предъявляющих повышенные требования к безопасности конфиденциальной информации, не следует использовать шифраторы «с произвольным доступом», т. к. принцип их построения противоречит этому требованию.

Обе приведенные в статье схемы шифраторов «с последовательным доступом» удовлетворяют указанным выше требованиям и могут рекомендоваться в качестве альтернативных режимов применения блочных шифров. Стойкость приведенных схем может быть повышена путём сокращения длины блока гаммы текста, однако «ценой» за такое «усиление» будет снижение общей производительности шифратора, что для ряда приложений может быть вполне приемлемо.

Литература: 1. A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996. 2. FIPS 81, «DES modes of operation», Federal Information Processing Standards Publication 81, U. S. Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1980. 3. ANSI X3.106, «American National Standard for Information Systems – Data Encryption Algorithm – Modes of Operation», American National Standards Institute, 1983. 4. ISO 8732, «Banking – Key management (wholesale)», International Organization for Standardization, Geneva, Switzerland, 1988 (first edition). 5. ISO/IEC 10116, «Information processing – Modes of operation for an n–bit block cipher algorithm», International Organization for Standardization, Geneva, Switzerland, 1991 (first edition). 6. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. М.: Госстандарт СССР. 7. W. Diffie and M. E. Hellman, «Privacy and authentication: An introduction to cryptography», Proceedings of the IEEE, 67 (1979), pp. 397–427. 8. Ш. Х. Михелович. Теория чисел. М.: «Высшая школа», 1962. 259 с.

УДК 681.3.06:519.248.681

СТАНДАРТ СИММЕТРИЧНОГО ШИФРОВАНИЯ 21 ВЕКА RIJNDAEL

Иван Горбенко, Леонид Скрыпник, Сергей Головашич, Татьяна Гриненко

Харьковский государственный технический университет радиоэлектроники

Аннотация: **Дається опис режимів застосування, порядку організації та виконання прямих і зворотних криптографічних перетворень, а також формування циклових ключів у стандарті 21 століття – криптоалгоритмі Rijndael. Приводяться результати аналізу основних характеристик і властивостей криптоалгоритму Rijndael, режимів роботи і застосування. Указується на ряд особливостей криптоалгоритму і необхідність удосконалювання алгоритму розгортання циклових ключів, який використовується в Rijndael.**

Summary: **The modes of operation, organisational procedures, encryption and decryption transformations as well as the key schedule in the XXI century standard – the cryptographic algorithm Rijndael are given. The main characteristics and properties analysis results, modes of operation and application are given. Several features of the cryptographic algorithm and necessity of Rijndael's key schedule improving are pointed out.**

Ключевые слова: **Стандарт симметричного шифрования, зашифрование, расшифрование, криптоалгоритм.**

Введение

Успешно завершился трехлетний проект 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. При выборе стандарта 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 шифрова-

ния, поточного шифрования с обратной связью, выработки кодов аутентификации, хеширования, а также генератора псевдослучайных чисел; эквивалентность сложности программной, аппаратной и аппаратно-программной реализаций и др.

В настоящей статье ставится, прежде всего, задача ознакомления специалистов в области информационной безопасности с криптографическим алгоритмом блочного симметричного шифрования, рекомендованного в качестве стандарта 21 века, а также приводится ряд оценок и результатов анализа свойств алгоритма.

I 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.

II 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.

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

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

Для l_u =256 бит (32 байтов) A_{ij} формируется аналогично (1).

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

$$k_u = \begin{vmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{vmatrix} \tag{3}$$

Для длин ключей k_u, равных 192 и 256 бит, исходный ключ задается аналогично (3).

