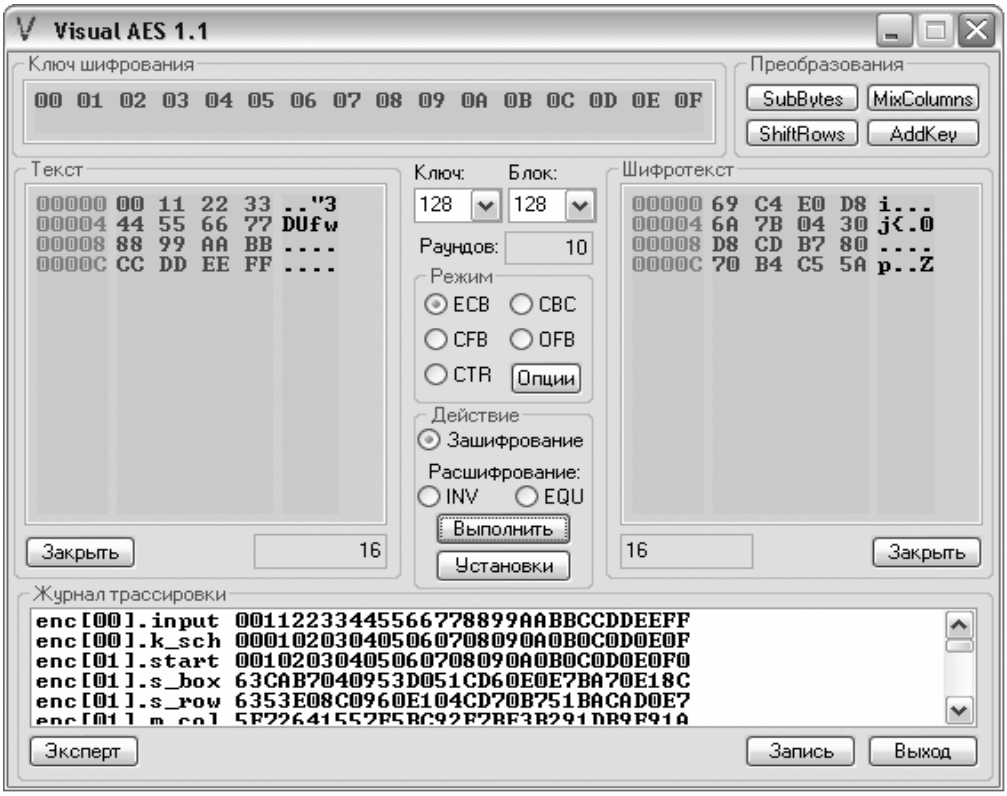


Рис. 1.14. Основное окно программы *Visual AES*

***Режим электронной кодовой книги* (*Electronic Codebook, ECB*)**

В режиме *ECB* каждый блок открытого текста заменяется блоком шифротекста. Все блоки открытого текста шифруются независимо друг от друга. Это важно для шифрованных файлов с произвольным доступом, например, файлов баз данных. Если база данных зашифрована в режиме *ECB*, любая запись может быть добавлена, удалена, зашифрована или расшифрована независимо от любой другой записи (при условии, что каждая запись состоит из целого числа блоков шифрования). Кроме того, обработка может быть параллельной: если используются несколько шифровальных процессоров, они могут зашифровывать или расшифровывать различные блоки независимо друг от друга.

На рис. 1.15 изображаено окно установок *ECB* и диаграмма операций, производимых программой *Visual AES* в данном режиме.

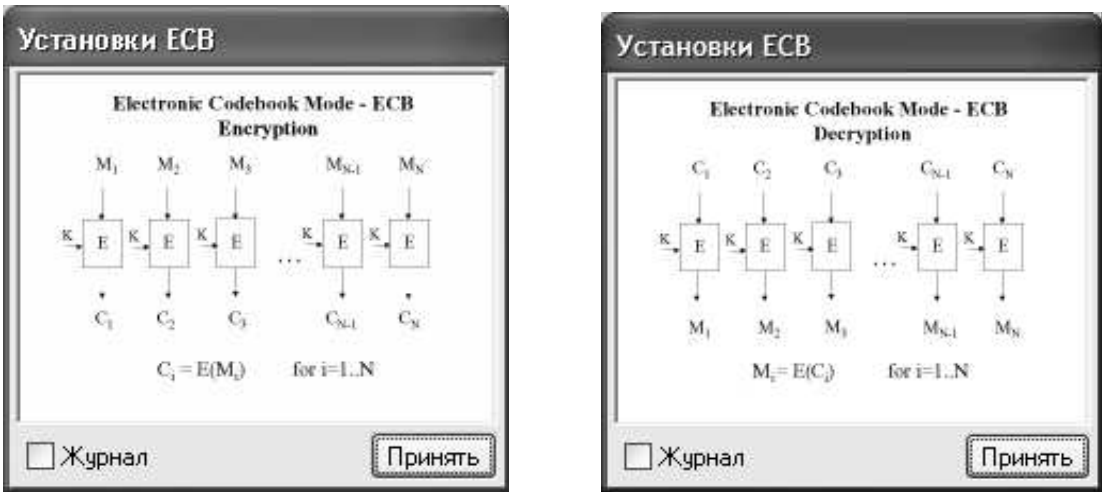


Рис. 1.15. Окно установок режима *ECB* программы *Visual AES*

К недостаткам режима *ECB* можно отнести то обстоятельство, что, если у криптоаналитика есть открытый текст и шифротекст нескольких сообщений, он может, не зная ключа, начать составлять шифровальную книгу. В большинстве реальных ситуаций фрагменты сообщений имеют тенденцию повторяться. Сообщения могут быть высокоизбыточными или содержать длинные строки нулей или пробелов.

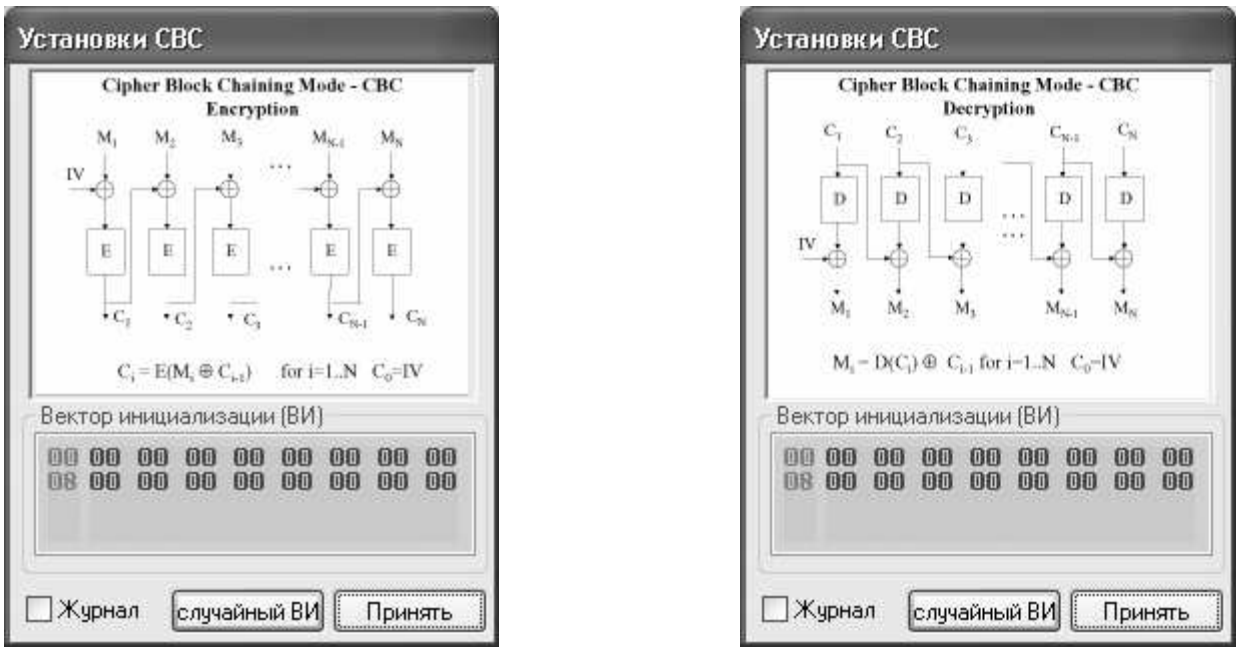
К достоинствам режима *ECB* можно отнести возможность шифрования нескольких сообщений одним ключом без снижения надежности. По существу, каждый блок можно рассматривать как отдельное сообщение, зашифрованное одним и тем же ключом. Ошибки в символах шифротекста ведут к некорректной расшифровке соответствующего блока открытого текста, однако не затрагивают остальной открытый текст.

Большинство сообщений не делятся точно на *n*-битовые блоки шифрования – в конце обычно оказывается укороченный блок. Однако режим *ECB* требует использовать строго *n*-битовые блоки. Для решения этой проблемы используют дополнение. Чтобы создать полный блок, последний блок дополняют некоторым стандартным шаблоном – нулями, единицами, чередующимися нулями и единицами.

Режим сцепления блоков шифротекста (*Cipher Block Chaining, CBC*)

Сцепление добавляет в блочный шифр механизм обратной связи: каждый блок используется для модифицирования шифрования следующего блока. Каждый блок шифротекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста. В режиме сцепления блоков шифротекста перед шифрованием над открытым текстом и предыдущим блоком шифротекста выполняется операция *XOR*. На рис. 1.16, *а* показан процесс шифрования в режиме *CBC*. Когда блок открытого текста зашифрован, полученный шифротекст сохраняется в регистре обратной связи. Следующий блок открытого текста перед шифрованием подвергается операции *XOR* с содержимым регистра обратной связи. Результаты операции *XOR* используются как входные данные для следующего этапа процедуры шифрования. Полученный шифротекст снова сохраняется в регистре обратной связи, чтобы подвергнуться операции *XOR* вместе со следующим блоком открытого текста, и так до конца сообщения. Шифрование каждого блока зависит от всех предыдущих блоков.

Расшифровка выполняется в обратном порядке (рис. 1.16, *б*). Блок шифротекста расшифровывается обычным путем, но сохраняется в регистре обратной связи. Затем следующий блок расшифровывается и подвергается операции *XOR* с содержимым регистра обратной связи. Теперь следующий блок шифротекста сохраняется в регистре обратной связи и так далее до конца сообщения.



а *б*

Рис. 1.16. Окна установок режима *CBC* программы *Visual AES*:

а – завшифровка; *б* – расшифровка

При шифровании в режиме *CBC* одинаковые блоки открытого текста превращаются в различающиеся друг от друга блоки шифротекста только в том случае, если различались какие-то предшествующие блоки открытого текста. Однако при шифровании двух идентичных сообщений создается один и тот же шифротекст.

Чтобы избежать этого, можно зашифровать в первом блоке какие-то произвольные данные. Этот блок случайных данных называют *вектором инициализации* (*IV*) (*Initialization Vector*, русский термин – синхропосылка), инициализирующей переменной или начальным значением сцепления. В качестве вектора *IV* удобно использовать метку времени либо какие-то случайные биты.

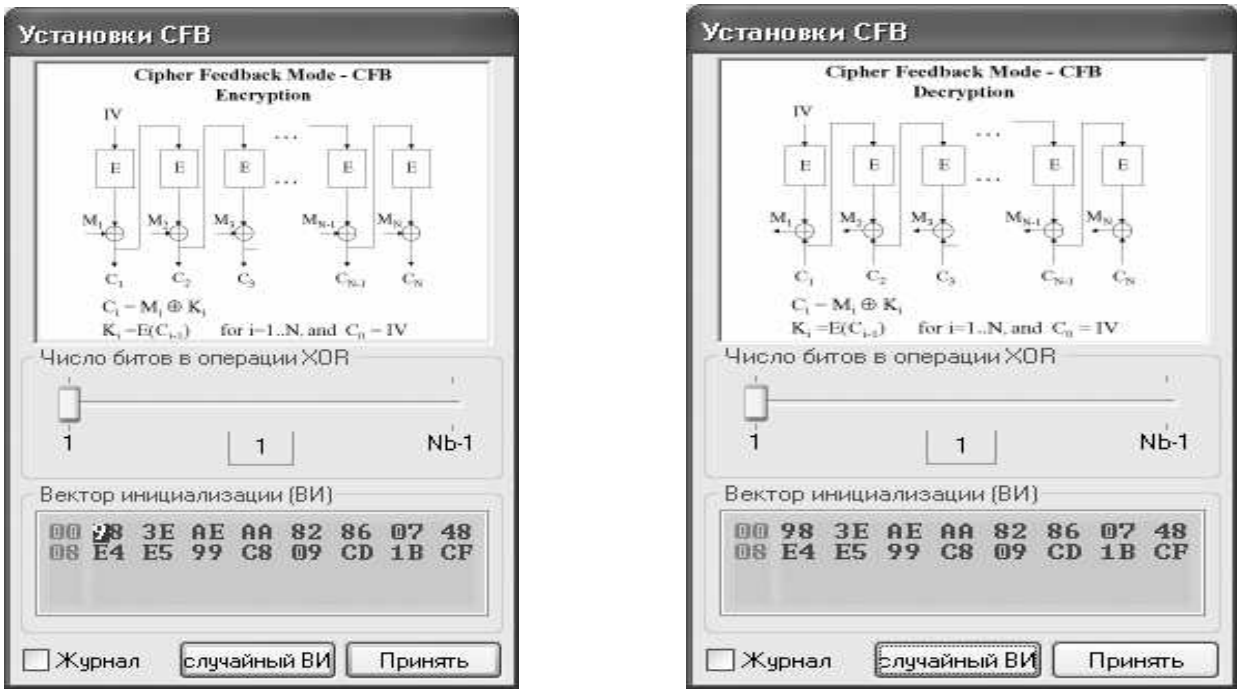
Вектор *IV* не обязательно хранить в секрете, его можно передавать открыто вместе с шифротекстом. Дополнение используется так же, как и в режиме *ECB*.

Режим обратной связи по шифротексту (*Cipher-Feedback, CFB*)

В режиме *CBC* начать шифрование до поступления полного блока данных невозможно. Для некоторых сетевых приложений это создает проблемы. Например, в защищенном сетевом окружении терминал должен иметь возможность передавать хосту каждый символ сразу после ввода. Если же данные нужно обрабатывать блоками в несколько байтов, режим *CBC* просто не работает.

В режиме *CFB* можно шифровать единицы данных размером не больше блока. Один из вариантов – шифрование по одному байту (этот режим называют 8-битовым *CFB*) с помощью 1-битового *CFB* можно шифровать данные и побитово, но полное шифрование блочным шифром единственного бита требует много ресурсов.

Рис. 1.17, *а*, *б* – демонстрация возможностей системы *Visual AES* по поддержке режима *n*-битового CFB, где $1 \le n \le Nb$.



а *б*

Рис. 1.17. Окна установок режима *CFB* программы *Visual AES*:

а – зашифровка; *б* – расшифровка

Как и в режиме *CBC*, первоначально очередь заполнена вектором инициализации *IV*. Очередь шифруется, затем выполняется операция *XOR* над *n* старшими (крайними левыми) битами результата и первым *n*-битовым символом открытого текста. В результате появляется первый *n*-битовый символ шифротекста. Теперь этот символ можно передать. Кроме того, полученные *n* битов попадают в очередь на место *n* младших битов, а все остальные биты сдвигаются на *n* позиций влево. Предыдущие *n* старших битов отбрасываются. Затем точно также шифруются следующие *n* битов открытого текста. Расшифровка выполняется в обратном порядке. Обе стороны – шифрующая и расшифровывающая – используют блочный алгоритм в режиме шифрования. Как и режим *CBC*, режим *CFB* сцепляет символы открытого текста, чтобы шифротекст зависел от всего предыдущего открытого текста.

Режим обратной связи по выходу (*Output-Feedback, OFB*)

Режим *OFB* (рис. 1.18) представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим подобен режиму *CFB* за исключением того, что *n* битов предыдущего выходного блока сдвигаются в крайние правые позиции очереди. Расшифровка выполняется в обратном порядке. Такой режим называют *n*-битовым режимом *OFB*.

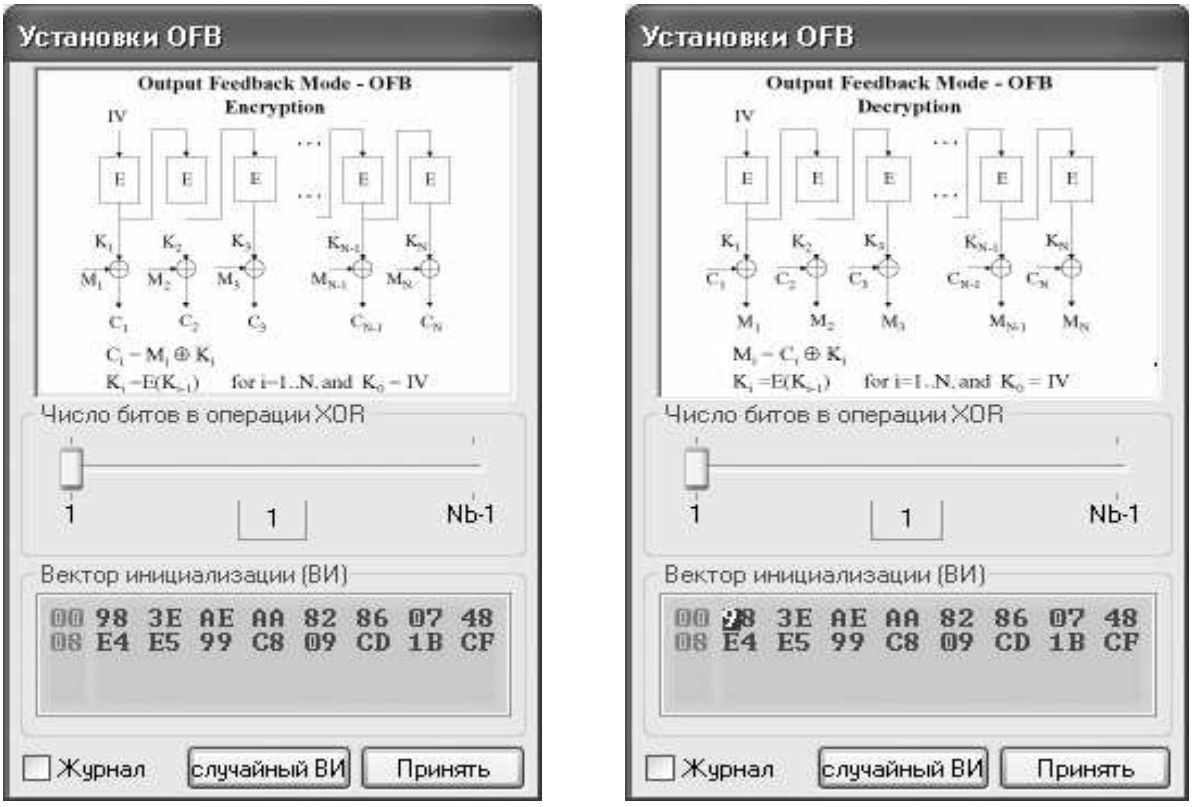
К достоинствам режима *OFB* относится то, что большую часть работы можно выполнить оффлайново, даже когда открытого текста сообщения еще вовсе не существует. Когда наконец сообщение поступает, для создания шифротекста необходимо выполнить операцию *XOR* над сообщением и выходом алгоритма.

