УДК 681.3.06:519.248.681

И. Д. ГОРБЕНКО, д-р техн. наук, Л. В. СКРЫПНИК, д-р техн. наук,
С. А. ГОЛОВАШИЧ, Т. А. ГРИНЕНКО

СТАНДАРТ СИММЕТРИЧНОГО ШИФРОВАНИЯ 21 ВЕКА: СВОЙСТВА, РЕЖИМЫ РАБОТЫ, РЕАЛИЗАЦИЯ

Введение

Мировое информационное сообщество встретило 21 век важным событием – успешным завершением трехлетнего проекта общественного создания и принятия в качестве стандарта 21 века алгоритма симметричного шифрования. Им стал один из 15-ти кандидатов – алгоритм RIJNDAEL [1-6], авторы Joan Daemen и Vincent Rijmen. Проект создания стандарта симметричного шифрования был инициирован Национальным Институтом Стандартов и Технологий (NIST) США в 1997г. Было организовано и проведено три этапа рассмотрения, анализа и обсуждения представленных кандидатов. В 1998 г. на Первой Конференции принято решение о приеме к открытому обсуждению 15 пакетов описаний кандидатов. В результате общественного обсуждения с учетом представленных на Вторую Конференцию материалов исследований, замечаний и комментариев, число кандидатов уменьшилось до 5. В качестве кандидатов остались MARS, RC6, RIJNDAEL, SERPENT и TWOFICH. В апреле 2000 г. на Третьей Конференции были рассмотрены все поступившие материалы и проведено широкое обсуждение свойств и характеристик кандидатов. Были учтены предложения и мнения ведущих специалистов криптологов, а также результаты, полученные сотрудниками NIST США.

При отборе оценивались:

- реальная защищенность алгоритмов от криптоаналитических атак;
- статистическая безопасность криптографических алгоритмов;
- надежность математической базы криптоалгоритмов;
- вычислительная сложность (скорость) выполнения зашифрования и расшифрования;
- сложность программной, аппаратной и аппаратно-программной реализации;
- вычислительная сложность (скорость) развертывания ключей;
- возможность работы с различными длинами информационных блоков и исходных ключей;
- возможность реализации на существующем спектре программных платформ и приложений;
- возможность применения алгоритма во всех рекомендуемых режимах работы – блочного шифрования, поточного шифрования, поточного шифрования с обратной связью, выработки кодов аутентификации, хеширования, а также генератора псевдослучайных чисел;
- эквивалентность сложности программной, аппаратной и аппаратно-программной реализаций и др.

Материалы исследований, комментарии, замечания, а также мнения экспертов и организаций представлены на сайте Третьей Конференции [5]. В результате голосования участников Конференции голоса распределились следующим образом:

MARS - 13, RC6 - 23, RIJNDAEL - 86, SERPENT - 59, TWOFICH - 31.

Таким образом, специалисты-криптологи, участвующие в работе Третьей конференции, высказались за принятие в качестве стандарта симметричного шифрования алгоритма RIJNDAEL. NIST после дополнительных исследований и тестирования рекомендовал RIJNDAEL в качестве стандарта 21века.

Необходимо подчеркнуть важность и отметить высокую эффективность реализации проекта AES. По существу за три года выполнения проекта сделан прорыв в области проектирования, тестирования и методик анализа свойств и характеристик блоковых симметричных криптоалгоритмов. Необходимо отметить важность этого проекта для Украины. Специалисты получили доступ к методическому аппарату универсальных методик и тестов, средств их реализации. Безусловно, что как сами алгоритмы, так и научно-методический аппарат, созданный в процессе их создания, еще долго будут в поле зрения ученых-криптологов и практиков. В настоящей статье ставится, прежде всего, задача ознакомления специалистов в области информационной безопасности, а также практиков с криптографическим алгоритмом блочного симметричного шифрования, рекомендованного в качестве стандарта 21 века. Вместе с тем в статье приводится ряд оценок и результатов анализа свойств алгоритма.

1. Общая характеристика криптоалгоритма

Алгоритм RIJNDAEL (RD) является блоковым симметричным криптоалгоритмом. Длина информационного блока l_n может быть равной 128, 192 или 256 бит. Криптографические преобразования в алгоритме осуществляются за счет преобразования блоков информации с использованием ключевых данных. Ключ, вводимый в средства реализации RIJNDAEL, называют исходным ключом K_u. Разрешенными длинами исходного ключа есть l_{k_u}=128, 192 и 256 бит. Ключи, используемые в циклах преобразования, формируются из исходного K_u. Для этого выполняется процедура развертывания из исходного цикловых ключей. Число развернутых ключей N_p определяется как $N_p = n_ц + 1$, а длина l_p развернутого ключа как $l_p = (n_ц + 1)l_u$, где $n_ц$ – есть число циклов преобразования, выполняемых в алгоритме, а l_u – длина информационного блока.

Алгоритм RD может применяться в пяти режимах:

- блочного шифрования;
- поточного шифрования;
- поточного шифрования с обратной связью;
- выработки ключевой хеш-функции (кода-аутентификации);
- генератора псевдослучайных последовательностей.

2. Представление преобразуемой информации и ключей, цикл преобразования

В алгоритме RD преобразования выполняются при задании длины блоков информации l_u=128, 192, 256 бит и длины исходных ключей l_{k_u}=128, 192 и 256 бит. Необходимая стойкость в алгоритме обеспечивается за счет многоциклового преобразования, причем, в каждом из циклов применяются табличные и ключевые преобразования, т.е. преобразования по фиксированным таблицам и развернутым ключам. Пусть необходимо зашифровать A_i блок информации длиной соответственно 128 (192 или 256 бит). Назовем значение $A_i = M_i$, где M_i – значение зашифровываемого блока, начальным состоянием. Далее в зависимости от номера цикла j преобразования состояние (state) будем обозначать как A_{ij}, представляя его в виде матрицы байтов. Входные данные A_i и выходные C_i рассматриваются как одномерные массивы из 8-битовых слов (байтов), пронумерованные по столбцам от 0 до $4n_c - 1$, где n_c есть количество столбцов.

Для l_u=128 бит (16 байтов)
$$A_{ij} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix} \quad (1)$$

Для l_u=192 бита (24 байтов)
$$A_{ij} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{vmatrix} \quad (2)$$

Для l_u=256 бит (32 байтов)
$$A_{ij} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \end{vmatrix} \quad (3)$$

Таким образом, состояние A_{ij} описывается матрицей байтов, в которой четыре строки и разное число столбцов – 4, 6 и 8.