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

Таблица 1

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

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

Round(State); //циклы $\overline{1, n_ц - 1}$

```
{
 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.

III 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.

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

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:

$$Y = C \cdot x + C_1 \left(\text{mod } x^8 + 1 \right) \tag{5}$$

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

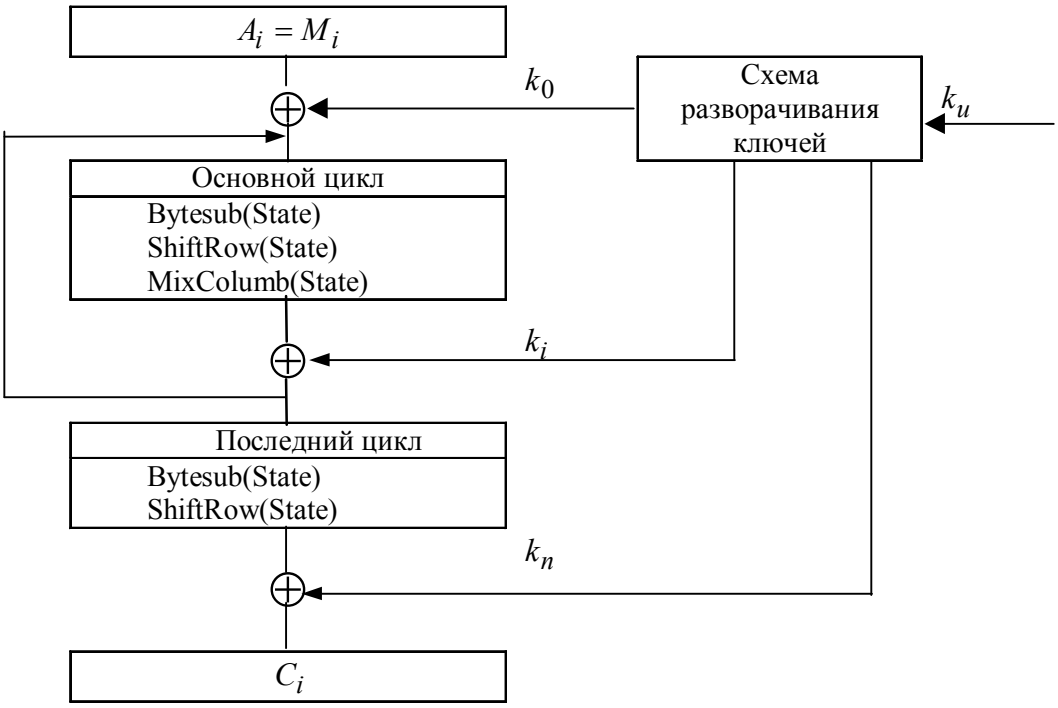

Рисунок 1 – Алгоритм RD блочного зашифрования на $n_ц$

В матричном представлении (5) имеет вид:

$$\begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{vmatrix} \oplus \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix} \qquad (6)$$

Преобразования (4) и (6) 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}:

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 $A_i((1),(2)\text{или}(3))$.

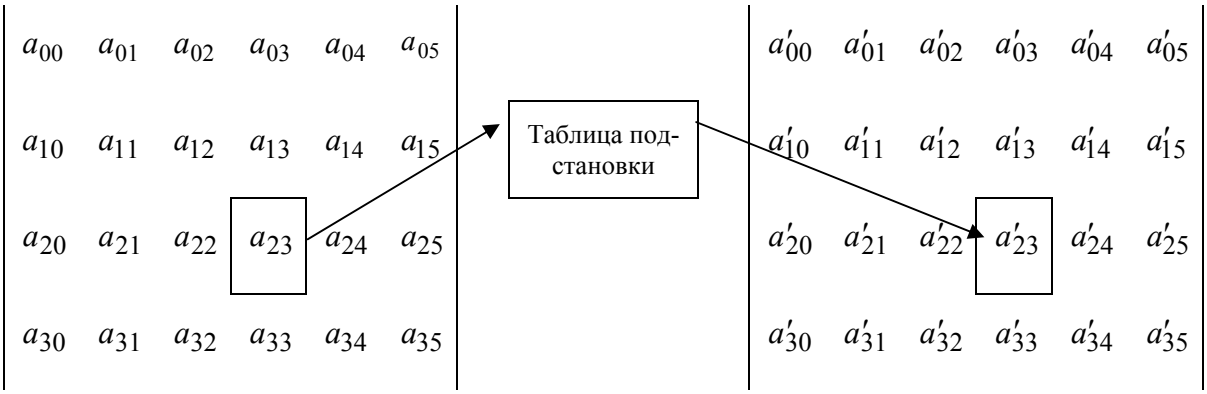

Рисунок 2 – Замена байт

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).

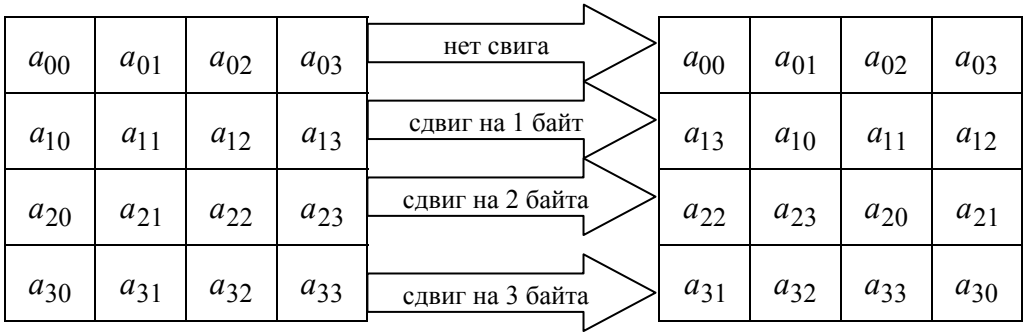

Рисунок 3 – Преобразование «сдвиг строчек»

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