В сдвиговый регистр *OFB* сначала надо загрузить вектор *IV*. Вектор должен быть уникальным, но сохранять его в тайне не обязательно.

Анализ *OFB* показывает, что *данный режим* целесообразно использовать, только если разрядность обратной связи совпадает с размером блока. В режиме *OFB* над гаммой и текстом выполняется операция *XOR*. Эта гамма в конце

концов повторяется. Существенно, чтобы она не повторялась для одного и того же ключа, в противном случае секретность не обеспечивается ничем.

Если разрядность обратной связи равна размеру блока, блочный шифр работает, выполняя перестановки m-битовых значений (где m – размер блока), и средняя длина цикла составляет $2^T - 1$. Когда разрядность обратной связи n меньше длины блока, средняя длина цикла снижается примерно до $2^{T/2}$.



а *б*

Рис. 1.18. Окна установок режима *OFB* программы *Visual AES*:
а – зашифровка; *б* – расшифровка

***Режим счетчика* (*Counter, CTR*)**

Блочные шифры в режиме счетчика используют последовательность чисел в качестве входов алгоритма (рис. 1.19). Для заполнения регистра используется счетчик, а не вывод алгоритма шифрования. После шифрования каждого блока значение счетчика возрастает на определенную константу, обычно на единицу. Свойства синхронизации и распространения ошибки этого

режима точно такие же, как у режима *OFB*. Режим счетчика устраняет проблему *n*-битового выхода в режиме *OFB*, когда *n* меньше длины блока.

а *б*

Рис. 1.19. Окна установок режима CTR программы *Visual AES*:
а – зашифровка; *б* – расшифровка

К счетчику не предъявляют какие-то особые требования. В частности, он не должен пробегать последовательно все возможные значения. В качестве входа блочного алгоритма можно использовать генераторы случайных чисел.

Выбор режима шифрования

Если необходимы главным образом простота и скорость можно рекомендовать режим *ECB* как самый простой и быстрый режим работы блочного шифра. Помимо уязвимости к вскрытию с повторной передачей, алгоритм в режиме *ECB* проще других для криптоанализа, поэтому для шифрования сообщений использовать режим *ECB* не рекомендуется.

Режим *ECB* удобно использовать для шифрования случайных данных, например, других ключей. Так как данные невелики по размеру и случайны, недостатки режима *ECB* становятся несущественными.

Для шифрования обычного открытого текста лучше всего использовать режимы *CBC*, *CFB*, *OFB* или *CTR*. Выбор режима зависит от потребностей. Для шифрования файлов лучше всего подходит режим *CBC*. Надежность значительно возрастает; и хотя иногда появляются ошибки в символах хранимых данных, почти никогда не бывает сбоев синхронизации.

Visual AES как средство визуализации (*обучения*)

Одной из важных функций системы *Visual AES* можно назвать наличие инструментария визуализации алгоритма *AES*, включающего в себя: графический материал, раскрывающий его суть, средства управления составом раундового преобразования (рис. 1.20), возможность ведения для каждого обрабатываемого блока подробного журнала раундовых преобразований.

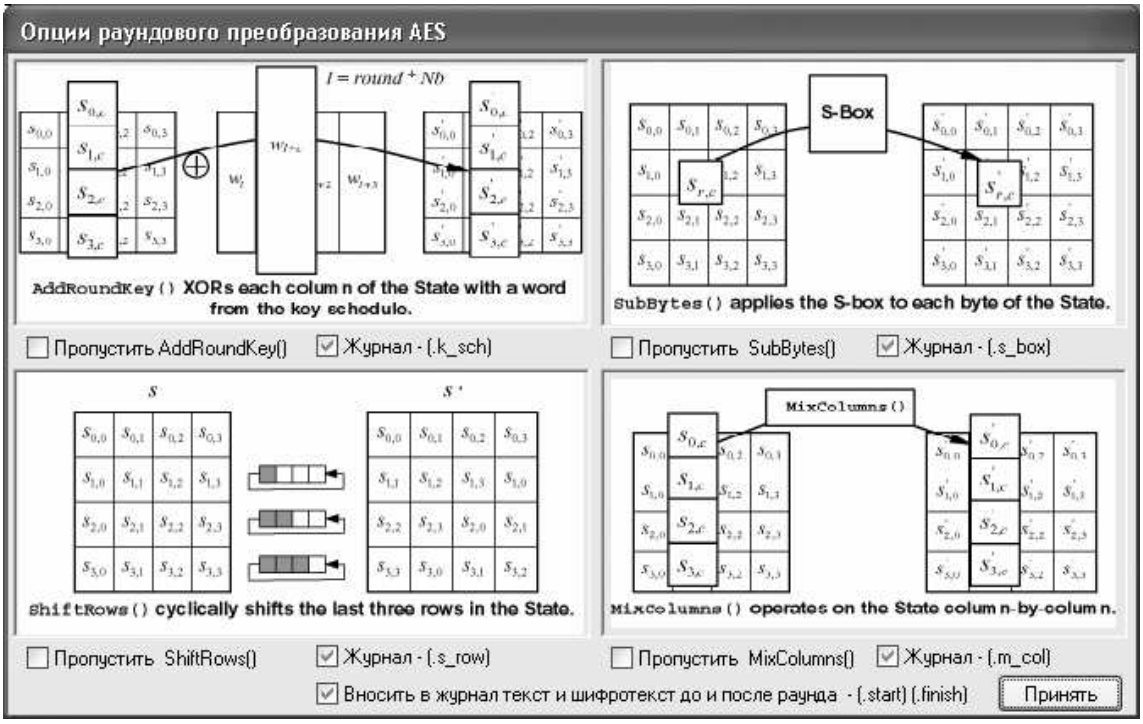


Рис. 1.20. Окно настроек визуализации раундовых преобразований программы *Visual AES*

С помощью установок, изображенных на рис. 1.20, можно управлять наполнением журнала и определять, какие операции будут включены в раундовое преобразование алгоритма *AES*, и соответственно, наблюдать за эволюцией данных в нестандартных условиях.

Журнал раундовых преобразований

Журнал раундовых преобразований посредством описанных выше установок программы *Visual AES* может содержать подробный отчет о трансформации данных каждого шифруемого и дешифруемого блока на всех этапах раундовых преобразований.

Visual AES как средство исследования

Специфической функцией программы *Visual AES* является возможность управления математическим аппаратом раундовых преобразований с целью исследования алгоритма *AES*, а также получения новых средств шифрования на основе методов, заложенных в нем.

Преобразование **SubBytes( )**

На рис. 1.21 изображено окно настроек математических преобразований в рамках раундовой операции *SubBytes*. Операции замены и обратной замены байтов для некоторого a в стандарте *AES* описываются следующими математическими выражениями:

$$\lambda(f) \equiv \left(\left(\left(X^4 + X^3 + X^2 + X + 1\right) \cdot f\right) \bmod \left(X^8 + X^4 + X^3 + X + 1\right) + X^6 + X^5 + X + 1\right) \bmod \left(X^8 + 1\right).$$

$$\lambda^{-1}(f) \equiv \left(\left(\left(X^6 + X^3 + X\right) \cdot f\right) \bmod \left(X^8 + X^4 + X^3 + X + 1\right) + X^2 + 1\right) \bmod \left(X^8 + 1\right).$$

$\sigma(a) = \lambda\left(a^{254}\right)$, $\sigma^{-1}(a) = \left(\lambda^{-1}(a)\right)^{254}$ – мультипликативная инверсия.

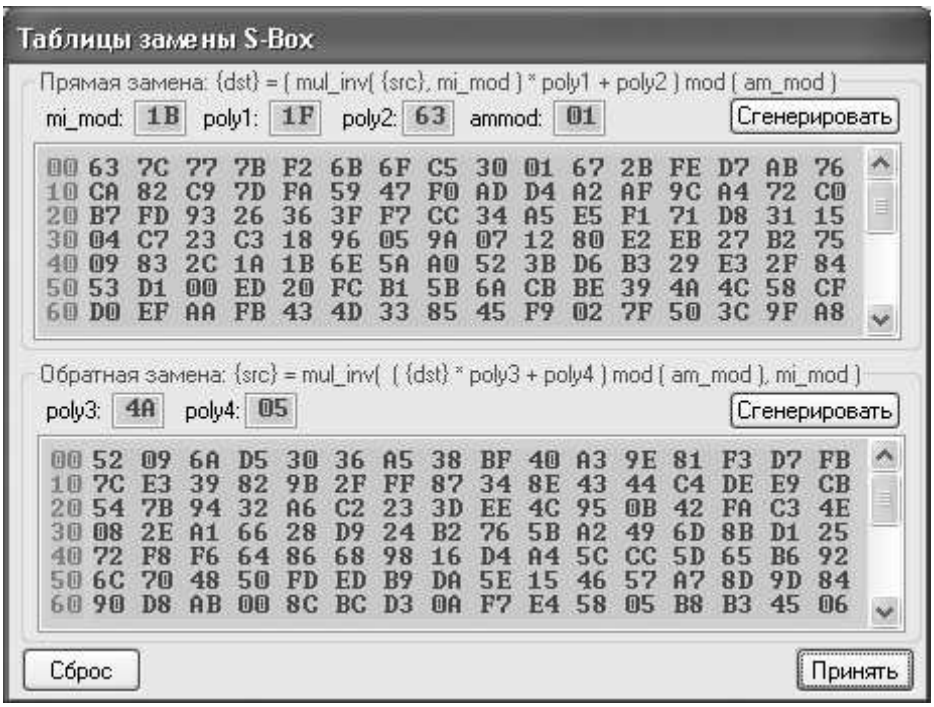


Рис. 1.21. Окно *Visual AES* настроек преобразования *SubBytes* программы *Visual AES*

Преобразование ShiftRows( )

Раундовое преобразование сдвига строк в стандарте *AES* имеет строго определенную конфигурацию, зависящую от размеров ключа и блока. В системе *Visual AES* существует возможность задания произвольных смещений для каждой из строк состояния (рис. 1.22).

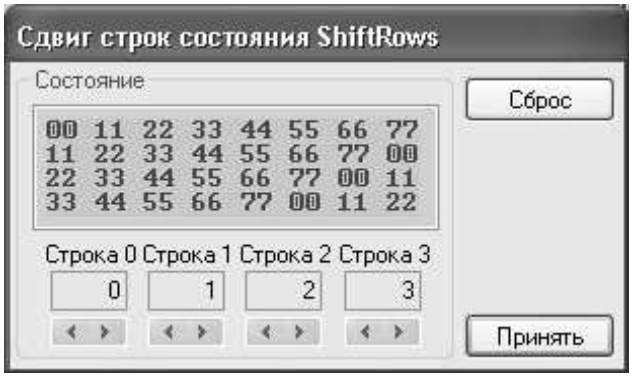


Рис. 1.22. Окно настроек преобразования *ShiftRows* программы *Visual AES*

Преобразование **MixColumns( )**

Преобразование перемешивания строк основывается на матричной операции умножения столбцов состояния, представленных в виде четырехбайтовых слов *g*, на многочлены *c* и *d* для прямого и обратного преобразований соответственно, причем последние сдвигаются на *n* байтов вправо, где *n* – номер столбца состояния, над которым производится операция:

$$c = (X, 1, 1, X+1),\ d = (X^3 + X^2 + X,\ X^3 + X^2 + 1,\ X^3 + X + 1);$$

$$\mu(g) \equiv c \cdot g \bmod(Y^4 + 1),\ \nu(g) \equiv d \cdot g \bmod(Y^4 + 1).$$

На рис. 1.23 показано окно комплекса *Visual AES*, позволяющее задавать значения полиномов *c* и *d* побайтово при сохранении в основной операции правил их сдвига.

Рис. 1.23. Окно настроек преобразования *MixColumns* программы *Visual AES*

Преобразование **AddRoundKey( )**

Специализированное окно трансформаций *AddRoundKey* (рис. 1.24) позволяет изучить содержимое развернутых (раундовых) подключей для каждой из возможных операций шифрования: зашифровывания, инверсного и эквивалентного расшифровывания, – и если это необходимо, произвести копирование этих данных в системный карман *Windows*.

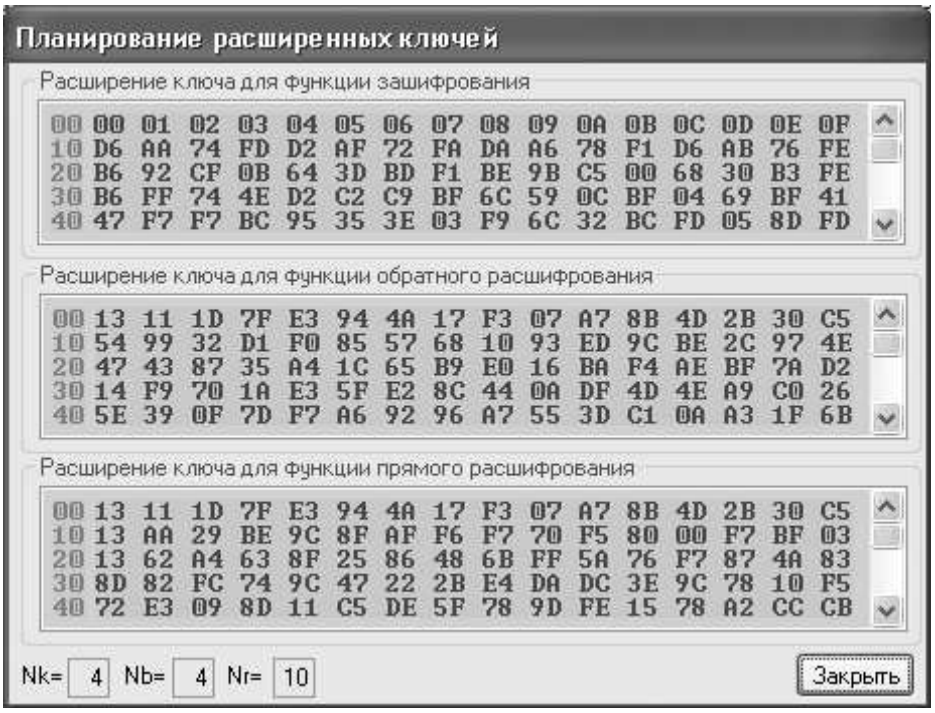


Рис. 1.24. Окно настроек преобразования *AddRoundKey* программы *Visual AES*

Visual AES как средство анализа

Инструментарий «Эксперт», вызываемый из среды *Visual AES*, предназначен для аналитических наблюдений за раундовыми преобразованиями алгоритма *AES* блоков данных. В качестве дополнительного к нему режима реализована единственная эффективная на сегодня атака «Квадрат», представленная авторами шифра в качестве демонстрации стойкости алгоритма (рис. 1.25).

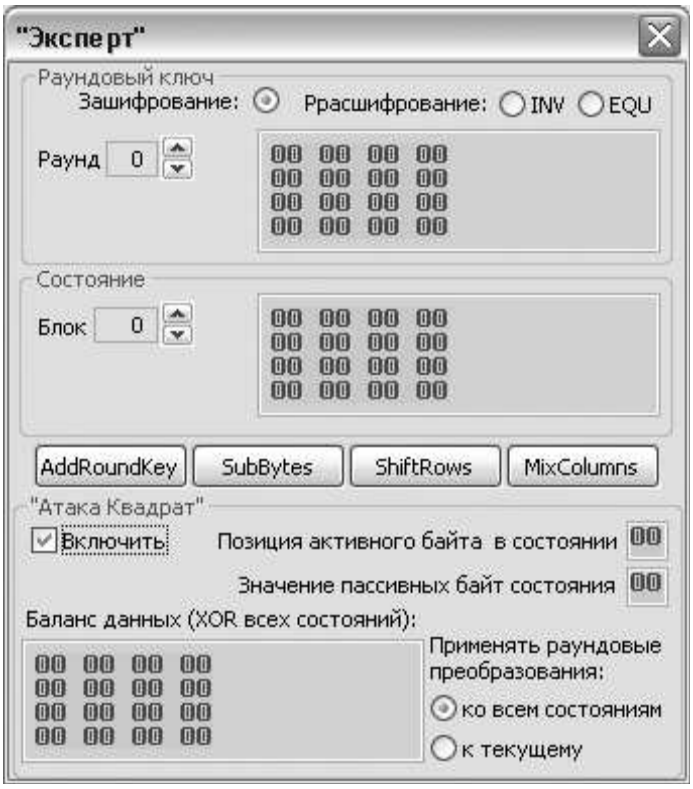


Рис. 1.25. Инструмент пошаговой трассировки системы *Visual AES*

Пошаговая трассировка алгоритма AES

В режиме пошаговой трассировки инструмент «Эксперт» функционирует следующим образом: в зависимости от установок основной программы (ключа, входных файлов и выбранного алгоритма) возможно переключение между блоками (состояниями) соответствующего файла с помощью прокрутки «Блок» и элементами расширенного ключа (прокрутка «Раунд») для последующего применения к состоянию операций раундовых преобразований в произвольном порядке с учетом настройки их математики и наблюдения за ними.