Аналогично состоянию информационного блока задается и исходный ключ k_u, например, для длины l_{k_i} =128 бит

$$k_u = \begin{vmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{vmatrix} \qquad (4)$$

Для длин ключей k_u, равных 192 и 256 бит, исходный ключ задается аналогично (4).

Таблица 1

$l_k \backslash l_м$	128	192	126
128	10	12	14
192	12	12	14
256	14	14	14

В алгоритме RD число циклов $n_ц$ преобразования зависит от длины информационного блока $l_м$ и длины исходного ключа l_{k_u}. В табл. 1 приведено значение $n_ц$ циклов преобразования как функции $l_м$ и l_{k_u}.

В каждом из циклов преобразования (за исключением последнего) выполняется три табличных преобразования и одно криптографическое. В процессе преобразования байты считываются по столбцам, т.е. $a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, \ldots$ и т.д. На языке псевдо-Си один цикл имеет вид

```
Round(State); //циклы 1, n_ц − 1
{
  Bytesub(State); //замена байт
  ShiftRow(State); //сдвиг строчек
  MixColumn(State); //перемешивание в столбцах
  AddRoundKey(State, RoundKey); //сложение с ключом
}
```

На последнем цикле преобразование MixColumn отсутствует, т.е. перемешивание в столбцах не выполняется. Алгоритм RD блочного зашифрования на $n_ц$ представлен на рис. 1.

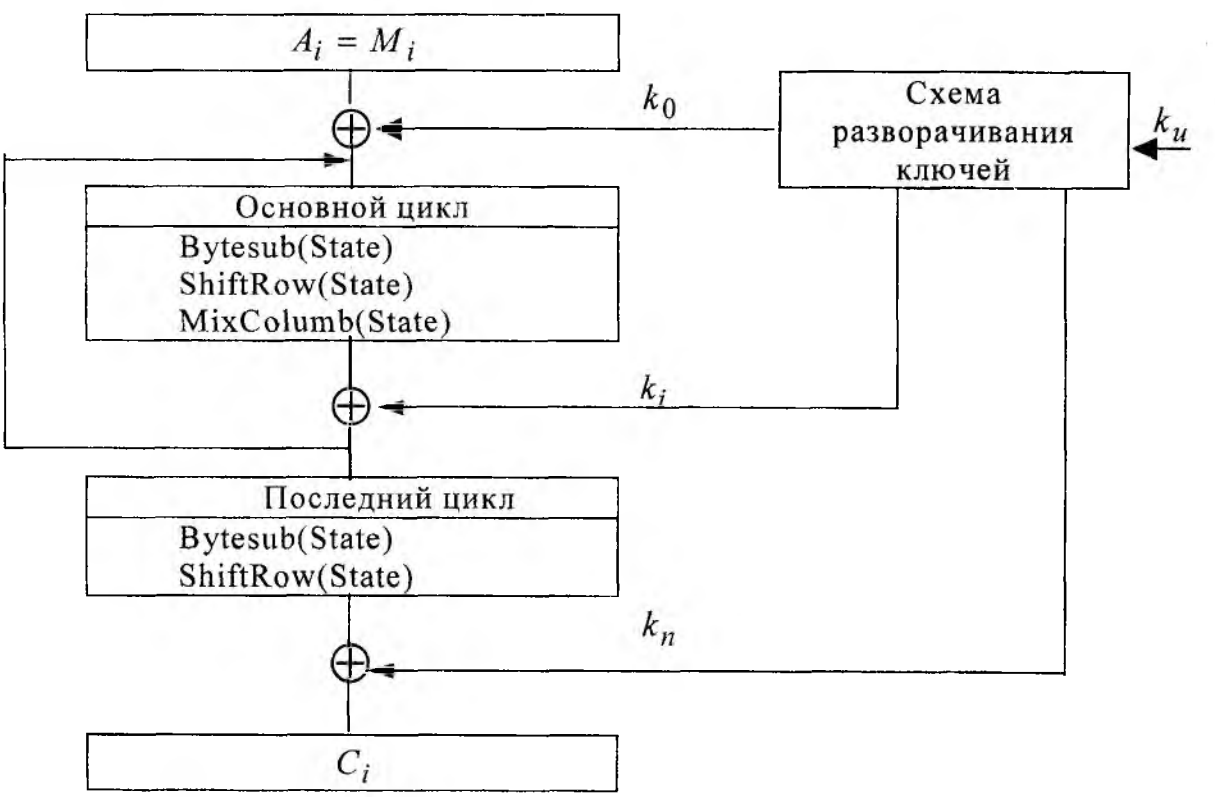


Рис. 1

3. Табличные и криптографические преобразования

Преобразование Bytesub (замена байт) математически может быть представлено в виде двух преобразований. При первом преобразовании каждый байт состояния a_{ij} заменяется мультипликативно обратным элементом a_{ij}^{-1} в поле Галуа GF(2^8), т.е.

$$a_{ij} \cdot a_{ij}^{-1} \equiv 1 \left(\mathrm{mod}\, f(x) = x^8 + x^4 + x^3 + x + 1\right) \tag{5}$$

Второе преобразование обеспечивает аффинное преобразование $a_{ij}^{-1}\left(x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 = x\right)$ по правилу:

$$Y = C \cdot x + C_1 \left(\mathrm{mod}\, x^8 + 1\right) \tag{6}$$

В матричном представлении (6) имеет вид:

$$\begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{vmatrix} \oplus \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix} \tag{7}$$

Преобразования (5) и (7) можно представить в виде одной таблицы подстановки типа «байт в байт». Если состояние A_i имеет вид (2), то после преобразования «замена байт» (Bytesub) получим состояние A_i', что можно представить преобразованием последовательной замены байт согласно таблице замены (S-box). На рис. 2 показано, как a_{23} байт заменяется по таблице на a'_{23}:

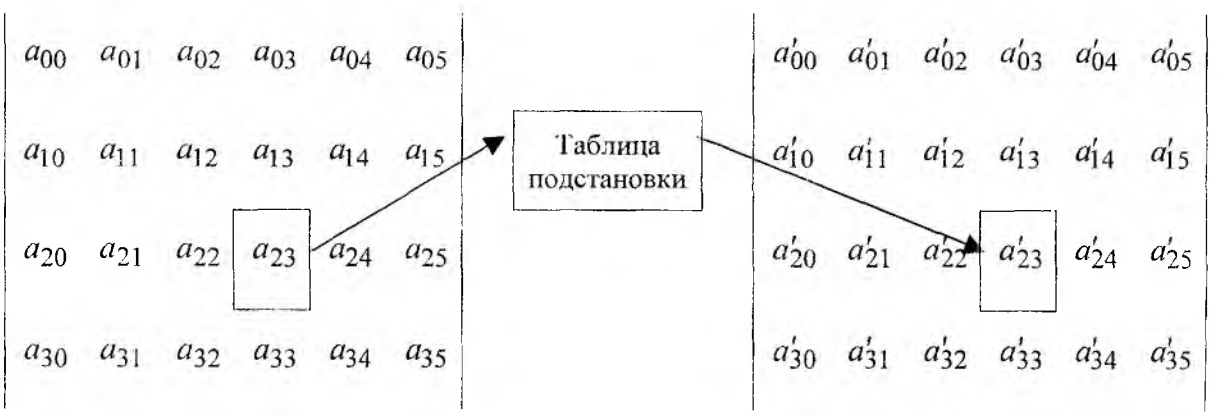


Рис. 2

Преобразование ShiftRow (Сдвиг строчек) обеспечивает циклический сдвиг строчек состояния. При этом первая строка циклически не сдвигается, а вторая, третья и четвертая циклически сдвигаются на величину, указанную в табл. 2. Причем, величины сдвига C_1, C_2, C_3 зависят от числа столбцов n_c состояния $A_i((1),(2)\,или\,(3))$.

Таблица 2

n_c	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

На рис. 3 приведен пример преобразования «сдвиг строчек» для состояния A_i (1), длина блока состояния 128 бит (16 байт).

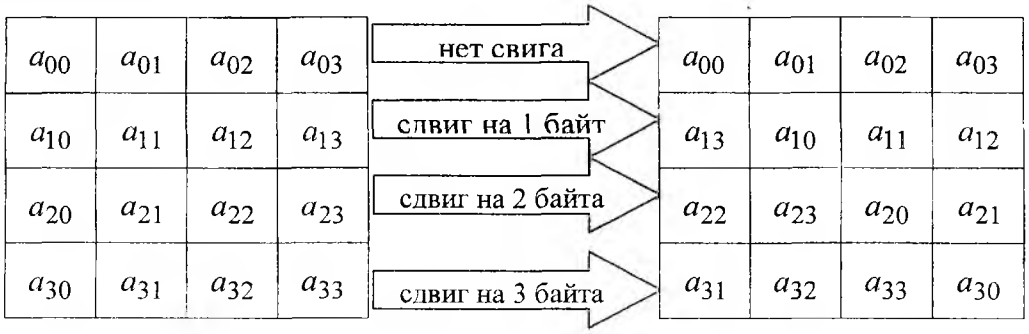


Рис. 3

Преобразование типа «перемешивание в столбцах» обеспечивает последовательное табличное преобразование элементов-байтов столбцов. Столбцы, всегда состоящие из четырех байт, представляются полиномами третьей степени:

$$a(x) = a_{3j}x^3 + a_{2j}x^2 + a_1 x + a_0 \left(\mathrm{mod}\, x^4 + 1\right),\ j = \overline{0, n_c}\,. \tag{8}$$

Коэффициенты $a(x)$ принимают в (8) значения в интервале от нуля (00000000) до 255 (11111111).

Перемешивание производится умножением $a(x)$ на константу

$$C(x) = '03'x^3 + '01'x^2 + '01'x + '02'\,, \tag{9}$$

т.е. преобразованный столбец $a'(x)$ есть

$$a'(x) = C(x)\cdot a(x)\left(\mathrm{mod}\, x^4 + 1\right) \tag{10}$$

В матричном виде преобразование (10) имеет вид:

$$\begin{vmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix} \tag{11}$$

На рис. 4 показано преобразование типа «перемешивание в столбцах» для A_i состояния вида (1).

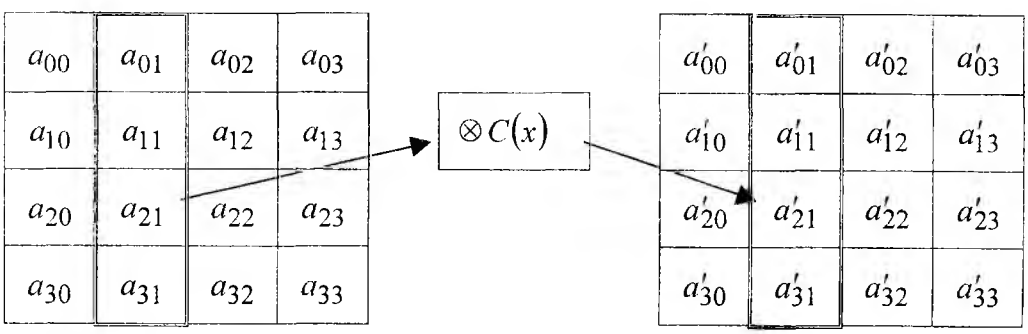


Рис. 4

Преобразование, представленное на рис. 4, может быть выполнено с использованием заранее составленной таблицы.

Криптографическое преобразование в алгоритме RD осуществляется посредством сложения по модулю 2 A_i состояния и k_ν ключа ν-го цикла преобразования. В общем виде это преобразование можно представить как:

$$A'_{ij} = A_{ij} \oplus K_{\nu j} \tag{12}$$

В матричном виде для состояния (1) его можно представить как:

$$\begin{vmatrix} a'_{00} & a'_{01} & a'_{02} & a'_{03} \\ a'_{10} & a'_{11} & a'_{12} & a'_{13} \\ a'_{20} & a'_{21} & a'_{22} & a'_{23} \\ a'_{30} & a'_{31} & a'_{32} & a'_{33} \end{vmatrix} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix} \oplus \begin{vmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{vmatrix} \tag{13}$$

Таким образом, зашифрование производится посредством побитного сложения бит A_i состояния и битов k_ν развернутого ключа.

После окончания процедуры зашифрования блок-криптограмма C_i представляет собой состояние, одномерный массив байтов которого формируется считыванием столбцов матрицы $C_{0,0} = a'_{00}, C_{1,0} = a'_{10}, C_{3,0} = a'_{30}, C_{0,1} = a'_{01}, C_{0,2} = a'_{02}, \ldots$.

4. Алгоритмы выработки цикловых ключей

В алгоритме RD применяется принцип разворачивания цикловых ключей $k_ц$. Сущность его заключается в том, что необходимое число цикловых ключей $k_ц$ вырабатывается из исходного ключа $K_и$. Как исходные, так и цикловые ключи представляются в виде одномерных массивов, например, $k_{00}, k_{10}, k_{20}, k_{30}, k_{01}, k_{11}, \ldots$ и т.д., причем, преобразование в одномерный массив производится последовательным считыванием по столбцам матричного представления ключа. Исходный ключ может иметь длину 128, 192 или 256 бит. Он может быть представлен соответственно 4(16), 6(24) и 8(32) столбцами (байтами). Развернутые цикловые ключи всегда имеют длину, равную длине информационного блока (13).