$$a(x) = a_{3j}x^3 + a_{2j}x^2 + a_1x + a_0 \left(\mathrm{mod}\, x^4 + 1\right),\ j = \overline{0, n_c}\ . \tag{7}$$

Коэффициенты $a(x)$ принимают в (7) значения в интервале от нуля (00000000) до 255 (11111111).

Перемешивание производится умножением $a(x)$ на константу

$$C(x) = '03'x^3 + '01'x^2 + '01'x + '02', \tag{8}$$

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

$$a'(x) = C(x) \cdot a(x) \left(\mathrm{mod}\, x^4 + 1\right) \tag{9}$$

В матричном виде преобразование (9) имеет вид:

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

$$\begin{vmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix} \qquad 10)$$

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

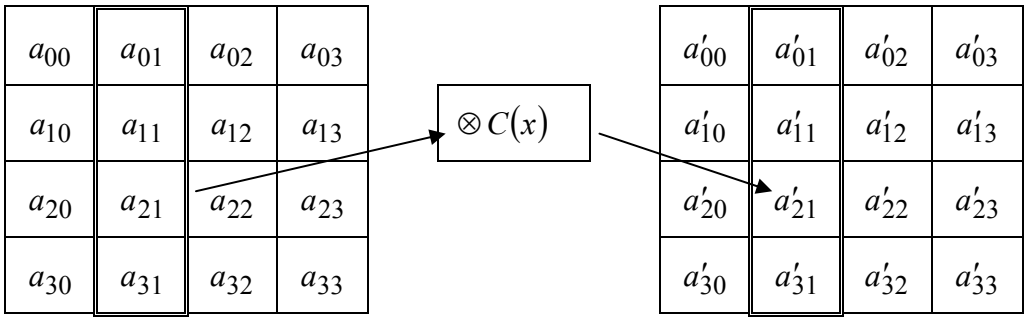

Рисунок 4 – Преобразование «перемешивание в столбцах»

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:

$$A'_{ij} = A_{ij} \oplus K_{\nu j} \qquad 11)$$

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

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

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 $C_{0,0} = a'_{00}, C_{1,0} = a'_{10}, C_{3,0} = a'_{30}, C_{0,1} = a'_{01}, C_{0,2} = a'_{02}, \text{K}$.

IV Выработка цикловых ключей