Данные преобразования носят локальный характер и в исходных файлах они не отражаются, т. е. при переключении между блоками история трассировки теряется.

Атака «Квадрат»

Атака «Квадрат» была специально разработана для одноименного шифра *SQUARE* (авторы J. Daemen, L. Knudsen, V. Rijmen). Атака использует при своем проведении байт-ориентированную структуру шифра. Учитывая, что *AES* унаследовал многие свойства шифра *SQUARE*, эта атака применима и к нему.

Атака «Квадрат» основана на возможности свободного подбора атакующим некоторого набора открытых текстов для последующего их зашифрования. Она независима от таблиц замен *S*-блоков, многочлена функции *MixColumns*( ) и способа разворачивания ключа.

После описания базовой атаки на четырех-раундовый *AES* будет показано, как эту атаку можно продлить на пять и даже шесть раундов. Эта атака для шести-раундового шифра *AES* эффективнее, чем полный перебор по всему ключевому пространству. Но уже для семи раундов «Квадрат» становится менее эффективным, чем полный перебор.

Предпосылки

Пусть Λ-набор – такой набор из 256 входных блоков (массивов *State*), каждый из которых имеет байты (назовем их *активными*), значения которых различны для всех 256 блоков. Остальные байты (будем называть их *пассивными*) остаются одинаковыми для всех 256 блоков из Λ-набора. То есть

$$\forall x,\, y \in \Lambda : \begin{cases} x_{ij} \neq y_{ij}, \text{ если байт с номером } ij \text{ активный;} \\ x_{ij} = y_{ij} \text{ в противном случае.} \end{cases}$$

Будучи подвергнутыми обработке функциями *SubBytes*( ) и *AddRoundKey*( ), блоки Λ-набора дадут в результате другой Λ-набор с активными байтами в тех же позициях, что и у исходного. Функция *ShiftRows*( ) сместит эти байты соответственно заданным в ней смещениям в строках массивов *State*. После функции *MixColumns*( ) Λ-набор в общем случае необязательно останется Λ-набором (т. е. результат преобразования может

перестать удовлетворять определению Λ-набора). Но поскольку каждый байт результата функции *MixColumns*( ) является линейной комбинацией (с обратимыми коэффициентами) четырех входных байтов 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 активными байтами.

Базовая атака «Квадрат» на четыре раунда

Рассмотрим Λ-набор, во всех блоках которого активен только один байт. Иначе говоря, значение этого байта различно во всех 256 блоках, а остальные байты одинаковы (скажем, равны нулю). Проследим эволюцию этого байта на протяжении трех раундов. В первом раунде функция *MixColumns*( ) преобразует один активный байт в столбец из четырех активных байт. Во втором раунде эти четыре байта разойдутся по четырем различным столбцам в результате преобразования функцией *ShiftRows*( ). Функция же *MixColumns*( ) следующего, третьего раунда преобразует эти байты в четыре столбца, содержащих активные байты. Этот набор все еще остается Λ-набором до того самого момента, когда он поступает на вход функции *MixColumns*( ) третьего раунда.

Основное свойство Λ-набора, используемое здесь, то, что поразрядная сумма по модулю 2 всех блоков такого набора всегда равна нулю. Действительно, поразрядная сумма неактивных (с одинаковыми значениями) байтов равна нулю по определению операции поразрядного *XOR*, а активные байты, пробегая все 256 значений, также при поразрядном суммировании дадут нуль.

Рассмотрим теперь результат преобразования функцией *MixColumns*( ) в третьем раунде байтов входного массива данных *a* в байты выходного массива

данных b. Покажем, что и в этом случае поразрядная сумма всех блоков выходного набора будет равна нулю, то есть

$$b = \bigoplus_{MixColomns(a),\, a \in \Lambda} \Big| b_{ij} = \bigoplus_{a \in \Lambda} \left(2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j}\right) =$$

$$= \overset{2}{\underset{a \in \Lambda}{\oplus}} a_{ij} \oplus \bigoplus_{a \in \Lambda} a_{(i+2)j} \oplus \bigoplus_{a \in \Lambda} a_{(i+3)j}.$$

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 $SubBytes(\ )$.

Мы предполагаем далее, 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:

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

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 (не дадут при поразрядном суммировании нулевой результат), значит догадка неверна. Перебрав максимум 2^8 варианта байта раундового ключа, мы найдем его истинное значение.

По такому же принципу могут быть определены и другие байты раундового ключа. За счет того, что поиск может производиться отдельно (читай – параллельно) для каждого байта ключа, скорость подбора всего значения раундового ключа весьма велика, а по значению полного раундового ключа, при известном алгоритме его развертывания не составляет труда восстановить начальный ключ шифрования.

Добавление пятого раунда в конец базовой атаки «Квадрат»

Если будет добавлен пятый раунд, то значения b_{ij} придется вычислять уже на основании выходных данных не четвертого, а пятого раунда, и дополнительно кроме байта раундового ключа четвертого раунда перебирать значения четырех байтов столбца раундового ключа для пятого раунда. Только так мы сможем выйти на значения сбалансированных байтов b_{ij} входных данных четвертого раунда. Таким образом, теперь нам нужно перебрать 2^{40} значения: 2^{32} варианта для четырех байтов столбца раундового ключа пятого раунда и для каждого из них 2^{8} варианта для одного байта четвертого раунда. Эту процедуру нужно будет повторить для всех четырех столбцов пятого раунда. Поскольку при подборе «верного» значения байта раундового ключа четвертого раунда количество «неверных» ключей уменьшается в 2^{8} раз, то работая одновременно с пятью Λ-наборами, можно с большой степенью вероятности правильно подобрать все 2^{40} бита. (Кавычки здесь означают то, что *верным* это значение может быть названо лишь с некоторой (но довольно большой) степенью вероятности). Теперь поиск может производиться отдельно (т. е. параллельно) для каждого столбца каждого из пяти Λ-наборов, что опять же гораздо быстрее полного перебора всех возможных значений ключа.

Добавление шестого раунда в начало базовой атаки «Квадрат»

Основная идея заключается в том, чтобы подобрать такой набор блоков открытого текста, который на выходе после первого раунда давал бы Λ-набор с одним активным байтом. Это требует предположения о значении четырех байтов ключа, используемых функцией $AddRoundKey(\,)$ перед первым раундом.

Для того, чтобы на входе второго раунда был только один активный байт, достаточно, чтобы в первом раунде один активный байт оставался на выходе

функции *MixColumns*( ). Это означает, что на входе *MixColumns*( ) первого раунда должен быть такой столбец, байты *a* которого для набора из 256 блоков в результате линейного преобразования

$$b_i = 2a_i \oplus 3a_{i-1} \oplus a_{i-2} \oplus a_{i+3},\ 0 \le i \le 3,$$

где *i* – номер строки. Для одного определенного *i* давали 256 различных значений, в то время как для каждого из остальных трех значений *i* результат этого преобразования должен оставаться постоянным.

Следуя обратно по порядку приложения функций преобразования в первом раунде, к *ShiftRows*( ) данное условие нужно применить к соответственно разнесенным по столбцам четырем байтам. С учетом применения функции *SubBytes*( ) и сложения с предполагаемым значением 4-байтового раундового ключа можно смело составлять уравнения и подбирать нужные значения байтов открытого текста, подаваемых на зашифровку, для последующего анализа результата:

$$b_{ij} = 2SubBytes\left(a_{ij} \oplus k_{ij}\right) \oplus 3SubBytes\left(a_{(i-1)(j-1)} \oplus k_{(i-1)(j-1)}\right) \oplus$$
$$\oplus SubBytes\left(a_{(i-2)(j-2)} \oplus k_{(i-2)(j-2)}\right) \oplus SubBytes\left(a_{(i-3)(j-3)} \oplus k_{(i-3)(j-3)}\right),$$
$$0 \le i,\ j \le 3.$$

Таким образом, получаем следующий алгоритм взлома. Имеем всего 2^{32} различных значения *a* для определенных *i* и *j*. Остальные байты для всех блоков одинаковы (пассивные байты). Предположив значения четырех байтов *k* ключа первого раунда, подбираем (исходя из вышеописанного условия) набор из 256 блоков. Эти 256 блоков станут Λ-набором после первого раунда. К этому Λ-набору применима базовая атака для четырех раундов. Подобранный с ее помощью один байт ключа последнего раунда фиксируется.

Теперь подбирается новый набор из 256 блоков для того же значения четырех байтов *k* ключа первого раунда. Опять осуществляется базовая атака, дающая один байт ключа последнего раунда.

Если после нескольких попыток значение этого байта не меняется, значит мы на верном пути. В противном случае нужно менять предположение о

значении чтырех байтов *k* ключа первого раунда. Такой алгоритм действий достаточно быстро приведет к полному восстановлению всех байтов ключа последнего раунда.

Реализация данной атаки средствами инструмента «Эксперт» выглядит следующим образом: при активации данного режима в динамической памяти создается Λ-набор, между элементами которого можно переключаться с помощью прокрутки «Блок». Параметры Λ-набора можно задать тут же (позиция активного байта в состоянии и значения пассивных байтов). Выбор раундового ключа осуществляется с помощью прокрутки «Раунд». Далее, по нажатии одной из кнопок, соответствующей одной из раундовых трансформаций, она будет применена ко всем элементам Λ-набора. Для контроля баланса данных выводится поразрядная сумма всех состояний Λ-набора.

В четырех-раундовом алгоритме данная реализация инструментария имеет очень высокую эффективность, но с увеличением числа раундов необходимы дополнительные средства для хранения сопроводительной информации и инструментарий статистического анализа.

Инструкция пользователя программного комплекса Visual AES

Программный комплекс *Visual AES* представляет собой многофункциональную систему визуализации, анализа и реализации шифрования данных с использованием алгоритма *AES*.

Основными компонентами системы являются:

• инструментарий шифрования по стандарту *AES*, реализующий зашифровку / расшифровку данных согласно алгоритму *AES* с поддержкой большинства существующих режимов шифрования с обратной связью (*ECB*, *CBC*, *CFB*, *OFB* и *CTR*);

• подсистема визуализации, с помощью соответствующего графического материала, а также специально встроенного инструментария («Журнал» и «Эксперт») раскрывающая суть алгоритма и принципов преобразований, реализованных в нем;

• подсистема анализа, включающая в себя средства управления математическим аппаратом преобразований, проводимых в процессе шифрования, и визуальная реализация известной атаки «Квадрат» на криптоалгоритм *AES*.

Начало работы

После запуска программы *Visual AES* на экране ПК отображается главное окно (рис. 1.26), содержимое которого условно можно разбить на три составляющих:

• настройка алгоритма («Ключ шифрования», «Преобразования»);

• настройка режима работы («Ключ», «Блок», «Режим», «Действие»);

• средства трассировки («Журнал трассировки», «Эксперт»).

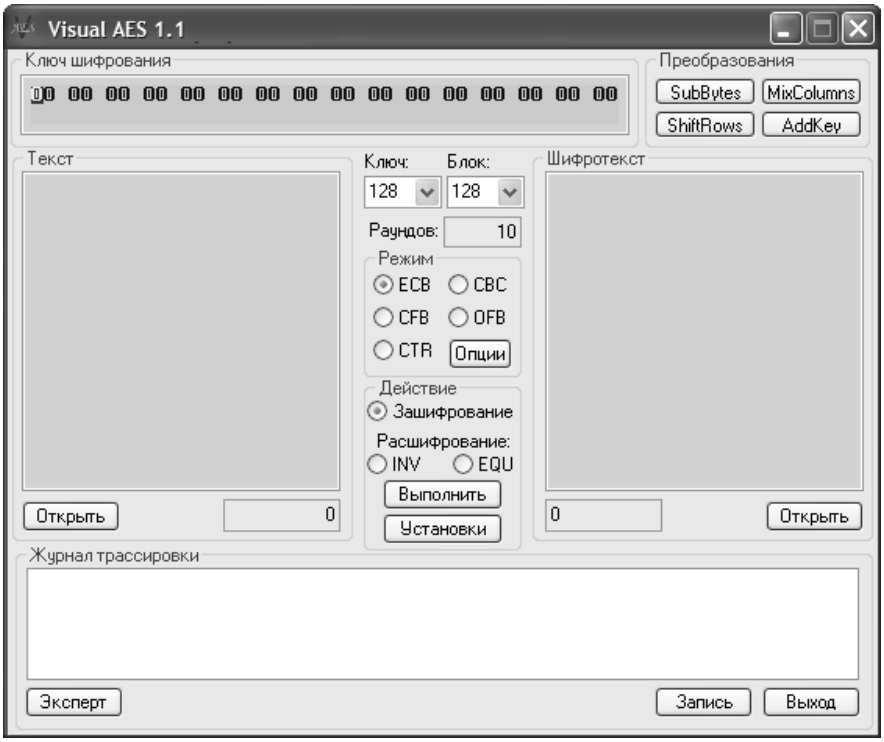


Рис. 1.26. Главное окно программы *Visual AES*

Поля «Текст» и «Шифротекст» предназначены, соответственно, для отображения содержимого открытого и зашифрованного входных файлов (все исходные данные, параметры и результаты в программе представляются в шестнадцатеричном виде).

Для начала работы необходимо:

1. Выбрать в полях «Ключ:» и «Блок:» требующиеся размеры для ключа и блока (рис. 1. 27).

Рис. 1.27. Выбор размеров для ключа и блока

2. Нажатием кнопок «Открыть» выбрать рабочие файлы на одном из носителей, представленных на данном ПК (условиями, накладываемыми блочным шифром *AES*, их длины будут выровнены между собой на границу размера блока – дополнительным байтам присваивается нулевое значение). В случае необходимости создания нового файла следует в поле «Имя файла» ввести имя несуществующего в текущей директории файла и в следующем задать его начальный размер в диалоговом окне (рис. 1.28).

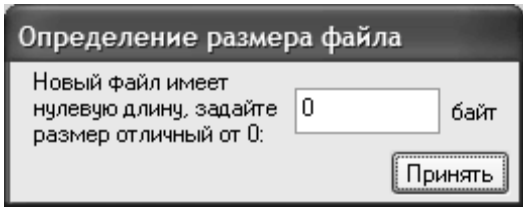


Рис. 1.28. Определение размера нового файла

3. Задать значение ключа шифрования в поле «Ключ шифрования» (рис. 1.29).

В результате описанных выше шагов система становится готовой к работе. Далее в зависимости от поставленной задачи выбирается определенный сценарий: шифрование, обучение, исследование, анализ.

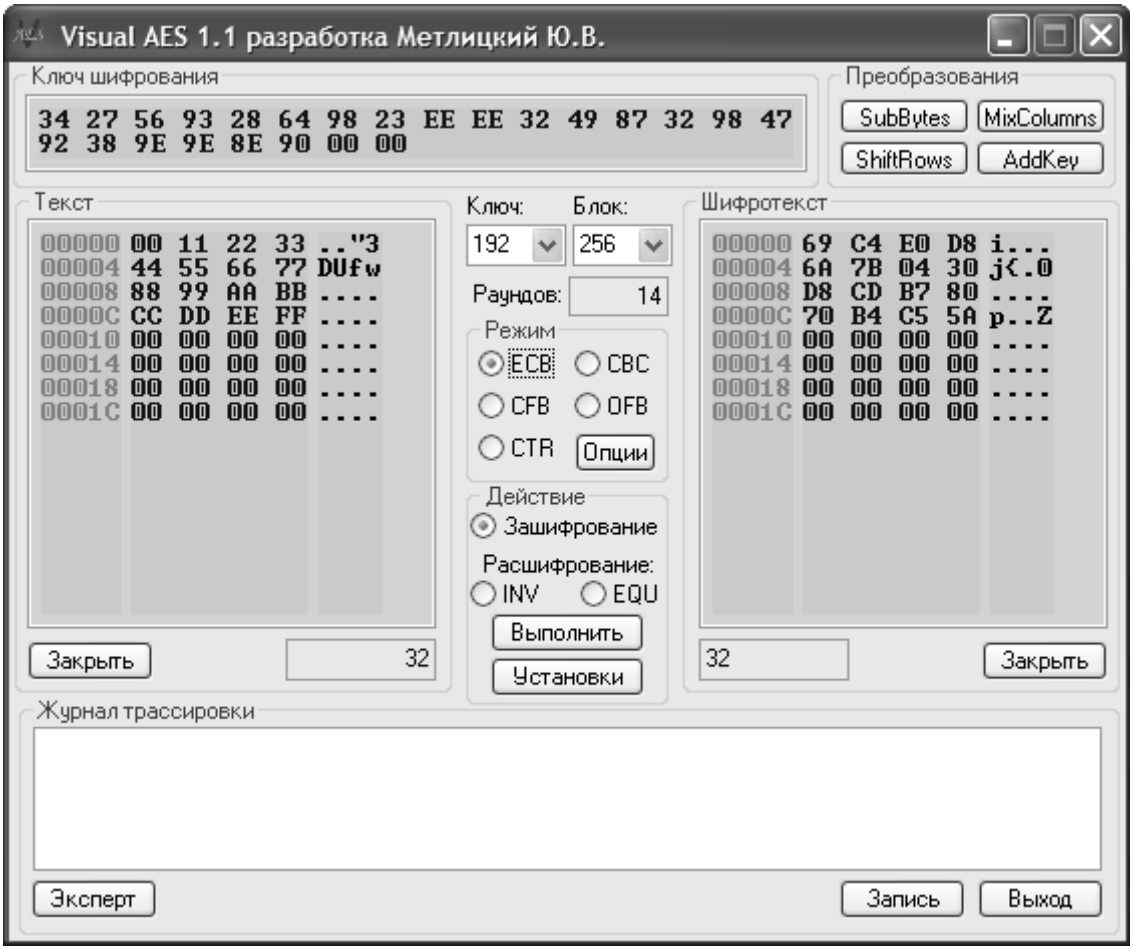


Рис. 1.29. Задание значения ключа шифрования

Шифрование с помощью Visual AES

Зашифровка или расшифровка выполняются над выбранными исходными файлами: если выполняется операция зашифровки, то содержимое окна «Текст» (файла предназначенного для зашифровки) преобразуется согласно алгоритму *AES* и помещается в окно «Шифротекст», и соответственно в файл шифротекста. В случае операции расшифровки выполняется обратное

действие: файл из окна «Шифротекст» подвергается операции расшифровки, а результат передается в окно «Текст» (файл открытого текста).