Для выработки цикловых ключей в алгоритме RD рекомендуется применять два алгоритма: первый, если длина исходного ключа $n_{k_и} \le 6$, и второй, если $n_k = 8$. Значение $n_{k_и}$ указано в числе столбцов (32 битных или 4 байтовых слов). Рекомендуемое число циклов преобразования как функция длины информационного блока $l_м$ и длины исходного ключа $k_и$ приведено в табл. 1.

Алгоритм 1. Исходный ключ представляет собой массив байтов размерности $4n_{k_и}$, $n_{k_и} = 4$ *или 6*. Расширенный массив развернутых ключей представляет собой массив размерности $n_м(n_ц + 1)$ 32-битных слов или $4n_м(n_ц + 1)$ байтов, где $n_м$ - число столбцов информационного блока (может быть 4, 6 или 8).

Исходный ключ задается массивом байтов $k_и$ размера $4n_{k_и}$, а развернутый – $k_р$ размера $n_м(n_ц + 1)$. Вначале исходный ключ переписывается в качестве значений первых байтов (слов) разворачиваемого. Затем производится развертывание остальных цикловых ключей.

Алгоритм развертывания ключа на языке псевдо-Си имеет вид ($n_r = n_ц$):

```
for (i = nk; i<nм×(nr+1); i++)
{
   temp = Kp[i-1];
   if (i%nk == 0)
   temp = SubByte(RotByte(temp)⊕Rcon[i/nk]);
   Kp[i] = K[i-nk]⊕temp;
}
```

В алгоритме функция SubByte($K_р$) формирует 4-байтовое слово, каждый байт которого в свою очередь формируется применением табличного преобразования ByteSub, задаваемого правилами (5)

и (7). Функция RotByte(K_p) производит циклический сдвиг входных байтов влево, например, RotByte(a,b,c,d)=(b,c,d,a).

Таким образом, при развертывании исходного ключа и формировании цикловых ключей в качестве первых n_k слов записывается исходный ключ. Каждое последующее слово $K_p(i)$ формируется на основе сложения по модулю 2 $K_p[i-n_k]$ слова и $K_p[i-1]$ слова. При этом для слов $i \equiv 0 (\mathrm{mod}\ n_k)$, т.е. кратных длине исходного ключа n_k, $K_p[i-1]$ слово преобразуется с использованием функций SubByte, RotByte, а также складывается с константой Rcon. Циклические константы не зависят от n_k и задаются как Rcon[i]=(RC[i],'00','00','00') (14), а R[i] вычисляется с использованием образующего элемента x='02', причем

$$RC[1]='01',\ a\ RC[i]=x\cdot[RC(i-1)]=x^{i-1}. \qquad (15)$$

Алгоритм 2, приведенный ниже, рекомендуется использовать при $n_k > 6$ $(n_k = 8)$.

Алгоритм 2

```
for (i = nk; i<nм×(nr+1); i++)
{
   temp = Kp[i-1];
   if (i%nk == 0)
   temp = SubByte(RotByte(temp)⊕Rcon[i/nk]);
   else if (i%nk == 4)
   temp = SubByte(temp);
   Kp[i] = K[i-nk]⊕temp;
}
```

Подчеркнем, что в алгоритме 2, как и в алгоритме 1, исходный ключ записывается в качестве первых слов расширенного ключа. Кроме того, в случае наличия ограничений на память, цикловые ключи могут формироваться динамически в каждом цикле соответственно.

Выбор цикловых ключей из массива Kp производится в каждом из циклов последовательно. При этом каждый раз выбирается цикловый ключ $l_ц$ длиной, равной длине информационного блока $l_м$.

5. Обратные преобразования (расшифрование)

Алгоритм RD в явном виде не является симметричным. Поэтому расшифрование должно производиться в обратном порядке. На рис. 5 приведена структурная схема расшифрования C_i блока.

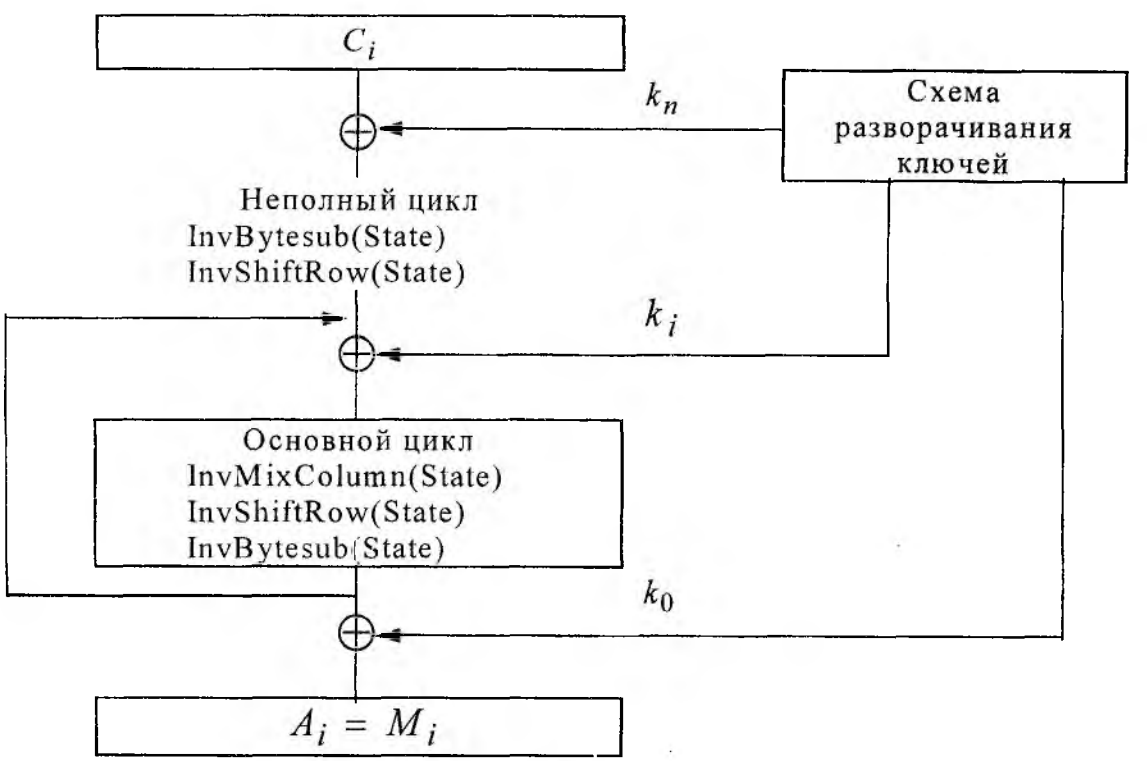


Рис. 5

В алгоритме используются инверсные преобразования Inv, отличительной особенностью которых является то, что при их выполнении используются другие таблицы преобразований.