В алгоритме RD применяется принцип разворачивания цикловых ключей k_u . Сущность его заключается в том, что необходимое число цикловых ключей k_u вырабатывается из исходного ключа K_u . Как исходные, так и цикловые ключи представляются в виде одномерных массивов, например, $k_{00}, k_{10}, k_{20}, k_{30}, k_{01}, k_{11}, \text{K}$ 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 (12).

Для выработки цикловых ключей в алгоритме RD применяются два алгоритма: первый, если длина исходного ключа $n_{k_u} \le 6$, и второй, если $n_k = 8$. Значение n_{k_u} 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 $k_ц$ приведено в табл. 1.

V 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.

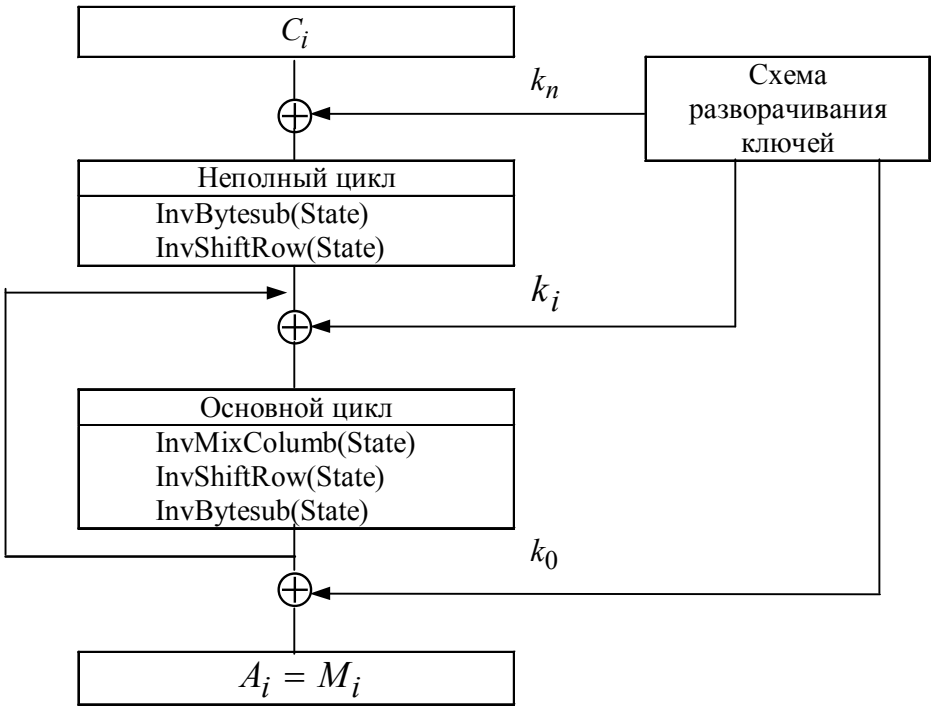

Рисунок 5 – Структурная схема расшифрования C_i блока

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.

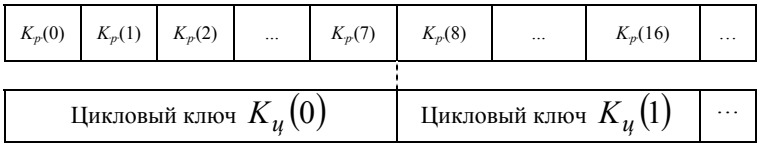

Рисунок 6

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 обрат-

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

ным. 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 таблицами прямых преобразований.

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
InvMixColumb(State); // Обратное перемешивание в столбцах

может быть заменена на последовательность

InvMixColumb(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
}
```