Для выполнения описанных выше операций необходимо при условии уже определенных ключа, его размера и размера блока выполнить следующие операции:

1. Выбрать необходимый режим из предлагаемого перечня режимов шифрования с обратной связью в группе «Режим» (*ECB*, *CBC*, *CFB*, *OFB* и *CTR*) (рис. 1.30).

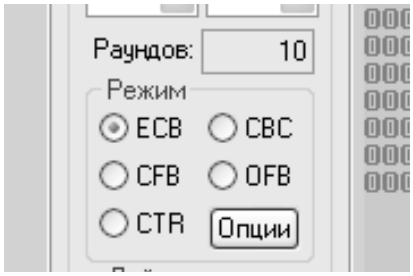


Рис. 1.30. Выбор режима шифрования

При переключении на одну из радиокнопок кнопка «Опции» автоматически перестраивается на управление установками именно того режима, который активизируется радиокнопкой. При нажатии на кнопку «Опции» выводится одно из окон установок, соответствующее выбранному режиму (см. рис. 1.15 – 1.19).

Поле «Вектор инициализации» представляет блок случайных данных, инициализирующих процедуру сцепления блоков начальным значением. Кнопка «случайный ВИ» позволяет присвоить ему результат вызова генератора псевдослучайной последовательности.

Опция «Журнал» позволяет в журнал трассировки помещать соответствующие сообщения режима (номер блока, значения каждого входного и выходного блоков).

Поле «Число битов в операции *XOR*» задает размер шифруемой битовой последовательности.

В каждом окне установок режима представлена графическая диаграмма его работы в зависимости от направления шифрования (зашифровка или расшифровка), а также соответствующие изменяемые параметры.

2. Задать одной из радиокнопок в группе «Действие» вид выполняемой операции: «Зашифрование» или «Расшифрование» (рис.1.31). В *Visual AES* реализованы два режима расшифрования – инверсный («*INV*») и эквивалентный («*EQU*»).

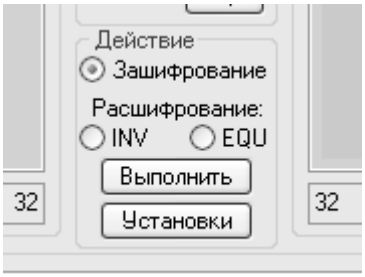


Рис. 1.31. Выбор операции зашифровки или расшифровки

3. После нажатия кнопки «Выполнить» операция шифрования будет выполнена, ее результат будет отображен в окне назначения и записан в соответствующий файл.

Обучение с помощью Visual AES

В процессе выполнения операций шифрования с целью изучения структуры алгоритма система *Visual AES* позволяет вести подробный журнал раундовых трансформаций, в котором по выбору пользователя возможно фиксирование всех состояний, которые проходит блок данных при зашифровке или расшифровке. «Журнал трассировки» расположен в нижней части основного окна программы (рис. 1.32), с помощью полосы прокрутки возможно перемещение по его записям вверх и вниз. В случае выбора некоторых установок, определяющих содержимое журнала, при нажатии кнопки «Выполнить» группы «Действие», журнал очищается и вновь последовательно заполняется результатами производимой операции (зашифровка/расшифровка).

Кнопка «Запись» позволяет сохранить его содержимое на внешний носитель для дальнейшего исследования.

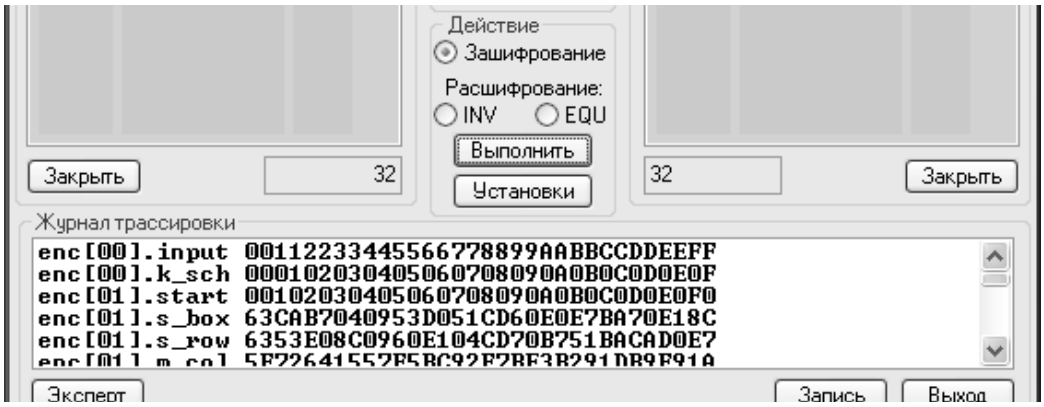


Рис. 1.32. Журнал трассировки

Обсудим все возможные установки журнала. Они доступны по нажатию кнопки «Установки» группы «Действие».

Кнопка «Принять» завершает работу с данным окном.

Согласно официальной документации на стандарт *AES* в окне «Опции раундового преобразования *AES*» перечислены последовательно исполняемые раундовые операции и изображены их диаграммы (рис. 1.33).

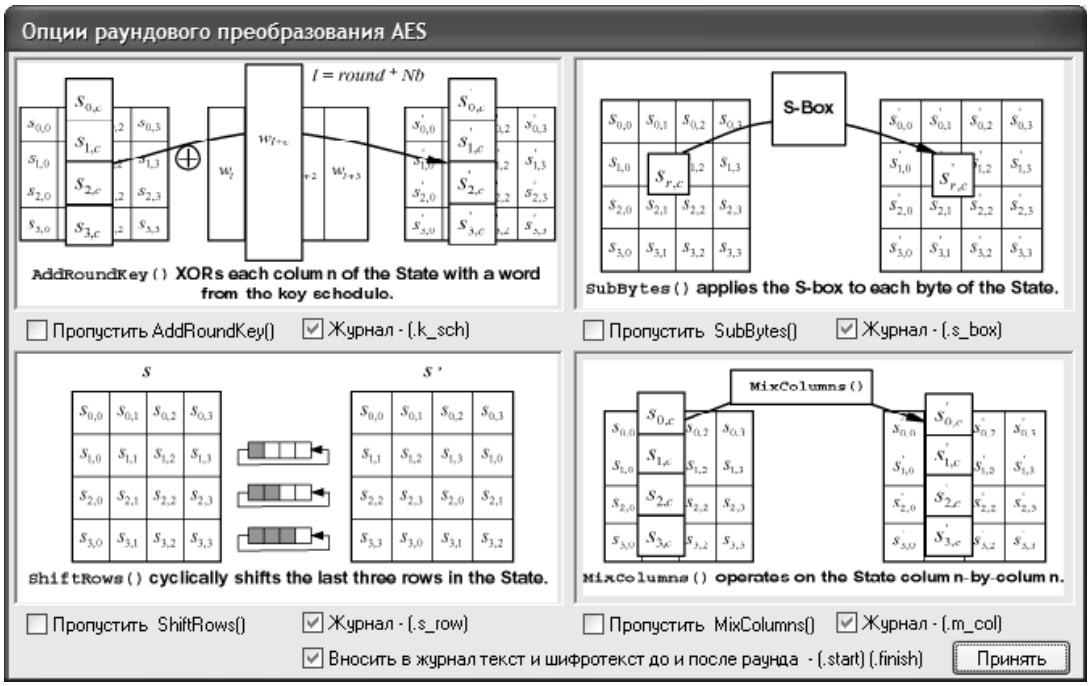


Рис. 1. 33. Окно «Опции раундового преобразования *AES*»

Под каждой из них размещены по паре элементов выбора, позволяющих соответственно включить или исключить из раундового преобразования данную операцию вообще или включить или исключить помещение в журнал трассировки промежуточных результатов последней.

Возможность включения или исключения самой раундовой операции из алгоритма будет обсуждаться в разделе «Исследование с помощью *Visual AES*».

Строка журнала

enc[04]. start FA636A2825B339C940668A3157244D17

обозначает, что в операции зашифровки («*enc*», возможны «*inv*» и «*equ*») на старте («.*start*») раунда номер 4 содержимое блока шифруемых данных выглядело следующим образом: FA636A2825B339C940668A3157244D17.

Ниже приводится перечень обозначений всех возможных элементов журнала:

.*input*[*xxx*] – данные перед операцией шифрования;

.*start*[*xxx*] – блок на начало раунда;

.*k_sch*[*xxx*] – значение раундового ключа;

.*k_add*[*xxx*] – блок после добавления раундового ключа;

.*s_box*[*xxx*] – блок после операций *(Inv)SubBytes*;

.*s_row*[*xxx*] – блок после операций *(Inv)ShiftRows*;

.*m_col*[*xxx*] – блок после операций *(Inv)MixColumns*;

.*final*[*xxx*] – результат шифрования.

Исследование с помощью Visual AES

Для исследования математического аппарата алгоритма *AES* в системе *Visual AES* предусмотрен инструмент «Преобразования», основная задача которого – предоставить пользователю возможность управления параметрами преобразования байтов данных шифруемых блоков.

Согласно приводимой документации на стандарт шифрования *AES*, каждое раундовое преобразование выполняет над блоком данных определенные

операции, схематически представленные на диаграммах окна «Опции раундового преобразования *AES*».

Преобразование SubBytes

(кнопка «*SubBytes*» группы «Преобразования»)

В рамках данного преобразования на каждом раунде все байты блока данных подвергаются замене: при зашифровке – согласно таблице «прямой замены», при расшифровке – согласно таблице «обратной замены». Элементы обеих таблиц имеют математическую зависимость с заменяемыми байтами, изображенную в окне «Таблицы замены *S-Box*» в виде формул (рис. 1.34). Параметрами этой зависимости можно управлять. После задания им новых значений необходимо нажать на кнопку «Сгенерировать» и программа создаст новую таблицу замен (прямой или обратной).

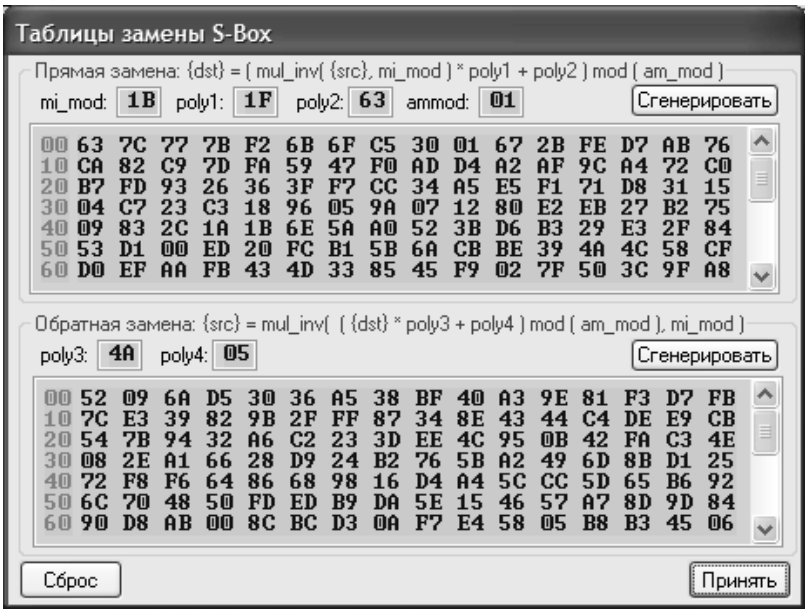


Рис. 1.34. Окно «Таблицы замены *S-Box*»

Для возврата в состояние, определяемое стандартом *AES*, требуется нажатие кнопки «Сброс».

Кнопка «Принять» завершает работы с данным окном.

Условные обозначения:

{*dst*} – результат прямого преобразования *S-Box* байта {*src*};

{*src*} – результат обратного преобразования *S-Box* байта {*dst*};

mi_mod – модуль, по которому вычисляется мультипликативная инверсия в поле $GF(2^8)$;

*poly*1 – произвольный многочлен в поле $GF(2^8)$, участвующий в операции умножения;

*poly*2 – произвольный многочлен в поле $GF(2^8)$, участвующий в операции сложения;

am_mod – модуль, по которому вычисляются значения преобразования замены байт в поле $GF(2^8)$;

mul_inv( ) – операция нахождения мультипликативной инверсии в поле $GF(2^8)$;

*poly*3 – произвольный многочлен в поле $GF(2^8)$, участвующий в операции умножения, обращающий операцию прямой замены;

*poly*4 – произвольный многочлен в поле $GF(2^8)$, участвующий в операции сложения, обращающий операцию прямой замены.

Преобразование ShiftRows

(кнопка «*ShiftRows*» группы «Преобразования»)

Следующим преобразованием в раунде является сдвиг строк блока на определенное число позиций – влево при зашифровки и вправо при расшифровки (по стандарту это 0, 1, 2, 3 позиции по порядку следования строк сверху вниз). С помощью стрелок возможна перестройка данных смещений в окне «Сдвиг строк состояния *ShiftRows*» для операции зашифровки. Для операции расшифрования *Visual AES* выбирает значения, противоположные заданным (рис. 1.35).

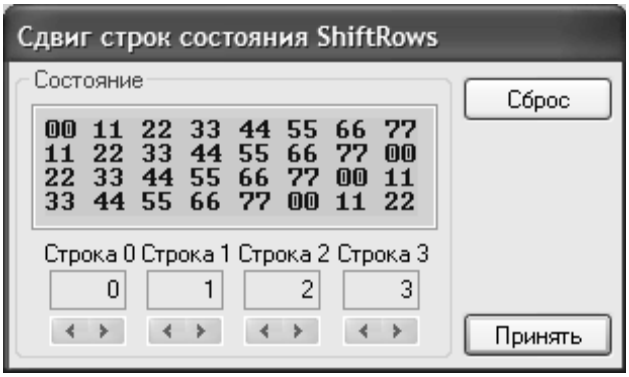


Рис. 1.35. Окно сдвига строк состояния *ShiftRows*

Восстановление исходных значений происходит при нажатии кнопки «Сброс». Кнопка «Принять» завершает работы с данным окном.

Преобразование MixColumns

(кнопка «*MixColumns*» группы «Преобразования»)

Операция *MixColumns* выполняет умножение каждого столбца в поле $GF\left(2^{32}\right)$ на многочлены, представленные в окне «Полиномы операции *MixColumns*» (рис. 1.36), по подулю *am_mod* (см. выше). Побайтовое их представление для прямого $c(x)$ и обратного $d(x)$ можно модифицировать для получения новых вариантов алгоритма.

Кнопка «Сброс» возвращает значения, определенные в стандарте шифрования *AES*.

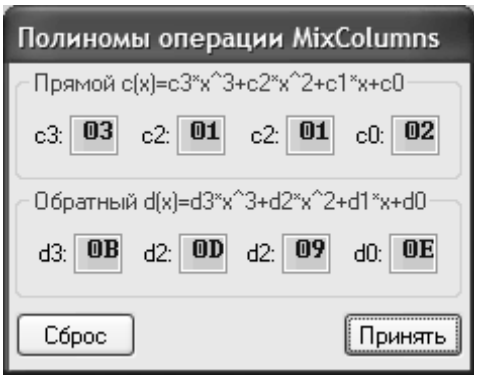


Рис. 1.36. Окно «Полиномы операции *MixColumns*»

Кнопка «Принять» завершает работу с данным окном.

Преобразование AddRoundKey

(кнопка «*AddRoundKey*» группы «Преобразования»)

Операция добавления раундового ключа требует предварительной генерации «графика» раудовых ключей по заданному ключу с привлечением специального алгоритма планирования. В окне «Планирование расширенных ключей» (рис. 1.37) в зависимости от выбранных размеров блока и ключа в основном окне программы отображаются «Расширение ключа для функции зашифрования», «Расширение ключа для функции обратного расшифрования» и «Расширение ключа для функции прямого расшифрования» – значения всех раундовых ключей для каждой из операций. В нижней части окна также выводятся характеристики выбранного алгоритма, а именно: *Nk* – размер ключа в 32-битовых словах, *Nb* – размер блока в 32-битовых словах и *Nк* – число раундов шифрования.