На рис. 6 приведен процесс, поясняющий выбор цикловых ключей из развернутого ключа, при длине циклового ключа равной 256 бит.

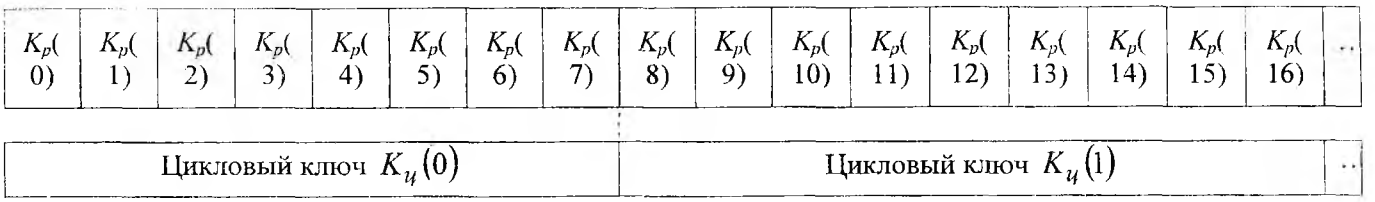


Рис. 6

Как показали исследования, на применяемые в RD исходные ключи не накладывается никаких ограничений. Поэтому можно предположить, что слабых ключей для этого алгоритма не существует. Следует заметить, что авторы RD алгоритма не представили обоснования алгоритмов разворачивания исходного ключа в цикловые. На наш взгляд, в качестве алгоритма разворачивания может использоваться любой алгоритм, который формирует $n_ц$ независимых случайных и равновероятных цикловых ключей.

В алгоритме зашифрования (рис. 1) в качестве первого используется нелинейное преобразование Bytesub(замена байт). Затем используются преобразования «сдвиг строчек» и «преобразование столбцов». При обратном преобразовании, т.е. расшифровании, порядок преобразований в цикле должен быть обратным. Вначале выполняется обратное преобразование в столбцах MixColumb, затем – обратное преобразование сдвиг строчек, последним – обратное нелинейное преобразование Bytesub. Анализ показывает, что для того, чтобы обратное преобразование (расшифрование) могло быть выполнено в той же последовательности, что и прямое (т.е. зашифрование), должна существовать таблица замены байт, реализующая сразу три шага – Bytesub, ShiftRow и MixColumb, а также обратная ей. Как удалось установить, это не так. Поэтому обратное преобразование в циклах должно выполняться в другом порядке – обратном. Следовательно, при расшифрвании нельзя использовать алгоритм, представленный на рис.1, а только алгоритм, представленный на рис.5. При этом таблицы обратных преобразований MixColumb, ShiftRow и Bytesub также должны быть другими, но однозначно связанными с прямыми.

Анализ алгоритма RD показывает, что порядок выполнения преобразований ShiftRow и Bytesub в нем может быть любой. Это объясняется тем, что преобразование ShiftRow только перемещает байты и не влияет на содержание (значение) байта. Преобразование Bytesub обеспечивает замену только отдельных байт независимо от их положения.

Кроме того, последовательность преобразования

```
AddRoundKey(State, RoundKey);      // Сложение состояния с цикловым ключом
InvMixColumn(State);               // Обратное перемешивание в столбцах
                                   может быть заменена на последовательность
InvMixColumn(State);               // Обратное перемешивание в столбцах
AddRoundKey(State, InvRoundKey);   //Сложение состояния с инверсным цикловым
                                    ключом.
```

Инверсный цикловый ключ можно получить посредством применения инверсного перемешивания в столбцах к соответствующему цикловому ключу RoundKey. Это объясняется тем, что названное A-преобразование линейное и A(x+y)=A(x)+A(y).

В целом обратное преобразование криптоалгоритма RD может быть задано следующим образом:

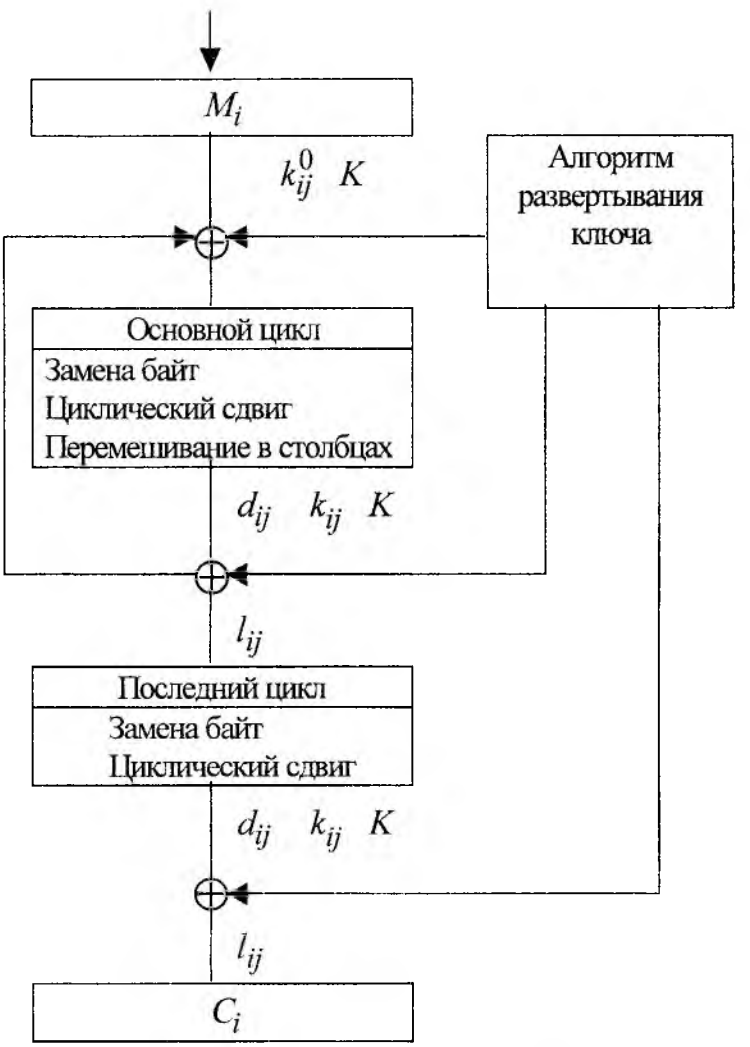
```
I_RD(State, CipherKey)                    // I_RD(состояние, исходный ключ шифрования)
{
I_KeyExprension(CipherKey, I_ExpandedKey); // I_Расширение ключа (исходный ключ,
                                           развернутые цикловые ключи)
```

```
AddRoundKey(State, I_ExpandedKey+nc*nr);        // Сложение по модулю 2 состояния с цикловым ключом

For (i = nr-1); I>0;i--)
Round(State, I_ExpandedKey+nc*i);               // Выполнение nr циклов
FinalRound(State, I_ExpandedKey);               // Выполнение последующего цикла
}
```