VI Организация табличных преобразований

Матричная структура цикловой функции алгоритма RD позволяет совместить вычисление последовательно расположенных слоев – нелинейного (S-блоки) и линейного (фиксированное умножение на матрицу и XOR с константой). Эти два преобразования могут быть реализованы с помощью одной общей таблицы размерностью $2^8 \times 32$ бита, т.е. входом в таблицу является байт, а выходом 32-битное число. Для реализации такого комбинированного преобразования над 32-битными аргументами потребуется 4 таких таблицы (обозначим их T_0–T_3), тогда цикловое преобразование может быть представлено в виде

$$l_j = T_0(a_{0j}) \oplus T_1(a_{1j-C1}) \oplus T_2(a_{2j-C2}) \oplus T_3(a_{3j-C3}) \oplus K_j,\ j = \overline{0, n_c - 1}. \qquad (13)$$

Учитывая, что столбцы матрицы линейного отображения представляют собой циклический сдвиг вектора (01,01,03,02) на 0,1,2,3 байта, то комбинированное преобразование может быть записано в виде

$$l_j = k_j \oplus T_0\lfloor b_{0j} \rfloor \oplus R_b\left(T_0\lfloor b_{1,j-C1} \rfloor \oplus R_b\left(T_0\lfloor b_{2,j-C2} \rfloor \oplus R_b\left(T_0\lfloor b_{3,j-C3} \rfloor\right)\right)\right). \qquad (14)$$

В (14) 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 (14) по сравнению с (13) 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*.

VII 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 (4) и (5): преобразование (4) ввиду нелинейности обеспечивает защиту от линейных и дифференциально-подобных атак, а (5) – от различных алгебраических вторжений. Оба преобразования, как следует из (13), 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]. Анализ данных позволил с определенной уверенностью считать алгоритм RD 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.

VIII 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. 3 приведены значения сложности развертывания цикловых ключей для различных длин исходных ключей.

Таблица 3

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 результатов конкурса Advanced Encryption Standard (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:

$$K_j^p = X\left(K_j, C_V\right), \tag{15}$$

где C_V означает V-ю константу 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.

$$n_i = \varphi\left(i, c_V\right). \tag{16}$$

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 (15) – соответствующий ему развернутый K_j^p ключ.

2. В каждом исходном K_j ключе изменялся і-й бит, $i = \overline{1, l_p}$, а затем, согласно (15), формировался со-

ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.

ответствующий 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. 7 приведена гистограмма коэффициентов размножения ошибок в развернутом ключе в зависимости от номера позиции измененного бита в исходном ключе для алгоритма RD (длина ключа 128 бит).

Анализ данных рис. 7 показывает, что для алгоритма RIJNDAEL коэффициент размножения «ошибки» в зависимости от позиции изменяемого бита является существенно неравномерным. Природа неравномерности объясняется особенностью алгоритма разворачивания ключа.

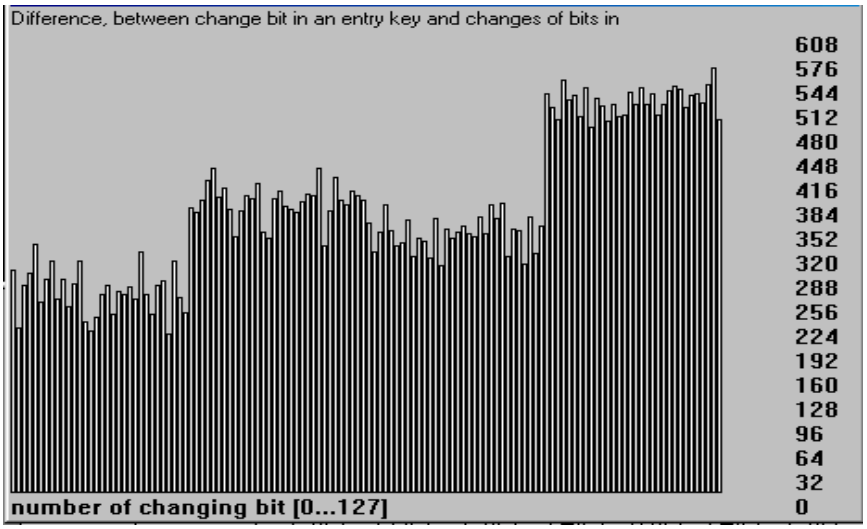

Рисунок 7 – Гистограмма коэффициентов размножения ошибок в развернутом ключе в зависимости от номера позиции измененного бита в исходном ключе для алгоритма RD

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.