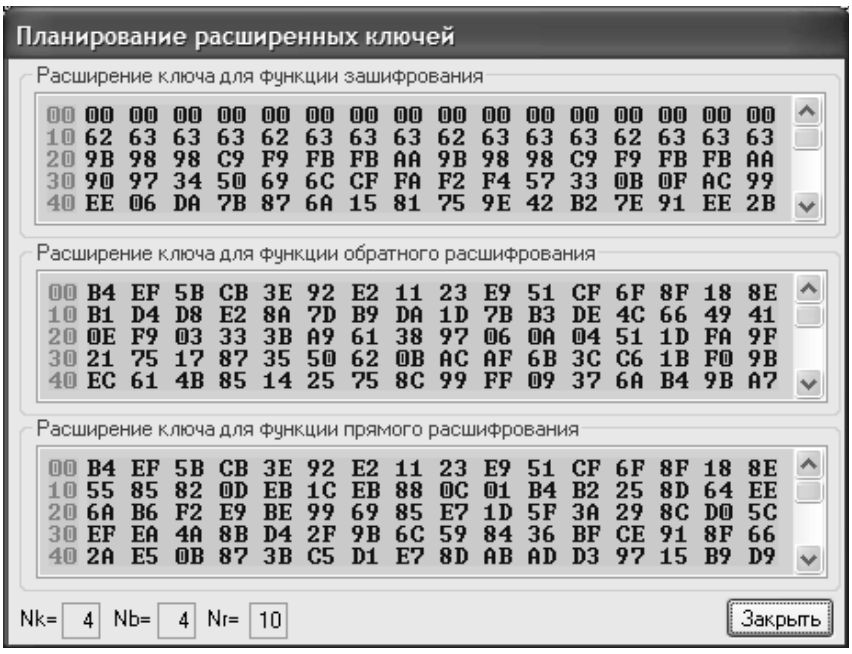


Рис. 1.37. Окно планирования расширенных ключей

Кнопка «Закрыть» завершает работу с данным окном.

Анализ с помощью Visual AES

Инструментарий «Эксперт» (рис. 1.38), доступный из основного окна программы *Visual AES*, предназначен для детального анализа данных – результатов раундовых преобразований шифра *AES*. В нем реализовано два режима работы: пошаговая трассировка раундовых трансформаций (основной режим) и атака «Квадрат» (активизирующаяся выбором кнопки «Включить» группы «Атака Квадрат»).

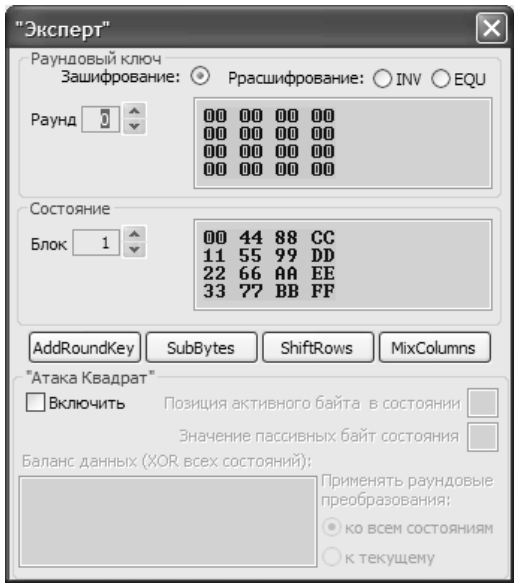


Рис. 1.38. Инструментарий «Эксперт»

В режиме трассировки последовательность работы следующая.

При переходе в режим «Эксперт» происходит копирование первого блока данных из одного из исходных файлов (в зависимости от выбора операции в группе «Действие» основного окна программы), а при его отсутствии создается временный блок с нулевым заполнением.

С помощью прокрутки «Блок» группы «Состояние» возможно переключение между смежными блоками данных исходных файлов. Прокрутка «Раунд» осуществляет выбор раундового ключа согласно его порядковому номеру. Радиокнопки «Зашифрование» и «Расшифрование» определяют применение раудовых операций. При нажатии на одну из кнопок

«*AddRoundKey*», «*SubBytes*», «*ShiftRows*» или «*MixColumns*» к текущему блоку данных (состоянию) применяется соответствующая трансформация и на экран выводится ее результат.

В режиме «Атака Квадрат» (рис. 1.39) последовательность работы следующая.

Правила управления результатами раундовых трансформаций такие же, как и в предыдущем режиме. Отличие состоит во входных данных (состояниях). Атака квадрат подробно описывается в «*AES Proposal: Rijndael*» *Joan Daemen, Vincent Rijmen*.

Для проведения атаки «Квадрат» создается набор из 256 блоков (состояний), заполняющихся значениями «Пассивных байтов», а в позиции «Активного байта» каждого состояния размещается порядковый номер (0-255) состояния.

Поле «Баланс данных» отображает результат сложения по модулю 2 всех состояний после каждой «кнопочной» раундовой трансформации (в атаке эта возможность требуется для контроля изменений в состояниях).

Применение любой раундовой операции можно ограничить либо текущим отображаемым в окне блоком, либо одновременным ее применением ко всем элементам набора (радиокнопки «Применять раундовые преобразования» группы «Атака Квадрат»).

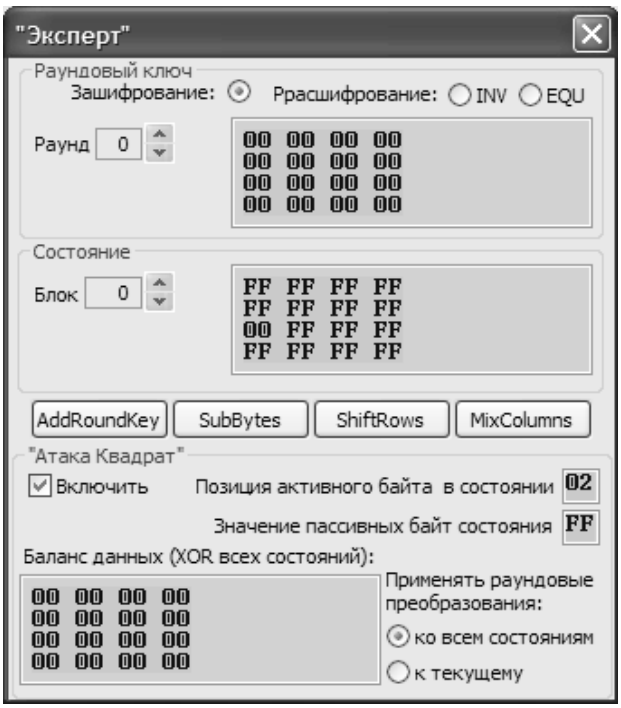


Рис. 1.39. Режимы «Атака Квадрат»

Завершение работы Visual AES

Для завершения работы системы *Visual AES* необходимо в основном окне программы нажать на кнопку «Выход» и подтвердить свое решение утвердительным ответом в окне «Выход» (рис. 1.40).

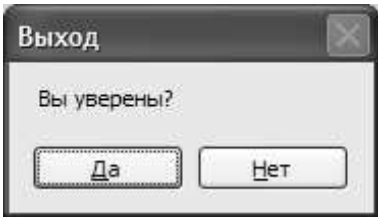


Рис.1.40. Окно «Выход»

Содержание отчета

Отчет должен содержать ответы на контрольные вопросы, условие задания и результат выполнения задания соответствующего варианта.

Контрольные вопросы

По разделу «Структура шифра AES»:

1. Каковы основные параметры шифра *AES*?
2. В чем суть архитектуры «Квадрат»?
3. Какие функции зашифровки используются в алгоритме *AES*?
4. Какие функции расшифровки используются в алгоритме *AES*?
5. Опишите принципы работы прямого и обратного расшифровываний?
6. В чем суть алгоритма выработки ключей?
7. Что такое раундовое преобразование?
8. Как производится сдвиг строк в алгоритме *AES*?
9. Как производится перемешивание столбцов в алгоритме *AES*?
10. Как производится добавление раундового ключа?
11. Каковы основные особенности *AES*?

По разделу «Система Visual AES»:

1. Каковы основные компоненты системы *Visual AES*?
2. Что такое режим электронной кодовой книги, каковы его преимущества и недостатки перед другими режимами кодирования?
3. В чем заключается режим сцепления блоков шифротекста?
4. В чем суть режима обратной связи по шифротексту, каковы его преимущества и недостатки?
5. В чем заключается режим обратной связи по выходу?
6. Что такое режим счетчика?
7. Что называется вектором инициализации?
8. Как выбирать (применять) режим шифрования?
9. Какие средства визуализации имеет система *Visual AES*?
10. Что такое журнал раундовых преобразований?
11. Как можно применять систему *Visual AES* в качестве средства исследования?

12. Как можно применять систему *Visual AES* в качестве средства анализа?

13. Что такое атака «Квадрат»?

Варианты заданий по программному комплексу Visual AES

Зашифровать запись, состоящую из слов: номер группы, фамилия, имя, отчество; первое и второе слова разделены двумя пробелами, остальные – одним пробелом; в конце записи – точка. Ключ и вектор инициализации придумать самостоятельно. Длина ключа – 128 , длина блока – 128.

Зашифровку и расшифровку провести в пяти режимах. Заполнить пошагово журнал трассировок и объяснить содержание каждой его записи.

Порядок выполнения:

1. Зашифровать исходный файл и распечатать оба файла и параметры кодировки.

2. Расшифровать зашифрованный файл и вывести на печать оба файла, указать параметры кодировки и используемые опции.

3. Распечатать журнал раундовых преобразований.

4. Произвести анализ с помощью пошаговой трассировки раундовых трансформаций; результаты распечатать.

5. Выполнить атаку «Квадрат» и распечатать шаги анализа.

Лабораторная работа № 9

АППАРАТНО-ПРОГРАММНЫЕ СРЕДСТВА ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

Цель работы: ознакомление с устройством и принципами работы отечественных и российских аппаратно-программных средств защиты компьютерной информации.

Виды и классификация аппаратно-программных средств защиты компьютерной информации

Аппаратно-программные средства, обеспечивающие повышенный уровень защиты, можно разбить на пять основных групп (рис. 2.1).

Рис. 2.1. Аппаратно-программные средства защиты информации

Первую группу образуют ***системы идентификации и аутентификации пользователей***. Такие системы применяются для

ограничения доступа случайных и незаконных пользователей к ресурсам компьютерной системы. Общий алгоритм работы этих систем заключается в том, чтобы получить от пользователя информацию, удостоверяющую его личность, проверить ее подлинность и затем предоставить (или не предоставить) этому пользователю возможность работы с системой.

При построении подобных систем возникает проблема выбора информации, на основе которой осуществляются процедуры идентификации и аутентификации пользователя. Можно выделить следующие типы:

1) секретная информация, которой обладает пользователь (пароль, персональный идентификатор, секретный ключ и т. п.); эту информацию пользователь должен запомнить или же могут быть применены специальные средства хранения этой информации;

2) физиологические параметры человека (отпечатки пальцев, рисунок радужной оболочки глаза и т.п.) или особенности поведения человека (особенности работы на клавиатуре и т. п.).

Системы идентификации, основанные на первом типе информации, принято считать ***традиционными.*** Системы идентификации, использующие второй тип информации, называются ***биометрическими***.

Вторую группу средств, обеспечивающих повышенный уровень защиты, составляют ***системы шифрования дисковых данных***. Основная задача, решаемая такими системами, состоит в защите от несанкционированного использования данных, расположенных на магнитных носителях.

Обеспечение конфиденциальности данных, располагаемых на магнитных носителях, осуществляется путем их шифрования с использованием симметричных алгоритмов шифрования. Основным классификационным признаком для комплексов шифрования служит уровень их встраивания в компьютерную систему.

Работа прикладных программ с дисковыми накопителями состоит из двух этапов: логического и физического.

Логический этап соответствует уровню взаимодействия прикладной программы с операционной системой (например, вызов сервисных функций чтения/записи данных). На этом уровне основным объектом является файл.

Физический этап соответствует уровню взаимодействия операционной системы и аппаратуры. В качестве объектов этого уровня выступают структуры физической организации данных – сектора диска.

В результате системы шифрования данных могут осуществлять криптографические преобразования данных на уровне файлов (защищаются отдельные файлы) и на уровне дисков (защищаются диски целиком).

К программам первого типа можно отнести архиваторы типа *arj*, которые позволяют использовать криптографические методы для защиты архивных файлов. Примером систем второго типа может служить программа шифрования *Diskreet*, входящая в состав популярного программного пакета *Norton Utilities*.

Другим классификационным признаком систем шифрования дисковых данных является способ их функционирования. **По способу функционирования** системы шифрования дисковых данных делят на два класса:

1) системы «прозрачного» шифрования;

2) системы, специально вызываемые для осуществления шифрования.

В системах «прозрачного» шифрования (шифрования «на лету») криптографические преобразования осуществляются в режиме реального времени, незаметно для пользователя. Например, пользователь записывает подготовленный в текстовом редакторе документ на защищаемый диск, а система защиты в процессе записи выполняет его шифрование.

Системы второго класса обычно представляют собой утилиты, которые необходимо специально вызывать для выполнения шифрования. К ним относятся, например, архиваторы со встроенными средствами парольной защиты.

К **третьей группе** средств, обеспечивающих повышенный уровень защиты, относятся ***системы шифрования данных, передаваемых по компьютерным сетям***. Различают два основных способа шифрования: канальное шифрование и оконечное (абонентское) шифрование.

В случае *канального шифрования* защищается вся передаваемая по каналу связи информация, включая служебную. Соответствующие процедуры шифрования реализуются с помощью протокола канального уровня семиуровневой эталонной модели взаимодействия открытых систем *OSI* (*Open System Interconnection*). Этот способ шифрования обладает следующим достоинством – встраивание процедур шифрования на канальный уровень позволяет использовать аппаратные средства, что способствует повышению производительности системы.

Однако, у данного подхода имеются существенные недостатки:

• шифрованию на данном уровне подлежит вся информация, включая служебные данные транспортных протоколов; это осложняет механизм маршрутизации сетевых пакетов и требует расшифровки данных в устройствах промежуточной коммутации (шлюзах, ретрансляторах и т. п.);

• шифрование служебной информации, неизбежное на данном уровне, может привести к появлению статистических закономерностей в шифрованных данных; это влияет на надежность защиты и накладывает ограничения на использование криптографических алгоритмов.

Оконечное* (*абонентское*) *шифрование позволяет обеспечить конфиденциальность данных, передаваемых между двумя прикладными объектами (абонентами). Оконечное шифрование реализуется с помощью протокола прикладного или представительного уровня эталонной модели *OSI*. В этом случае защищенным оказывается только содержание сообщения, вся служебная информация остается открытой. Данный способ позволяет избежать проблем, связанных с шифрованием служебной информации, но при этом возникают другие проблемы. В частности, злоумышленник, имеющий доступ к

каналам связи компьютерной сети, получает возможность анализировать информацию о структуре обмена сообщениями, например, об отправителе и получателе, о времени и условиях передачи данных, а также об объеме передаваемых данных.

Четвертую группу средств защиты составляют ***системы аутентификации электронных данных***. При обмене электронными данными по сетям связи возникает проблема аутентификации автора документа и самого документа, т. е. установления подлинности автора и проверки отсутствия изменений в полученном документе. Для аутентификации электронных данных применяют код аутентификации сообщения (имитовставку) или электронную цифровую подпись. При формировании кода аутентификации сообщения и электронной цифровой подписи используются разные типы систем шифрования.

Имитовставка вырабатывается из открытых данных посредством специального преобразования шифрования с использованием секретного ключа и передается по каналу связи в конце зашифрованных данных. Имитовставка проверяется получателем сообщения, владеющим секретным ключом, путем повторения процедуры, выполненной ранее отправителем, над полученными открытыми данными.

Электронная цифровая подпись (ЭЦП) представляет собой относительно небольшое количество дополнительной аутентифицирующей цифровой информации, передаваемой вместе с подписываемым текстом. Для реализации ЭЦП используются принципы асимметричного шифрования. Система ЭЦП включает процедуру формирования цифровой подписи отправителем с использованием секретного ключа отправителя и процедуру проверки подписи получателем с использованием открытого ключа отправителя.

Пятую группу средств образуют ***средства управления ключевой информацией***. Под ключевой информацией понимается совокупность всех

используемых в компьютерной системе или сети криптографических ключей. Безопасность любого криптографического алгоритма определяется используемыми криптографическими ключами. В случае ненадежного управления ключами злоумышленник может завладеть ключевой информацией и получить полный доступ ко всей информации в компьютерной системе или сети.

Основным классификационным признаком средств управления ключевой информацией является вид функции управления ключами. Различают следующие основные **виды функций управления ключами**: генерация ключей, хранение ключей и распределение ключей.

Способы ***генерации ключей*** различаются для симметричных и асимметричных криптосистем. Для генерации ключей симметричных криптосистем используются аппаратные и программные средства генерации случайных чисел, в частности схемы с применением блочного симметричного алгоритма шифрования. Генерация ключей для асимметричных криптосистем представляет существенно более сложную задачу в связи с необходимостью получения ключей с определенными математическими свойствами.

Функция ***хранения ключей*** предполагает организацию безопасного хранения, учета и удаления ключей. Для обеспечения безопасного хранения и передачи ключей применяют их шифрование с помощью других ключей. Такой подход приводит к *концепции иерархии ключей*. В иерархию ключей обычно входят главный ключ (мастер-ключ), ключ шифрования ключей и ключ шифрования данных. Следует отметить, что генерация и хранение мастер-ключей являются критическими вопросами криптографической защиты.

Распределение ключей является самым ответственным процессом в управлении ключами. Этот процесс должен гарантировать скрытность распределяемых ключей, а также оперативность и точность их распределения. Различают **два основных способа распределения ключей** между пользователями компьютерной сети:

• применение одного или нескольких центров распределения ключей;

• прямой обмен сеансовыми ключами между пользователями.

Аппаратно-программные средства криптографической защиты информации серии Криптон

Концептуальный подход фирмы АНКАД к защите информации в компьютерных системах и сетях

Полностью контролируемые компьютерные системы. Любая компьютерная система (КС) использует стандартное и специализированное оборудование и программное обеспечение, выполняющее определенный набор функций: аутентификацию пользователя, разграничение доступа к информации, обеспечение целостности информации и ее защиты от уничтожения, шифрование и электронную цифровую подпись и др.

Целостность и ограничение доступа к информации обеспечиваются специализированными компонентами системы, использующими криптографические методы защиты.

Надежность защиты информации в компьютерной системе определяется:

- конкретным перечнем и свойствами функций КС;
- используемыми в функциях КС методами;
- способом реализации функций КС.