6. Организация табличных преобразований

При практической реализации алгоритма шифрования одним из наиболее важных требований является требование минимизации сложности зашифрования/расшифрования. На рис. 7 приведена схема алгоритма зашифрования с более подробным, чем на рис. 1, описанием преобразований. Здесь также рассмотрим циклы преобразования, основное внимание уделив на организацию идентичных операций в циклах.



$$a_{ij} = M_{ij} \oplus K_{ij}^0$$
$$b_{ij} = S\lfloor a_{ij} \rfloor$$
$$C_{ij} = b_{ij}'$$
$$d_{ij} = c_{ij} \cdot C$$
$$l_{ij} = d_{ij} \oplus K_{ij}$$
$$b_{ij} = S\lfloor a_{ij} \rfloor$$
$$C_{ij} = b_{ij}'$$
$$l_{ij} = C_{ij} \oplus K_{ij}^n$$
$$C_j = l_{ij}$$

Рис. 7

Рассмотрим основной цикл в обратном порядке преобразования, опираясь на уже приведенный выше материал. Байты состояния будем задавать a_{ij} (байт i-й строки и j-го столбца). Соотношение $l_{ij} = d_{ij} \oplus K_{ij}$ представим в развернутом виде – столбцами состояний, начиная с l_{ij} :

$$\begin{bmatrix} l_{0j} \\ l_{1j} \\ l_{2j} \\ l_{3j} \end{bmatrix} = \begin{bmatrix} d_{0j} \\ d_{1j} \\ d_{2j} \\ d_{3j} \end{bmatrix} \oplus \begin{bmatrix} k_{0j} \\ k_{1j} \\ k_{2j} \\ k_{3j} \end{bmatrix}, j = \overline{0, n_c - 1}. \tag{16}$$

Далее d_{ij} состояние представим как результат выполнения преобразования «перемешивание в столбцах», причем, входным является состояние C_{ij}:

$$\begin{bmatrix} d_{0j} \\ d_{1j} \\ d_{2j} \\ d_{3j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} C_{0j} \\ C_{1j} \\ C_{2j} \\ C_{3j} \end{bmatrix}, \quad j = \overline{0, n_c - 1}. \tag{17}$$

В (17) матрица перемешивания представлена константой C. Циклический сдвиг строк, т.е. состояние C_{ij}, положив, что входом циклического преобразования есть состояние a_{ij}, представим в виде:

$$\begin{bmatrix} C_{0j} \\ C_{1j} \\ C_{2j} \\ C_{3j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-C_1} \\ b_{2,j-C_2} \\ b_{3,j-C_3} \end{bmatrix}, \quad j = \overline{0, n_c - 1}. \tag{18}$$

Константы C_1, C_2 и C_3 выбираются из табл. 1.

Табличное преобразование «замена байт» представим в виде последовательной замены a_{ij} байт по таблице подстановки S, при этом выходом является состояние b_{ij}, причем:

$$b_{ij} = S[a_{ij}], \quad i = \overline{0,3}, \; N = \overline{0, n_c - 1}. \tag{19}$$

Преобразование (19) представлено одной таблицей подстановки, в которой учитываются два преобразования – замена a_{ij} байта на обратный элемент и аффинное преобразование вида (7). Возможность и алгоритм построения S-таблицы рассмотрим отдельно.

Последовательность преобразований (16) – (19) можно представить в виде одного, заменив состояние d_{ij} :

$$\begin{bmatrix} l_{0j} \\ l_{1j} \\ l_{2j} \\ l_{3j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S[a_{0j}] \\ S[a_{1j-C1}] \\ S[a_{2j-C2}] \\ S[a_{3j-C3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0j} \\ k_{1j} \\ k_{2j} \\ k_{3j} \end{bmatrix}. \tag{20}$$

Далее, учитывая то, что умножение производится по правилу «строка на столбец» (20) представим в виде:

$$\begin{bmatrix} l_{0j} \\ l_{1j} \\ l_{2j} \\ l_{3j} \end{bmatrix} = \begin{bmatrix} S[a_{0j}] \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \oplus S[a_{1j-C1}] \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \oplus S[a_{2j-C2}] \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \oplus S[a_{3j-C3}] \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \end{bmatrix} \oplus \begin{bmatrix} k_{0j} \\ k_{1j} \\ k_{2j} \\ k_{3j} \end{bmatrix}. \tag{21}$$

В выражении введены скобки [ ], которые указывают на необходимость выполнения операций сначала в квадратных скобках. Вычисления значения $[l_{ij}]$ можно свести к построению четырех таблиц.

$$T_0[a]=\begin{bmatrix} S[a]\cdot 02 \\ S[a] \\ S[a] \\ S[a]\cdot 03 \end{bmatrix}; T_1[a]=\begin{bmatrix} S[a]\cdot 03 \\ S[a]\cdot 02 \\ S[a] \\ S[a] \end{bmatrix}; T_2[a]=\begin{bmatrix} S[a] \\ S[a]\cdot 03 \\ S[a]\cdot 02 \\ S[a] \end{bmatrix}; T_3[a]=\begin{bmatrix} S[a] \\ S[a] \\ S[a]\cdot 03 \\ S[a]\cdot 02 \end{bmatrix}. \quad (22)$$

В выражениях (22) умножения выполняются побайтно и по модулю $m(x)=x^8+x^4+x^3+x+1$. Причем, вместо операции замены конкретного значения байта записана полная таблица замены байт. Это позволяет выполнить операцию умножения таблично. Всего нужно 4 таблицы T_v по 4 таблицы замены в каждой из них. Общий объем 4 таблицы. $N_T=4\cdot 4\cdot 2^8=2^{12}=4096$ *байт*.

Преобразование (21) с учетом (22) можно представить:

$$l_j=T_0(a_{0j})\oplus T_1(a_{1j-C1})\oplus T_2(a_{2j-C2})\oplus T_3(a_{3j-C3})\oplus K_j,\ j=\overline{0,n_c-1}. \quad (23)$$

Операцию нахождения T_v будем называть табличным поиском. Следует учитывать, что каждая из таблиц задается 4-х байтовым числом, результатом l_j также будет 4-х байтовое значение l_j. Всего за цикл вычисления необходимо выполнить 4 операции сложения по модулю 2 и 4 поиска по таблицам T_0, T_1, T_2 и T_3.