Перечень используемых функций соответствует классу защищенности, присвоенному КС в процессе сертификации, и одинаков для систем одного класса. Поэтому при рассмотрении конкретной КС следует обратить внимание на используемые методы и способ реализации наиболее важных функций: аутентификацию и проверку целостности системы. Здесь следует отдать предпочтение криптографическим методам: шифрования (ГОСТ 28147 – 89), электронной цифровой подписи (ГОСТ Р 34.10 – 94) и функции хэширования

(ГОСТР 34.11 – 94), надежность которых подтверждена соответствующими государственными организациями России.

Программная реализация функций КС. Большинство функций современных КС реализованы в виде программ, поддержание целостности которых при запуске системы и особенно в процессе ее функционирования является трудной задачей.

Проверка целостности одних программ с помощью других не является надежной. Необходимо четко представлять, каким образом обеспечивается целостность собственно программы проверки целостности. Если обе программы находятся на одних и тех же носителях, доверять результатам такой проверки нельзя. В связи с этим к программным системам защиты от несанкционированного доступа (НСД) следует относиться с особой осторожностью.

Аппаратная реализация функций КС. Использование аппаратных средств снимает проблему обеспечения целостности системы. В большинстве современных систем защиты от НСД применяется зашивка программного обеспечения в ПЗУ или в аналогичную микросхему. Таким образом, для внесения изменений в ПО необходимо получить доступ к соответствующей плате и заменить микросхему.

В случае использования универсального процессора реализация подобных действий потребует применения специального оборудования, что еще более затруднит проведение атаки. Использование специализированного процессора с реализацией алгоритма работы в виде интегральной микросхемы полностью снимает проблему нарушения целостности этого алгоритма.

На практике для повышения класса защищенности КС функции аутентификации пользователя, проверки целостности (платы КРИПТОН-НСД, АККОРД и др.), криптографические функции (платы КРИПТОН-4, КРИПТОН-4К/8, КРИПТОН-4К/16, КРИПТОН-4/*PCI*, КРИПТОН-7/*PCI*, КРИПТОН-8/*PCI*),

образующие ядро системы безопасности, реализуются аппаратно (табл. 2.1), все остальные функции – программно.

Таблица 2.1

Аппаратные средства защиты КС и их характеристики

Наименование	Описание
КРИПТОН-4	Шифрование по ГОСТ 28147–89 (специализированным шифропроцессором «Блюминг-1»). Генерация случайных чисел. Хранение 3 ключей и 1 узла замены в шифраторе. Загрузка ключей в устройство до загрузки ОС с дискеты или со смарт-карты, минуя оперативную память ПК. Защита от НСД. Скорость шифрования до 350 Кбайт/с. Интерфейс шины *ISA*-8
КРИПТОН-4К/8	Функции устройства КРИПТОН-4. Более современная, чем в КРИПТОН-4, отечественная элементная база (шифропроцессор «БлюминМК»). Аппаратный журнал работы с устройством. Загрузка ключей с *Touch-Memory*. Скорость шифрования до 610 Кбайт/с. Интерфейс шины *ISA*-8
КРИПТОН-4К/16	Функции устройства КРИПТОН-4К/8. Функции электронного замка персонального компьютера – разграничение доступа, проверка целостности ОС. Скорость шифрования до 950 Кбайт/с. Интерфейс шины *ISA*-16
КРИПТОН-4/*PCI*	Функции устройства КРИПТОН-4К/16. Скорость шифрования до 1100 Кбайт/с. Интерфейс шины *PCI Target*. Возможность параллельной работы нескольких плат
КРИПТОН-7/*PCI*	Функции устройства КРИПТОН-4/*PCI*. Хранение до 1000 ключей (таблиц сетевых ключей) в защищенном ОЗУ. Управление доступом к ключам. Скорость шифрования до 1300 Кбайт/с. Интерфейс шины *PCI Master/Target*. Возможность параллельной работы нескольких плат
КРИПТОН-8/*PCI*	Функции устройства КРИПТОН-7/*PCI*. Хранение 32 ключей и 2 узлов замены в шифраторе, до 4000 ключей в защищенном ОЗУ. Аппаратная реализация быстрой смены ключей. Скорость шифрования до 8800 Кбайт/с. Интерфейс шины *PCI Master/Target*. Возможность параллельной работы нескольких плат
КРИПТОН-НСД	Шифрование по ГОСТ 28147–89 (программой из ПЗУ). Генерация случайных чисел. Защита от НСД. Загрузка ключей с дискет, смарт-карт и *Touch-Memory*
Специализированная сетевая плата	Размещение коммуникационных модулей внутри платы для исключения их обхода (стадия разработки)

Частично контролируемые компьютерные системы. К таким системам можно отнести современные КС, аттестовать программное обеспечение которых полностью не представляется возможным.

Безопасность в таких КС может быть обеспечена:

• использованием специальных аттестованных (полностью контролируемых) аппаратно-программных средств, выполняющих ряд защищенных операций и играющих роль специализированных модулей безопасности;

• изоляцией от злоумышленника ненадежной компьютерной среды, отдельного ее компонента или отдельного процесса с помощью полностью контролируемых средств.

В частично контролируемых КС использование каких-либо программно реализованных функций, отвечающих за шифрование, электронную цифровую подпись, доступ к информации, доступ к сети и т. д., становится показателем непрофессионализма администратора безопасности. Основную опасность представляет при этом возможность перехвата ключей пользователя, используемых при шифровании и предоставлении полномочий доступа к информации.

Одним из наиболее известных и надежных аппаратных модулей безопасности являются платы серии КРИПТОН, обеспечивающие как защиту ключей шифрования и электронной цифровой подписи, так и неизменность их алгоритмов. Все используемые в системе ключи могут шифроваться на мастер-ключе (загружаемом в плату минуя шину компьютера) и храниться на внешнем носителе в зашифрованном виде. Они расшифровываются только внутри платы, в которой применяются специальные методы фильтрации и зашумления для предотвращения возможности считывания ключей с помощью специальной аппаратуры. В качестве ключевых носителей используются дискеты, микропроцессорные электронные карточки (смарт-карты) и «таблетки» *Touch-Memory*.

Для построения защищенной сети необходимо прежде всего обеспечить защиту ее компонентов. К основным компонентам сети относятся:

• абонентские места, персональные компьютеры или терминалы клиента;

• центры коммутации пакетов, маршрутизаторы, шлюзы и сетевые экраны;

• корпоративный сервер, локальные серверы и серверы приложений;

• отдельные сегменты сетей.

Защита каждого из компонентов (как правило, компьютера) складывается из:

• исключения несанкционированного доступа к компьютеру со стороны консоли;

• разграничения доступа к ресурсам компьютера со стороны консоли;

• исключения несанкционированного доступа к компьютеру со стороны сети;

• разграничения доступа к ресурсам компьютера со стороны сети;

• обеспечения секретности используемых для защиты криптографических ключей.

Кроме того, необходимо также защитить сеть целиком от проникновения извне и каналы обмена с другими сетями.

Основные элементы и средства защиты от несанкционированного доступа

Фирма АНКАД известна на российском рынке как разработчик, производитель и поставщик аппаратно-программных криптографических средств защиты информации серии КРИПТОН. Традиционно они выпускались в виде устройств с минимальным программным обеспечением. Встраивание их в конечные системы осуществлялось пользователем. В настоящий момент наряду с производством и поставкой устройств фирма предлагает готовые

решения: от программ абонентского шифрования и электронной подписи до защиты отдельных рабочих мест и систем в целом.

В состав средств криптографической защиты информации (СКЗИ) фирмы АНКАД включены (рис. 2.2):

• устройства криптографической защиты данных (УКЗД) и их программные эмуляторы;

• контроллеры смарт-карт;

• системы защиты информации от несанкционированного доступа (СЗИ НСД);

• программы абонентского шифрования, электронной подписи и защиты электронной почты;

• коммуникационные программы «прозрачного» шифрования IP-пакетов и ограничения доступа к компьютеру по сети;

• криптомаршрутизаторы;

• библиотеки поддержки различных типов смарт-карт;

• библиотеки функций шифрования и электронной цифровой подписи для различных операционных систем.

Отдельным рядом (семейством) устройств с использованием криптографических методов защиты являются специализированные модули безопасности для терминального оборудования, контрольно-кассовых машин, банкоматов и другого оборудования, используемого в платежных и расчетных системах.

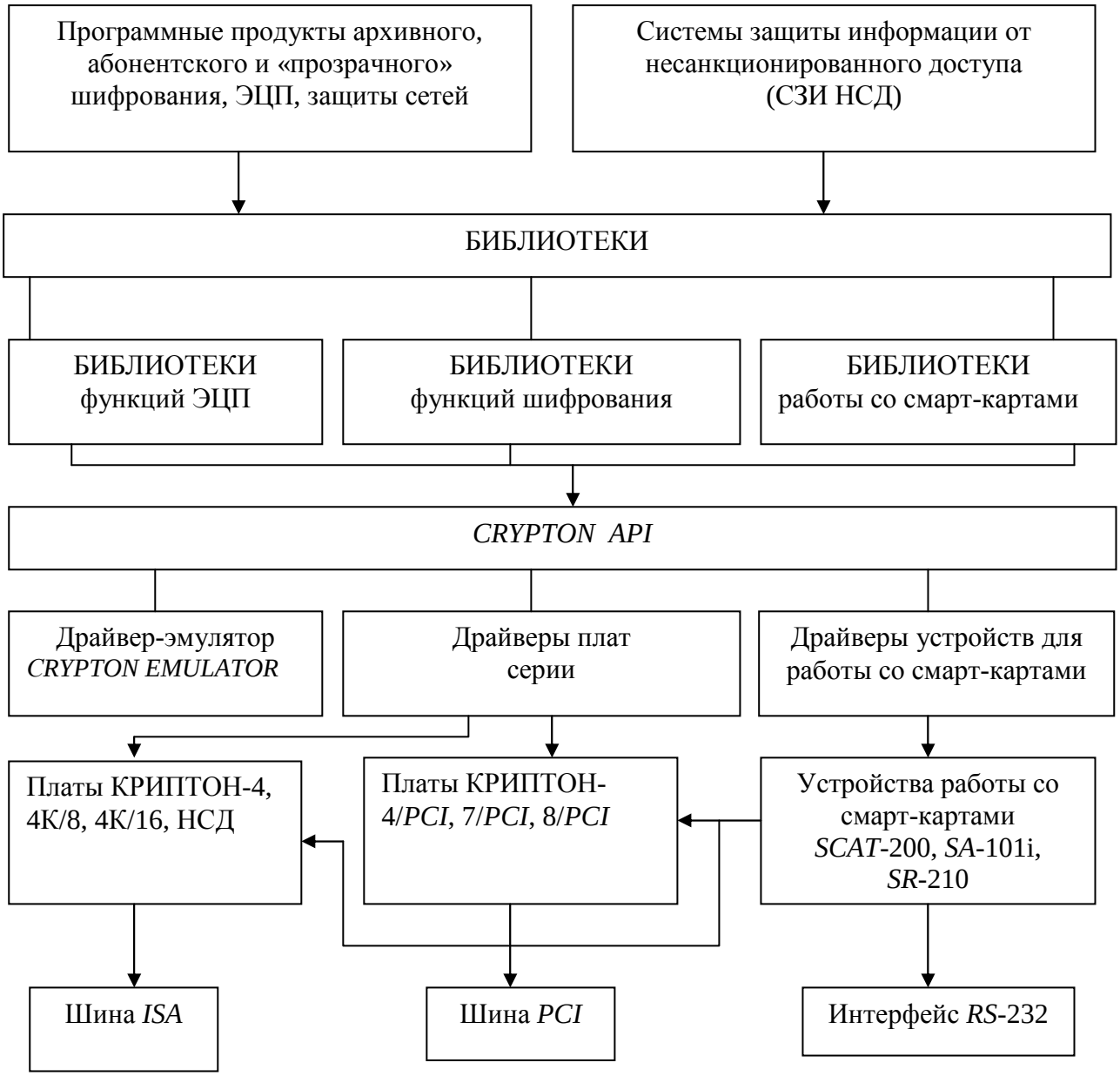


Рис. 2.2. Структура средств криптографической защиты информации

Устройства криптографической защиты данных серии КРИПТОН

Отличительной особенностью семейства устройств криптографической защиты данных (УКЗД) фирмы АНКАД является разработанная ею специализированная микропроцессорная элементная база для наиболее полной и достоверной аппаратной реализации российского стандарта шифрования (табл. 2.2). Серийно выпускаются УКЗД КРИПТОН-4, 4К/8 и 4К/16,

предназначенные для шифрования по ГОСТ 28147–89 и генерации случайных чисел при формировании ключей.

Для систем защиты информации от несанкционированного доступа разработана специальная плата КРИПТОН-НСД, выполняющая программное шифрование по ГОСТ 28147–89, аппаратную генерацию случайных чисел, загрузку ключей с дискет, смарт-карт или *Touch Memory*.

Таблица 2.2

Характеристики микропроцессорных элементов УКЗД серии КРИПТОН

Наименование	Описание
SA-101i (Адаптер смарт-карт)	Запись / чтение информации на / с смарт-карты *EEPROM* (протокол 12*C*). Интерфейс с УКЗД серии КРИПТОН, обеспечивающий прямую загрузку ключей в устройство.
SCAT-200 (Контроллер смарт-карт)	Шифрование по ГОСТ 28147–89, *DES*. Память для хранения одного мастер-ключа. Генерация случайных чисел. Запись / чтение информации на / с смарт-карты. Протоколы карт: 12C, *GPM*, *ISO* 7810 T = 0. Интерфейс *RS*-232 с компьютером и специализированный интерфейс с УКЗД серии КРИПТОН.
SR-210 (Контроллер смарт-карт)	Запись / чтение информации на / с смарт-карты. Протоколы карт: 12C, *GPM*, *ISO* 7816 T = 0, T = 1. Интерфейс *RS*-232 с компьютером и специализированный интерфейс с УКЗД серии КРИПТОН.

Для встраивания в конечные системы пользователя УКЗД имеют два уровня интерфейса в виде набора команд устройства и библиотеки функций. Наиболее важными особенностями рассматриваемых плат являются:

• наличие загружаемого до загрузки операционной системы мастер-ключа, что исключает его перехват;

• выполнение криптографических функций внутри платы, что исключает их подмену или искажение;

• наличие аппаратного датчика случайных чисел;

• реализация функций проверки целостности файлов операционной системы и разграничения доступа к компьютеру;

• высокая скорость шифрования: от 350 Кбайт/с (КРИПТОН-4) до 8800 Кбайт/с (КРИПТОН 8/*PCI*).

Допустимо параллельное подключение нескольких устройств одновременно в одном персональном компьютере, что может значительно повысить интегральную скорость шифрования и расширить другие возможности при обработке информации.

Средства серии КРИПТОН независимо от операционной среды обеспечивают:

• защиту ключей шифрования и электронной цифровой подписи (ЭЦП);

• неизменность алгоритма шифрования и ЭЦП.

Устройства для работы со смарт-картами

Для ввода ключей, записанных на смарт-карты, предлагаются разработанные фирмой АНКАД устройства для работы со смарт-картами, функции которых приведены в табл. 2.2.

Адаптер смарт-карт *SA*-101i предназначен для чтения и записи информации на смарт-картах. Адаптер подключается к УКЗД КРИПТОН и позволяет вводить в него ключи шифрования, хранящиеся на смарт-карте пользователя.

На одной смарт-карте могут быть размещены:

• таблица заполнения блока подстановок устройств защиты (ГОСТ 28147–89);

• главный ключ шифрования;

• секретный и открытый ключи электронной цифровой подписи пользователя;

• открытый ключ ЭЦП сертификационного центра;

• идентификатор пользователя системы защиты от несанкционированного доступа КРИПТОН-ВЕТО.

Адаптер *SA*-101i выпускается во внутреннем исполнении и легко встраивается в персональный компьютер на свободное место, предназначенное для дисковода.

Универсальный контроллер смарт-карт *SCAT*-200 предназначен для работы со смарт-картами. Контроллер *SCAT*-200 может подключаться как к УКЗД, так и к интерфейсу *RS*-232. Наиболее важными представляются следующие функции контроллера:

• запись информации на смарт-карту;

• чтение информации со смарт-карты;

• шифрование по ГОСТ 28147–89 и *DES*;

• хранение секретных ключей (так же, как в плате КРИПТОН-4);

• генерация случайной последовательности;

• набор на клавиатуре *PIN*-кода.

В контроллере могут применяться электронные карточки:

• открытая память (протокол *I2C*);

• защищенная память (серия *GPM*);

• микропроцессорные карты (*PCOS*).

Универсальный контроллер *SCAT*-200 позволяет строить информационные системы на базе смарт-карт, что делает его полезным для систем:

• безналичных расчетов (дебетно/кредитные карты);

• контроля доступа (хранение прав доступа);

• хранения конфиденциальной информации (медицина, страхование, финансы);

• защиты информации (хранение идентификаторов, паролей и ключей шифрования).

Контроллер может использоваться в компьютерах, электронных кассовых аппаратах, электронных замках, торговых автоматах, бензоколонках, платежных терминалах на базе *IBM* совместимых компьютеров. Контроллер *SCAT*-200 – совместный продукт фирмы АНКАД и АО «Скантек».

Системы защиты информации от несанкционированного доступа

Система криптографической защиты информации от НСД КРИПТОН-ВЕТО. Система ограничивает круг лиц и их права доступа к информации на персональном компьютере. Ее реализация основана на технологиях «прозрачного» шифрования логических дисков по алгоритму ГОСТ 28147–89 и электронной цифровой подписи по ГОСТ 34.10/11–94.

Персональный компьютер при этом может использоваться в качестве:

• абонентского пункта;

• центра коммутации пакетов;

• центра выработки ключей.

В состав основных функций системы КРИПТОН-ВЕТО включены следующие:

• обеспечение секретности информации в случае кражи винчестера или ПК;

• обеспечение защиты от несанкционированного включения компьютера;

• разграничение полномочий пользователей по доступу к ресурсам компьютера;

• проверка целостности используемых программных средств системы в момент ее включения;

• проверка целостности программы в момент ее запуска на выполнение;

• запрещение запуска на выполнение посторонних программ;

• ведение системного журнала, регистрирующего события, возникающие в системе;

• обеспечение «прозрачного» шифрования информации при обращении к защищенному диску;

• обнаружение искажений, вызванных вирусами, ошибками пользователей, техническими сбоями или действиями злоумышленника.

Основным аппаратным элементом системы являются серийно выпускаемые аттестованные ФАПСИ платы серии КРИПТОН, с помощью которых проверяется целостность системы и выполняется шифрование по ГОСТ 28147–89. Система предполагает наличие администратора безопасности, который определяет взаимодействие между управляемыми ресурсами.

Аппаратно-программные средства ЗКИ ОКБ САПР

ОКБ САПР предлагает следующие решения в области защиты информации:

1. Линейка ПСКЗИ семейства ШИПКА™:

– ШИПКА – персональное средство криптографической защиты информации.

2. Линейка СЗИ НСД семейства АККОРД™:

– Аккорд-АМДЗ – аппаратный модуль доверенной загрузки, предназначенный для применения на *IBM*-совместимых ПЭВМ, рабочих станциях ЛВС с целью защиты средств вычислительной техники и информационных ресурсов от несанкционированного доступа;

– Аккорд-5.5 – серия контроллеров для разных системных шин, совмещающих функции АМДЗ и аппаратно реализованные криптографические функции;

– Крещендо – устройство для высокопроизводительной обработки информации;

– Программно-аппаратный комплекс Аккорд-*NT*/2000 *V*3.0 – в составе Аккорд-АМДЗ и специального программного обеспечения, реализующего правила разграничения доступа к информации, предназначен для

разграничения доступа пользователей к рабочим станциям, терминалам и терминальным серверам;

– Модуль доверенной загрузки Аккорд-*XR* – предназначен для применения на ПЭВМ (рабочих станциях ЛВС) типа *IBM PC*, функционирующих под управлением ОС *Microsoft Windows XP Professional* (*Service Pack* 1*a*);

3. Подсистемы автоматизации работы с СЗИ Аккорд и ШИПКА.

– Подсистема распределенного аудита и управления Аккорд-РАУ – это ПО для автоматизации управления защитой информации в АС.

– *Privac y* – ПАК для криптографической защиты с помощью ПСКЗИ ШИПКА данных на жестком диске и передаваемых по сети

4. Коммутатор *SATA*-устройств.

5. Устройство блокировки *USB*-портов.

6. Нормативно-методические документы по политике информационной безопасности.

Линейка ПСКЗИ семейства ШИПКА™

ШИПКА – это *USB*-устройство, представляющее собой «специализированный компьютер» размером со среднюю флешку. Этот компьютер имеет собственный процессор, который реализует криптографические алгоритмы самостоятельно, без участия процессора того компьютера, в *USB*-разъем которого ШИПКА вставлена (рис. 2.3).

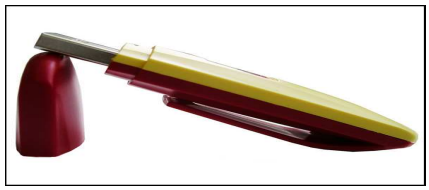


Рис. 2.3 Устройство ШИПКА

Криптографическая защита информации – это, в первую очередь, шифрование и электронная цифровая подпись. Искушенные пользователи прекрасно знают, что зашифровать файл или подписать его ЭЦП можно и без дополнительного устройства – с помощью ПО, устанавливаемого на компьютер. Зачем тогда нужна ШИПКА?

Для того чтобы обойти «тонкие места». Они заключаются в том, *как* и *где* производится преобразование данных.

Первое тонкое место – *как генерируется* и *где хранится* ключ (или ключевая пара). *Генерировать* ключи в компьютере опасно, поскольку это среда ненадежная, позволяющая злоумышленнику, например, подсунуть пользователю заранее записанный ключ или сгенерировать ключ так, чтобы его было можно просчитать. ШИПКА генерирует ключи самостоятельно, имея для этого физический датчик случайных чисел.

Хранить ключ в памяти компьютера опасно потому, что существуют различные возможности его оттуда извлечь или восстановить. Кроме того, ключи человеку могут понадобиться не только на своем ПК, а записывать такую информацию на разного рода носители еще более опасно. В случае же использования ШИПКИ ключи всегда доступны их владельцу, но никогда не доступны никому другому (рис. .2.4).

Рис. 2.4. Окно управления ключами в ПСКЗИ ШИПКА

Второе тонкое место − правильность реализации самого преобразования. ПО компьютера может быть модифицировано злоумышленником таким образом, что, создавая видимость зашифровывания, в действительности оно будет производить совсем другие действия, никак не защищающие, а возможно, даже разрушающие данные. Кроме того, даже если преобразование протекает правильно, в этот момент ничто не мешает злоумышленнику перехватить ключ. ШИПКА выполняет все защитные действия, не обращаясь к ПО компьютера: она в этом не нуждается, поскольку сама является компьютером. При этом ПО ШИПКИ невозможно модифицировать извне, поэтому ШИПКА − это *доверенная среда*, значит, преобразованиям, которые она выполняет, можно *доверять*. При этом ШИПКА − это небольшое по размеру и простое в использовании устройство.

Линейка СЗИ НСД семейства АККОРД™

СЗИ НСД Аккорд-АМДЗ – это аппаратный модуль доверенной загрузки (АМДЗ) для *IBM*-совместимых ПК – серверов и рабочих станций локальной сети, обеспечивающий защиту устройств и информационных ресурсов от несанкционированного доступа.

Доверенная загрузка – это загрузка различных операционных систем только с заранее определенных постоянных носителей (например, только с жесткого диска) после успешного завершения специальных процедур: проверки целостности технических и программных средств ПК (с использованием механизма пошагового контроля целостности) и идентификации / аутентификации пользователя.

Комплекс начинает работу сразу после выполнения штатного *BIOS* компьютера – до загрузки операционной системы, и обеспечивает доверенную загрузку ОС, поддерживающих файловые системы *FAT* 12, *FAT* 16, *FAT* 32, *NTFS*, *HPFS*, *EXT*2*FS*, *EXT*3*FS*, *FreeBSD*, *Sol*86*FS*, *QNXFS*, *MINIX*. Это, в частности, ОС семейств *MS DOS*, *Windows* (*Windows* 9*x*, *Windows ME*, *Windows*

NT, *Windows* 2000, *Windows XP*, *Windows* 2003, *Windows Vista*), *QNX*, *OS*/2, *UNIX*, *LINUX*, *BSD* и др.

Аккорд-АМДЗ может быть реализован в пяти основных вариантах: Аккорд-5 (для шинного интерфейса *PCI*), унифицированный контроллер для шин *PCI* и *PCI-X* – Аккорд-5*MX*, его функциональный аналог в стандарте *mini-PCI* – Аккорд-5*MX mini-PCI*, унифицированный контроллер (*PCI*, *PCI-X*) Аккорд-5.5, имеющий мощную аппаратно реализованную криптографическую подсистему, а также его версия для шины *PCIe* – Аккорд-5.5.e (рис. 2.5).

Контроллеры могут быть оснащены интерфейсом блокировки двух и более физических каналов (*FDD*, *HDD* (*IDE*), *ATX*, *EATX*). В Аккорд-5.5 также реализована возможность отключения питания компьютера в случае, если за *N* секунд не начал работу *BIOS* АМДЗ. Аккорд-АМДЗ позволяет использовать для идентификации пользователей смарт-карты, устройства *iButton*, устройства считывания отпечатков пальцев, а также устройство ШИПКА.

Рис. 2.5. Контроллер «Аккорд 5.5 *mini-PCI-expre*»

Комплекс применим для построения систем защиты информации от несанкционированного доступа в соответствии с руководящими документами ФСТЭК Гостехкомиссии России «Защита от несанкционированного доступа к информации».

Программно-аппаратный комплекс средств защиты информации Аккорд-NT/2000 V3.0

Программно-аппаратный комплекс средств защиты информации (ПАК СЗИ) Аккорд-*NT*/2000 *V*3.0 предназначен для разграничения доступа пользователей к рабочим станциям, терминалам и терминальным серверам. Комплекс работает на всей ветви операционных систем (ОС) *Microsoft NT* +, на терминальных серверах, построенных на базе ОС *Windows* 2000 *Advanced Server* и на базе серверов семейства *Windows* 2003, и на ПО *Citrix Metaframe XP*, работающем на этих ОС.

Аккорд-*NT*/2000 *V*3.0 обеспечивает:

- защиту от несанкционированного доступа к ПЭВМ;
- идентификацию / аутентификацию пользователей до загрузки операционной системы с последующей передачей результатов успешной идентификации / аутентификации в операционную систему;
- аппаратный контроль целостности системных файлов и критичных разделов реестра;
- доверенную загрузку ОС;
- контроль целостности программ и данных, их защиту от несанкционированных модификаций;
- создание индивидуальной для каждого пользователя изолированной рабочей программной среды;
- запрет запуска неразрешенных программ;
- разграничение доступа пользователей к массивам данных и программам с помощью дискреционного контроля доступа;
- разграничение доступа пользователей и процессов к массивам данных с помощью мандатного контроля доступа;
- автоматическое ведение протокола регистрируемых событий в энергонезависимой памяти аппаратной части комплекса;

- усиленную аутентификацию терминальных станций с помощью контроллера Аккорд или ПСКЗИ ШИПКА;
- идентификацию / аутентификацию пользователей, подключающихся к терминальному серверу (с использованием ТМ-идентификатора или ПСКЗИ ШИПКА);
- опциональную автоматическую идентификацию в системе *Windows NT+* и на терминальном сервере пользователей, аутентифицированных защитными механизмами контроллера АМДЗ (при таком подходе, избегая повторной идентификации пользователей, можно гарантировать, что ОС будет загружена под именем того же пользователя, который был аутентифицирован в контроллере АМДЗ, и к терминальному серверу подключится тот же самый пользователь);
- управление терминальными сессиями;
- контроль печати на принтерах, подключенных как к терминальным серверам, так и к пользовательским терминалам, который позволяет протоколировать вывод документов на печать и маркировать эти документы (в качестве маркера может выступать гриф секретности документа, имя пользователя, имя принтера, имя документа и другая служебная информация).

Комплекс использует собственную систему разграничения доступа (мандатный и дискреционный методы контроля) и служит фильтром между ядром ОС и расположенным выше прикладным ПО терминальных служб. Это значит, что действия, разрешенные прикладным ПО, но запрещенные АККОРДОМ − будут запрещены пользователю. Комплексная защита терминальной сессии обеспечивается тогда и только тогда, когда пользователь может работать только с защищенного терминала и только с защищенным терминальным сервером. Для этого в момент создания терминальной сессии со стороны терминального сервера необходимо аутентифицировать не только пользователя, но и терминал, а со стороны терминала необходимо убедиться в том, что терминальный сервер действительно тот, с которым должен работать

пользователь. Это возможно только при наличии активных СЗИ и на терминале, и на сервере. Стало быть, для защиты терминальной сессии необходимо установить комплекс не только на терминальный сервер, но и на терминалы.

В предлагаемой системе защиты терминальных сессий комплексы Аккорд, установленные на терминальных серверах и на пользовательских терминалах, взаимодействуют в рамках виртуальных каналов, построенных на протоколах *RDP* и *ICA*. Это позволяет использовать уже установленную связь между сервером и терминалом, а не устанавливать новую. При этом состав полномочий пользователя инвариантен как к протоколу подключения, так и к типу терминального сервера (*Microsoft* или *Citrix*).

В течение всего сеанса работы пользователя ведется подробный журнал событий, в котором фиксируются все действия пользователя на терминальном сервере.

Программное обеспечение комплексов позволяет администратору безопасности информации описать любую непротиворечивую политику безопасности на основе наиболее полного набора атрибутов (табл. 2.3).

Таблица 2.3

Атрибуты политики безопасности

Операции с файлами	
R	разрешение на открытие файлов только для чтения
W	разрешение на открытие файлов для записи
C	разрешение на создание файлов на диске
D	разрешение на удаление файлов
N	разрешение на переименование файлов
V	видимость файлов
O	эмуляция разрешения на запись информации в открытый файл

Окончание таблицы

Операции с каталогами	
M	создание каталогов на диске
E	удаление каталогов с диска
G	разрешение перехода в этот каталог
n	переименование подкаталогов
S	наследование прав на все вложенные подкаталоги
1	наследование прав на 1-й уровень вложенности
0	запрет наследования прав на все вложенные подкаталоги
Прочее	
X	разрешение на запуск программ
Регистрация	
r	регистрация в журнале операций чтения при обращении к объекту
w	регистрация в журнале операций записи при обращении к объекту

Для разграничения прав доступа к информационным ресурсам, кроме дискреционного, осуществлен мандатный принцип доступа субъектов.

Кроме этого, администратор БИ для каждого субъекта – пользователя системы – определяет:

- перечень файлов, целостность которых должна контролироваться системой, и опции контроля;
- запуск стартовой задачи (для функционально замкнутых систем);
- наличие либо отсутствие привилегий супервизора;
- детальность журнала доступа;
- назначение / изменение пароля для аутентификации;
- временные ограничения – время по дням недели (с дискретностью 30 мин), в которое разрешено начало работ для данного субъекта;
- параметры управления экраном – гашение экрана через заранее определенный интервал времени (в случае, если в течение указанного

интервала действия оператором не выполнялись), подача соответствующих звуковых и визуальных сигналов.

ПО Аккорд-*NT*/2000 имеет интерфейс подключения внешних антивирусных модулей. В качестве такого модуля может применяться Антивирусное ядро *Vba*32, разработанное фирмой ВирусБлокАда, или Антивирусное ядро *DrWeb*. Совместная работа Аккорд-*NT*/2000 и антивирусного ядра позволяет не только добавить в ПО Аккорд-*NT*/2000 новую функцию обнаружения и обезвреживания вредоносных программ (при этом динамически проверяются только те объекты, к которым обращается пользователь), но и существенно ускорить работу за счет исключения дублирования проверок (проверки производятся одновременно, а не последовательно, соответственно каждый объект проверяется единожды).

Также в Аккорд-NT/2000 реализована возможность контроля доступа к *USB*-устройствам и контроля печати на принтерах.

Основные характеристики Аккорд-*NT*/2000 *V*3.0 изложены в табл. 2.4.

Таблица 2.4

Основные характеристики Аккорд-*NT*/2000 *V*3.0

Работа под операционными системами	*Windows NT*, все версии *Windows* 2000, *Windows XP*
Класс защиты	до 1В включительно
Наличие сертификата Гостехкомиссии России	№ 1161/1 12 февраля 2007 г.
Используемые контроллеры	Аккорд-5, Аккорд-5*MX*, Аккорд-5.5, Аккорд-5.5 *PCIe*, Аккорд-5.5MP, Аккорд-5.5ME
Идентификация (тип идентификатора)	*Touch memory DS*-199x, ПСКЗИ ШИПКА
Аутентификация пользователя	По паролю, вводимому с клавиатуры

Устройство блокировки USB-портов

Устройство (рис. 2.6.) предназначено для блокировки двух портов *USB*, поддерживает стандарты *USB* 1.1 и *USB* 2.0.

Рис. 2.6. Устройство блокировки *USB*-портов

Подключается на внутренние разъемы *USB* на материнской плате, которые обычно выводятся на переднюю панель системного блока.

Таким образом, физически заблокировав доступ к внешним *USB*-разъемам на задней панели материнской платы, можно разграничить доступ пользователей к *USB*-портам, расположенным спереди или вынесенным на заднюю панель с помощью *USB*-брэкета .

Квантовая криптография

Квантовая криптография

Базовой задачей криптографии является шифрование данных и аутентификация отправителя. Это легко выполнить, если как отправитель, так и получатель имеют псевдослучайные последовательности битов, называемые ключами. Перед началом обмена каждый из участников должен получить ключ, причем эту процедуру следует выполнить с наивысшим уровнем конфиденциальности так, чтобы никакая третья сторона не могла получить доступ даже к части этой информации. Задача безопасной пересылки ключей может быть решена с помощью квантовой рассылки ключей *QKD* (*Quantum Key*

Distribution). Надежность метода основывается на нерушимости законов квантовой механики. Злоумышленник не может отвести часть сигнала с передающей линии, так как нельзя поделить электромагнитный квант на части. Любая попытка злоумышленника вмешаться в процесс передачи вызовет непомерно высокий уровень ошибок. Степень надежности в данной методике выше, чем в случае применения алгоритмов с парными ключами (например, *RSA*). Здесь ключ может генерироваться во время передачи по совершенно открытому оптическому каналу. Скорость передачи данных при этой технике не высока, но для передачи ключа она и не нужна. По существу квантовая криптография может заменить алгоритм Диффи – Хеллмана, который в настоящее время часто используется для пересылки секретных ключей шифрования по каналам связи.

Первый протокол квантовой криптографии (BB84) был предложен и опубликован в 1984 году Беннетом и Брассардом. Позднее идея была развита Экертом в 1991 году. В основе метода квантовой криптографии лежит наблюдение квантовых состояний фотонов. Отправитель задает эти состояния, а получатель их регистрирует. Здесь используется квантовый принцип неопределенности, когда две квантовые величины не могут быть измерены одновременно с требуемой точностью. Так, поляризация фотонов может быть ортогональной (горизонтальной или вертикальной), диагональной (45° или 135°) или циркулярной (левой или правой). Измерение одного вида поляризации рандомизирует другую составляющую. Таким образом, если отправитель и получатель не договорились между собой, какой вид поляризации брать за основу, получатель может разрушить посланный отправителем сигнал, не получив никакой полезной информации.