Анализ показывает, что таблицы T_1, T_2 и T_3 отличаются от T_0 только по циклическим сдвигам на C_1, C_2 и C_3 байт. Поэтому вычисления (23) можно свести к рекуррентному определению согласно выражению:

$$l_j=k_j\oplus T_0\lfloor b_{0j}\rfloor\oplus R_b\left(T_0\lfloor b_{1,j-C1}\rfloor\oplus R_b\left(T_0\lfloor b_{2,j-C2}\rfloor\oplus R_b\left(T_0\lfloor b_{3,j-C3}\rfloor\right)\right)\right). \quad (24)$$

В (24) используются 4-х байтовые слова, полученные после циклического сдвига на величины C_1, C_2 и C_3 соответственно b_{1j}, b_{2j} и b_{3j}.

При этом вычисление (24) по сравнению с (23) требует трех дополнительных операций циклического сдвига. Но в этом случае достаточно одной таблицы преобразования длиной $N_T=4\cdot 2^8=2^{10}=1024$ *байт*.

7. Анализ стойкости криптоалгоритма

В процессе всех этапов общественного обсуждения и анализа алгоритма RD, а также исследования и тестирования его со стороны NIST США, особое внимание уделялось анализу и оценке по критериям реальной криптозащищенности, статистической безопасности и надежности математической базы криптоалгоритма [1-3]. На сегодняшний день не известно, а возможно и не существует ни одного метода криптоанализа, сложность реализации которого была бы меньше, чем методы, основанные на «грубой силе» (переборе). Тем не менее, RD алгоритм оказался защищенным от атак, реализованных на основе:

- дифференциального криптоанализа;
- поиска наилучшей дифференциальной характеристики;
- расширения для дифференциального криптоанализа;
- линейного криптоанализа;
- вторжения с использованием связанных ключей;
- вторжения с частичным угадыванием ключа;
- вторжения на основе обработки сбоев;
- интерполяционного (алгебраического) вторжения;
- поиска лазеек.

Основу криптозащищенности в алгоритме составляют табличные и криптографические преобразования на множестве циклов. В качестве табличных преобразований используются преобразования (замена) байт, преобразование (сдвиг) строк и преобразование (перемешивание) в столбцах. При этом наибольший вклад в криптостойкость вносят преобразования вида (5) и (6): преобразование (5) ввиду нелинейности обеспечивает защиту от линейных и дифференциально-подобных атак, а (6) – от раз-

личных алгебраических вторжений. Оба преобразования, как следует из (23), обеспечивают табличную нелинейную многократную замену байт. Многократное выполнение замены байт совместно с преобразованием строк и столбцов, а также зашифрование с использованием цикловых ключей, дают определенную уверенность в стойкости криптоалгоритма RD.

Что касается поиска лазеек, то пока в самом алгоритме их обнаружить не удалось. На наш взгляд, лазейки могут быть встроены в алгоритм формирования исходных ключей или в алгоритм развертывания ключей, т.е. формирования цикловых ключей.

Статистическая безопасность проверялась по методике, изложенной в [6]. В табл. 3 в качестве примера приведены статистики (не лучшие) связанности C_i и M_i (функция F_1), а также зависимости C_i и C_k (функция F_2) при различных фактических значениях данных M_i и исходных ключей k_u. Причем, $\overline{m}(F_r)$ - среднее значение, а $m^2(F_r)$ - дисперсия, вычисляемые в метрике Хэмминга (по числу совпадающий битов).

Таблица 3

Варианты	$m(F_1)$	$m^2(F_1)$	$m(F_2)$	$m^2(F_2)$	min F_1/max F_1	min F_1/max F_1
1. M_i – случайные, k_u - случайный постоянный	64,16	32,25	63,95	32,12	44/90	44/85
2. M_i – случайные, k_u - единичный постоянный	63,79	31,86	64,03	31,07	39/90	44/85
3. M_i – единичное, k_u - случайный постоянный	63,99	32,40	63,97	32,30	39/90	42/86
4. M_i – нулевые, k_u - случайный изменяемый	63,98	33,47	64,05	31,92	39/90	42/86
5. M_i – случайные, k_u - случайный изменяемый	63,88	31,70	64,00	30,67	39/90	42/86

Анализ данных таблицы позволяет высказать гипотезу о биноминальном законе распределения функций F_1 и F_2, так как биномиального закона распределения математическое ожидание $\overline{m}(F_v)$ и $m^2(F_v)$ соответственно равны 64 и 32. В то же время величины, подчиняющиеся биномиальному закону распределения, являются случайными и равновероятными.

Таким образом, алгоритм RD можно с определенной уверенностью считать статистически безопасным.

Что касается доказательства надежности математического аппарата, то это сложная и отдельная проблема. Ее надо решать, начиная с обоснования терминологии, понятий, критериев, методов и методик и т.д.

8. Анализ алгоритмов (схем) развертывания ключей

К алгоритмам развертывания ключей предъявляются требования по двум критериям:

- случайность, равновероятность и независимость цикловых ключей;
- минимизация сложности развертывания цикловых ключей.

В табл. 4 приведены значения сложности развертывания цикловых ключей для различных длин исходных ключей.

Таблица 4

Длина ключа	128	192	256
Количество тактов	2066	2418	2937

Важной характеристикой, которая существенно влияет на выбор того или иного криптоалгоритма, является сложность (скорость) прямых и обратных криптографических преобразований. При выборе алгоритма эта характеристика может быть решающей. Результаты исследований в этом направлении представлены отдельной статьей.

В процессе анализа AES мы пришли к выводу, что определенные требования должны быть предъявлены и к развернутым ключам. В частности, желательно, чтобы изменение на каждой из позиции исходного ключа бита приводило к равномерному изменению битов в развернутом ключе. Были проведены экспериментальные исследования такого эффекта размножения ошибок для RD. По существу изучался эффект размножения ошибок в развернутом K^p-ключе при изменении каждого бита исходного K_j - ключа. Для всех алгоритмов процедуру разворачивания ключа можно представить в виде:

$$K_j^p = X\left(K_j, C_V\right), \tag{25}$$

где C_V – V-я константа алгоритма развертывания ключа.

Число измененных n_i битов в развернутом ключе можно представить как функцию φ номера измененного бита исходного ключа и констант, т.е.

$$n_i = \varphi\left(i, c_V\right). \tag{26}$$

Таким образом, в эксперименте изменялся i-й бит исходного ключа и определялось количество измененных битов по следующей методике:

1. Формировался K_j исходный ключ, а затем согласно (25) – соответствующий ему развернутый K_j^p ключ.
2. В каждом исходном K_j ключе изменялся i-й бит, $i = \overline{1, l_p}$, а затем, согласно (25), формировался соответствующий ему развернутый K_{j1}^p ключ, где l_p - длина развернутого ключа.
3. Вычислялось расстояние Хэмминга n_i между развернутыми ключами K_j^p для каждой i-й позиции и различных ключей.

На наш взгляд, лучшим алгоритмом развертывания ключа может быть тот, для которого, во-первых, распределение $\overline{n_i}$, $i = \overline{1, l_p}$ подчиняется равномерному закону, а во-вторых, математическое ожидание распределения $\bar{n}$ ближе к $l_p/2$. Физически в первом случае это означает, что изменение каждого бита приводит к лавинному эффекту с равномерным размножением ошибок, а во втором, что изменение одного бита приводит в среднем к изменению половины битов развернутого ключа. В целом при выполнении обоих условий развернутые ключи отличаются в половине бит, поэтому их можно считать независимыми.

На рис. 8 и 9 приведены гистограммы коэффициентов размножения ошибок в развернутом ключе в зависимости от номера позиции измененного бита в исходном ключе для алгоритмов RD и RC6 соответственно (длина ключа 128 бит).

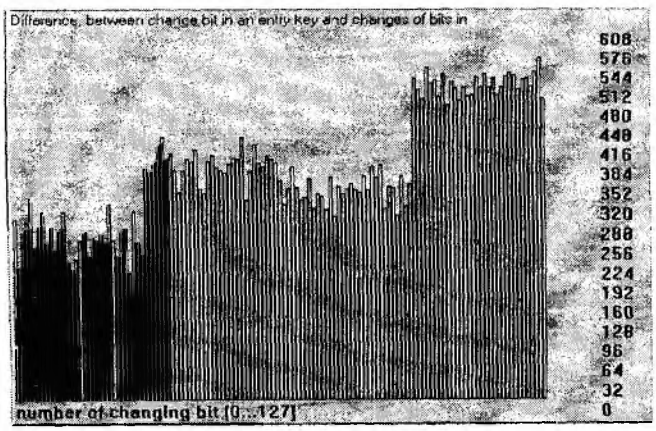


Рис. 8

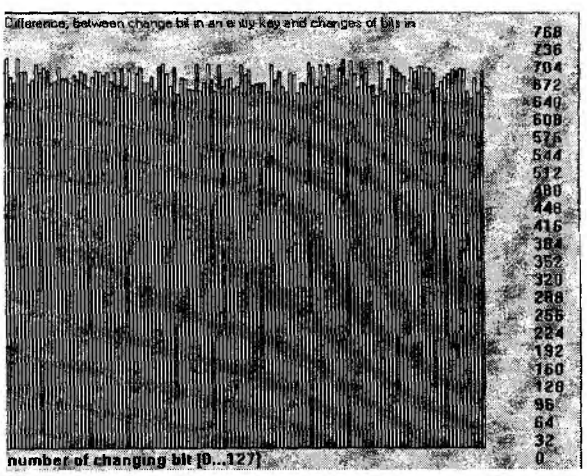


Рис. 9

Анализ данных рис. 8 показывает, что для алгоритма RIJNDAEL коэффициент размножения «ошибки» в зависимости от позиции изменяемого бита является существенно неравномерным. Природа неравномерности объясняется особенностью алгоритма разворачивания ключа.

Как следует из рис. 9, коэффициент размножения ошибок для алгоритма RC6 [3] близок к равномерному и имеет один и тот же характер для различных длин исходных ключей.

На наш взгляд, выявленный факт неравномерности размножения «ошибок» является слабой стороной схемы развертывания ключей, применяемого в RD. Если лазейка в этом алгоритме имеется, то начало ее может начинаться со значительной вероятностью с указанного недостатка.

Из рис. 9 следует, что алгоритм разворачивания ключей RC6 по статистике «размножения» ошибок лучше.

Заключение

В практической криптографии на мировом уровне успешно проведен важный научно-практический эксперимент "общественного" создания и обсуждения претендентов на стандарт симметричного шифрования 21 века. Ему предшествовала по существу неудачная попытка создания стандарта США в виде различных версий Skipjack. Явно понятно, что заказчиком такого эксперимента были США в лице NIST и других структур. Эксперимент закончен успешно. Лучшие криптологи участвовали в эксперименте, дали на обсуждение свои лучшие разработки, методики и знания. Эксперимент позволил специалистам освоить многогранные знания, методы и методики, используемые в передовых странах.

Блоковый криптоалгоритм, вернее его разработчики, обоснованно победили в конкурсе на лучший алгоритм среди кандидатов на стандарт 21 века. При современном уровне знаний алгоритм RD является защищенным от существующих аналитических атак, наименее опасной для него является атака типа "грубая сила". Алгоритм обеспечивает приемлемую скорость зашифрования/расшифрования, может быть реализован на процессорах различной разрядности.

С определенной вероятностью можно предположить, что в нем отсутствуют лазейки. Вместе с тем необходимо отметить недостаточную обоснованность используемой схемы развертывания ключа. По сравнению со схемами других кандидатов схема развертывания цикловых ключей RD уступает им. Следует предположить, что эта схема будет или должна быть усовершенствована или заменена по ряду причин.

Безусловно, что алгоритм RD еще многие годы будет подвергаться анализу и совершенствованию. Выражаем признательность специалистам и всем интересующимся практической криптографией за внимание, уделенное нашей статье.

Список литературы: 1. *Announcing Development of Federal Information Processing Standard for Advanced Encryption Standard.* Federal Register. 1997. Vol. 62, №1. Pp. 93-94. 2. *Announcing Reguest for Candidate Algorithm Nomination for Advanced Encryption Standard (AES).* Federal Register. 1997. Vol. 62, №177. Pp. 48051-48058. 3. *М.Ф. Бондаренко, И.Д. Горбенко, А.В. Потий.* Улучшеный стандарт симметричного шифрования XXI века: концепция создания и свойства кандидатов. Радиотехника. 2000. Вып. 114. С.5-15. 4. *Status report on the 2-nd round of the Development of the Advanced Encryption Standard* / AES home page. Http://www.nist.gov/aes. 5. *Status report on the 3-th round of the Development of the Advanced Encryption Standard* / AES home page. http://www.nist.gov/aes. 6. *Joan Daemen, Vincent Rijmen.* The Rijndael Block Cipher. AES Proposal: Rijndael, Document version 2. 3.09.99.

Харьковский государственный технический университет радиоэлектроники
Служба безопасности Украины

Поступила в редколлегию 18.01.2001