Отправитель кодирует отправляемые данные, задавая определенные квантовые состояния, получатель регистрирует эти состояния. Затем получатель и отправитель совместно обсуждают результаты наблюдений. В конечном итоге со сколь угодно высокой достоверностью можно быть

уверенным, что переданная и принятая кодовые последовательности тождественны. Обсуждение результатов касается ошибок, внесенных шумами или злоумышленником, и ни в малейшей мере не раскрывает содержимого переданного сообщения. Может обсуждаться четность сообщения, но не отдельные биты. При передаче данных контролируется поляризация фотонов. Поляризация может быть ортогональной (горизонтальной или вертикальной), циркулярной (левой или правой) и диагональной (45° или 135°).

В качестве источника света может использоваться светоизлучающий диод или лазер. Свет фильтруется, поляризуется и формируется в виде коротких импульсов малой интенсивности. Поляризация каждого импульса модулируется отправителем произвольным образом в соответствии с одним из четырех состояний (горизонтальная, вертикальная, лево- или право-циркулярная).

Получатель измеряет поляризацию фотонов, используя произвольную последовательность базовых состояний (ортогональная или циркулярная). Получатель открыто сообщает отправителю, какую последовательность базовых состояний он использовал. Отправитель открыто уведомляет получателя о том, какие базовые состояния использованы корректно. Все измерения, выполненные при неверных базовых состояниях, отбрасываются. Измерения интерпретируются согласно двоичной схеме: лево-циркулярная поляризация или горизонтальная – 0, право-циркулярная или вертикальная – 1. Реализация протокола осложняется присутствием шума, который может вызвать ошибки. Вносимые ошибки могут быть обнаружены и устранены с помощью подсчета четности, при этом один бит из каждого блока отбрасывается.

Беннет в 1991 году предложил следующий протокол.

1. Отправитель и получатель договариваются о произвольной перестановке битов в строках, чтобы сделать положения ошибок случайными.

2. Строки делятся на блоки размера k (k выбирается так, чтобы вероятность ошибки в блоке была мала).

3. Для каждого блока отправитель и получатель вычисляют результаты и открыто оповещают о них друг друга. Последний бит каждого блока удаляется.

4. Для каждого блока, где четность оказалась разной, получатель и отправитель производят итерационный поиск и исправление неверных битов.

5. Чтобы исключить кратные ошибки, которые могут быть не замечены, операции пунктов 1 – 4 повторяются для большего значения k.

6. Для того чтобы определить, остались или нет необнаруженные ошибки, получатель и отправитель повторяют псевдослучайные проверки:

- получатель и отправитель открыто объявляют о случайном перемешивании позиций половины битов в их строках;
- получатель и отправитель открыто сравнивают четности. Если строки отличаются, четности должны не совпадать с вероятностью $\frac{1}{2}$;
- если имеет место отличие, получатель и отправитель используют двоичный поиск и удаление неверных битов;
- если отличий нет, после m итераций получатель и отправитель получают идентичные строки с вероятностью ошибки 2^{-m}.

Схема реализации однонаправленного канала с квантовым шифрованием показана на рис. 2.7. Передающая сторона находится слева, а принимающая – справа. Ячейки Покеля служат для импульсной вариации поляризации потока квантов передатчиком и для анализа импульсов поляризации приемником. Передатчик может формировать одно из четырех состояний поляризации (0, 45, 90 и 135 градусов). Собственно передаваемые данные поступают в виде управляющих сигналов на эти ячейки.

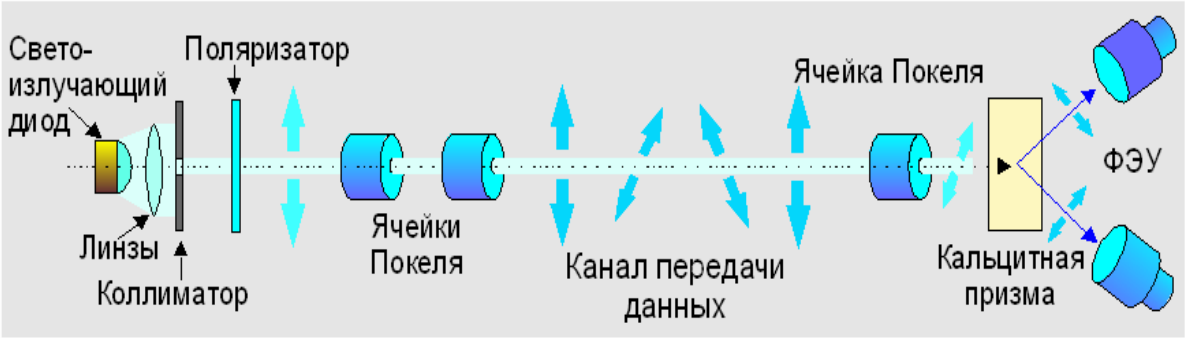


Рис. 2.7. Практическая схема реализации идеи квантовой криптографии

На принимающей стороне после ячейки Покеля ставится кальцитовая призма, которая расщепляет пучок на два фотодетектора (ФЭУ), измеряющих две ортогональные составляющие поляризации. При формировании передаваемых импульсов квантов приходится решать проблему их интенсивности. Если квантов в импульсе 1000, есть вероятность того, что 100 квантов по пути будет отведено злоумышленником на свой приемник. Анализируя позднее открытые переговоры между передающей и принимающей стороной, он может получить нужную ему информацию. В идеале число квантов в импульсе должно быть около одного. Здесь любая попытка отвода части квантов злоумышленником приведет к существенному росту числа ошибок у принимающей стороны. В этом случае принятые данные должны быть отброшены и попытка передачи повторена. Но, делая канал более устойчивым к перехвату, мы в этом случае сталкиваемся с проблемой «темнового» шума (выдача сигнала в отсутствии фотонов на входе) приемника (ведь мы вынуждены повышать его чувствительность). Для того чтобы обеспечить надежную транспортировку данных, логическому нулю и единице могут соответствовать определенные последовательности состояний, допускающие коррекцию одинарных и даже кратных ошибок.

Дальнейшего улучшения надежности криптосистемы можно достичь, используя эффект *EPR* (*Einstein – Podolsky – Rosen*). Эффект *EPR* возникает, когда сферически симметричный атом излучает два фотона в противоположных

направлениях в сторону двух наблюдателей. Фотоны излучаются с неопределенной поляризацией, но в силу симметрии их поляризации всегда противоположны. Важной особенностью этого эффекта является то, что поляризация фотонов становится известной только после измерения. На основе *EPR* Экерт предложил криптосхему, которая гарантирует безопасность пересылки и хранения ключа. Отправитель генерирует некоторое количество *EPR* фотонных пар. Один фотон из каждой пары он оставляет для себя, второй посылает своему партнеру. При этом, если эффективность регистрации близка к единице, при получении отправителем значения поляризации 1 его партнер зарегистрирует значение 0, и наоборот. Ясно, что таким образом партнеры всякий раз, когда требуется, могут получить идентичные псевдослучайные кодовые последовательности. Практически реализация данной схемы проблематична из-за низкой эффективности регистрации и измерения поляризации одиночного фотона.

Неэффективность регистрации является платой за секретность. Следует учитывать, что при работе в однофотонном режиме возникают чисто квантовые эффекты. При горизонтальной поляризации (*H*) и использовании вертикального поляризатора (*V*) результат очевиден – фотон не будет зарегистрирован. При 45° поляризации фотона и вертикальном поляризаторе (*V*) вероятность регистрации – 50 %. Именно это обстоятельство и используется в квантовой криптографии. Результаты анализа при передаче двоичных разрядов представлены в табл. 2.5. Здесь предполагается, что для передатчика логическому нулю соответствует поляризация *V*, а единице – +45°, для принимающей стороны логическому нулю соответствует поляризация – 45°, а единице – *H*.

Понятно, что в первой и четвертой колонке поляризации передачи и приема ортогональны и результат детектирования будет отсутствовать. В колонках 2 и 3 коды двоичных разрядов совпадают и поляризации не ортогональны. По этой причине с вероятностью 50 % может быть позитивный

результат в любом из этих случаев (и даже в обоих). В таблице предполагается, что успешное детектирование фотона происходит для случая колонки 3. Именно этот бит становится первым битом общего секретного ключа передатчика и приемника.

Таблица 2.5

Результаты анализа при передаче двоичных разрядов

Передаваемый бит	1	0	1	0
Поляризация передачи	+45°	*V*	+45°	*V*
Поляризация приема	– 45°	– 45°	*H*	*H*
Биты кода на приеме	0	0	1	1
Результаты приема	–	–	+	–

Понятно, что в первой и четвертой колонке поляризации передачи и приема ортогональны и результат детектирования будет отсутствовать. В колонках 2 и 3 коды двоичных разрядов совпадают и поляризации не ортогональны. По этой причине с вероятностью 50 % может быть позитивный результат в любом из этих случаев (и даже в обоих). В таблице предполагается, что успешное детектирование фотона происходит для случая колонки 3. Именно этот бит становится первым битом общего секретного ключа передатчика и приемника.

Однофотонные состояния поляризации более удобны для передачи данных на большие расстояния по оптическим кабелям. Такого рода схема показана на рис. 2.8 (алгоритм B92).

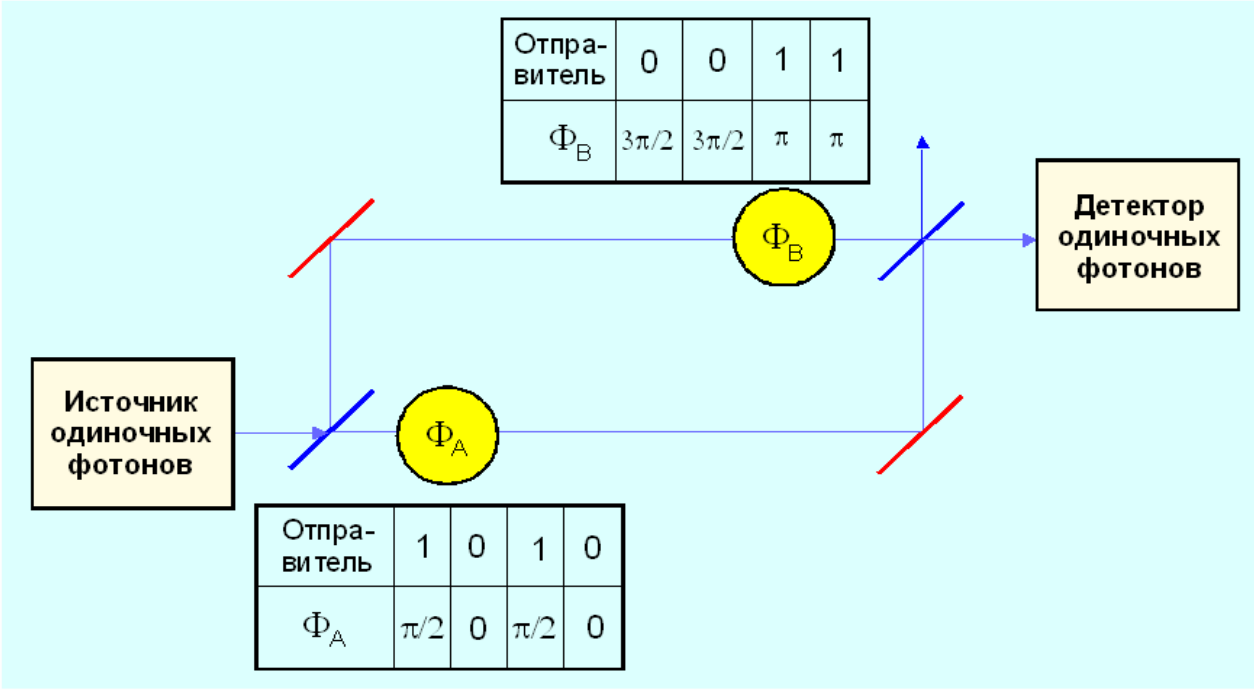


Рис. 2.8. Реализация алгоритма B92

В алгоритме B92 приемник и передатчик создают систему, базирующуюся на интерферометрах Маха – Цендера. Отправитель определяет углы фазового сдвига, соответствующие логическому нулю и единице ($\Phi_A = \pi/2$), а приемник задает свои фазовые сдвиги для логического нуля ($\Phi_B = 3\pi/2$) и единицы ($\Phi_B = \pi$). В данном контексте изменение фазы 2π соответствует изменению длины пути на одну длину волны используемого излучения.

Хотя фотоны ведут себя при детектировании как частицы, они распространяются как волны. Вероятность того, что фотон, посланный отправителем, будет детектирован получателем, равна

$$P_D = \cos^2\{(\Phi_A - \Phi_B)/2\},$$

и характеризует интерференцию амплитуд волн, распространяющихся по верхнему и нижнему путям (см. рис. 2.8). Вероятность регистрации будет варьироваться от 1 (при нулевой разности фаз) до 0. Здесь предполагается, что отправитель и получатель используют фазовые сдвиги $(\Phi_A, \Phi_B) = (0, 3\pi/2)$ для

нулевых битов и $(\Phi_A, \Phi_B) = (\pi/2, \pi)$ для единичных битов (для алгоритма BB84 используются другие предположения).

Для регистрации одиночных фотонов помимо ФЭУ могут использоваться твердотельные лавинные фотодиоды (германиевые и *InGaAs*). Для понижения уровня шума их следует охлаждать. Эффективность регистрации одиночных фотонов лежит в диапазоне 10 – 40 %. При этом следует учитывать также довольно высокое поглощение света оптическим волокном (~0,3 – 3 ДБ/км). Схема интерферометра с двумя волокнами достаточно нестабильна из-за разных свойств транспортных волокон и может успешно работать только при малых расстояниях. Лучших характеристик можно достичь, мультиплексируя оба пути фотонов в одно волокно (рис. 2.9).

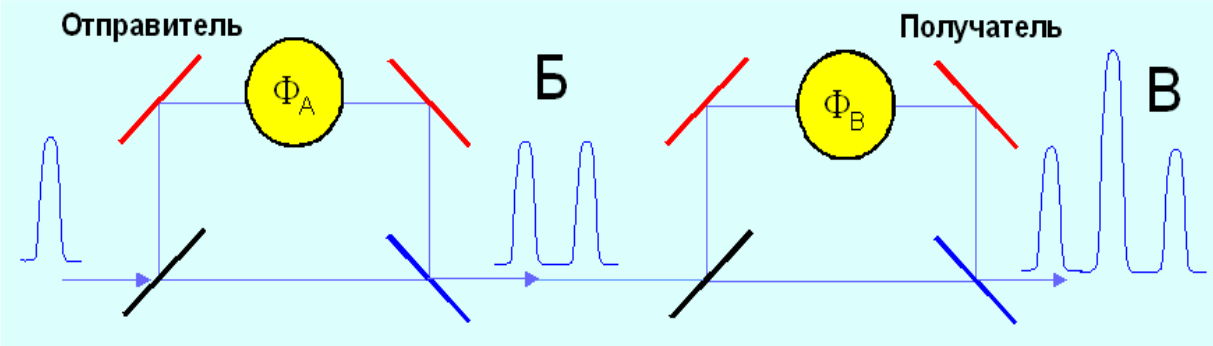


Рис. 2.9. Интерферометр с одним транспортным волокном

В этом варианте отправитель и получатель имеют идентичные неравноплечие интерферометры Маха–Цандера (красным цветом отмечены зеркала). Разность фаз длинного и короткого путей ΔT намного больше времени когерентности светового источника. По этой причине интерференции в пределах малых интерферометров не происходит (Б). Но на выходе интерферометра получателя она возможна (В). Вероятность того, что фотонные амплитуды сложатся (центральный пик выходного сигнала интерферометра В) равна

$$P = \left(\frac{1}{8}\right)[1 + \cos(\Phi_A - \Phi_B)].$$

Следует заметить, что эта амплитуда сигнала в четыре раза меньше, чем в случае, показанном на рис. 2.8. Разветвители пучка (полупрозрачные зеркала) могут быть заменены на оптоволоконные объединители (*coupler*). Практические измерения для транспортного кабеля длиной 14 км показали эффективность генерации бита ключа на уровне $2{,}2 \cdot 10^{-3}$ при частоте ошибок (*BER*) около 1,2 %.

Контрольные вопросы

По разделам: «Виды и классификация аппаратно-программных средств защиты компьютерной информации», «Аппаратно-программные средства защиты компьютерной информации»:

1. Как подразделяются аппаратно-программные средства защиты информации?
2. Что такое «имитовставка»?
3. Какие компьютерные системы считаются полностью контролируемыми?
4. В чем принципиальное различие между программными и аппаратными способами реализации функций КС?
5. Какие функции криптографичекой защиты данных выполняют аппаратные средства серии КРИПТОН?
6. Как обеспечивается безопасность в частично контролируемых КС?
7. Какие средства КЗИ выпускает фирма АНКАД?

По разделу «Аппаратно-программные средства защиты компьютерной информации»:

1. Какие возникают проблемные места при шифровании данных?
2. Для каких целей применяется ПСКЗИ ШИПКА?

3. Что такое Линейка СЗИ НСД семейства АККОРД™?
4. Какие ПАК СЗИ вы знаете и в чем их суть?
5. Что такое устройство блокировки *USB*?

По разделу «Квантовая криптография»:

1. На чем базируется надежность квантового метода рассылки ключей?
2. В чем заключается квантовый принцип неопределенности?
3. Что такое и как применяется поляризация фотонов?
4. В чем сущность протокола Беннета?
5. Что такое эффект *EPR*?
6. От чего зависит вероятность регистрации фотона?
7. В чем суть алгоритма B92?
8. Как работает интерферометр с одним волокном?

ЛИТЕРАТУРА

1. Метлицкий, Ю. В. Стандарт криптографической защиты *AES*. Система Visual AES [Электронный ресурс] / Ю.В. Метлицкий – 2003. – Режим доступа: http://www.winaes.narod.ru. – Дата доступа: 04.02.2010.

2. Романец, Ю.В. Защита информации в компьютерных системах и сетях / Ю.В. Романец, П.А. Тимофеев, В.Ф. Шаньгин – М: Радио и связь, 2001. – 380 с.

ПРИЛОЖЕНИЕ

Обучающая программа PC AES.exe