Заключение

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.

Литература: 1. *Announcing Development of Federal Information Processing Standard for Advanced Encrytion*

Standard. – Federal Register. Vol. 62, № 1. 1997. Pp. 93-94. 2. Announcing Reguest for Candidate Algorithm Nomination for Advanced Encryption Standard (AES). - Federal Register. Vol. 62, № 177. 1997. Pp. 48051-48058. 3. М. Ф. Бондаренко, И. Д. Горбенко, А. В. Потий. Улучшенный стандарт симметричного шифрования XXI века: концепция создания и свойства кандидатов. Радиотехника. Вып. 114. 2000. 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.

УДК 681.3.06

СТАТИСТИЧЕСКОЕ ТЕСТИРОВАНИЕ ГЕНЕРАТОРОВ СЛУЧАЙНЫХ И ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ С ИСПОЛЬЗОВАНИЕМ НАБОРА СТАТИСТИЧЕСКИХ ТЕСТОВ NIST STS

Александр Потий, Светлана Орлова, Татьяна Гриненко
Харьковский государственный технический университет радиоэлектроники

Аннотация: **Рассматриваются некоторые аспекты выбора и исследования генераторов случайных и псевдослучайных чисел. Выходные данные таких генераторов могут использоваться во многих криптографических приложениях, например при генерации ключевых данных. Генераторы, которые используются в криптографических приложениях, должны удовлетворять более жёстким требованиям, чем остальные генераторы. Выходные последовательности таких генераторов должны быть непредсказуемыми. В этой статье обсуждаются критерии оценки качества и выбора надежных генераторов. Также рассматриваются статистические тесты, которые могут быть использованы как первый шаг при определении, является ли генератор пригодным для конкретных криптографических приложений. Рассматривается методика статистического тестирования генераторов.**

Summary: **This article discusses some aspects of selecting and testing random and pseudorandom number generators. The outputs of such generators may be used in many cryptographic applications, such as the generation of key material. Generators suitable for use in cryptographic applications may need to meet stronger requirements than for other applications. In particular, their outputs must be unpredictable in the absence of knowledge of the inputs. Some criteria for characterising and selecting appropriate generators are discussed in this article. The subject of statistical testing and its relation to cryptanalysis is also discussed, and some recommended statistical tests are provided. These tests may be useful as a first step in determining whether or not a generator is suitable for a particular cryptographic application.**

Ключевые слова: **Генератор случайных чисел, критерий для проверки гипотезы, *P*-значение, криптографические приложения.**

Введение

Тестирование генераторов случайных и псевдослучайных чисел (ГСЧ и ГПСЧ), используемых в криптографических приложениях, является актуальной задачей как в практическом, так и в теоретическом плане. Несмотря на значительные наработки в данной области, разработчики, тем не менее, нуждаются в удобном инструментарии, способном предоставить приемлемую метрику, которая позволит достаточно ясно исследовать степень случайности последовательностей, порождаемых ГСЧ (ГПСЧ), и обеспечить разработчиков достаточным объемом информации для принятия решения относительно “качества” генератора.

На сегодняшний день разработано достаточно большое количество различных типов ГСЧ (ГПСЧ). Однако для демонстрации их статистических свойств использовались различные подходы к статистическому тестированию. Чаще всего набор и методику тестирования предлагал сам разработчик генератора. Таким образом, сложилась ситуация, которая характеризуется тем, что невозможно объективно сравнить различные генераторы с единых позиций. Выходом из этого положения является использование некоторого стандартного набора статистических тестов, объединенных единой методикой расчета необходимых показателей эффективности ГПСЧ и принятия решения о случайности формируемых последовательностей. Наиболее известным набором статистических тестов является набор из пяти тестов, предложенный Кнутом в его классической работе “Искусство программирования для ЭВМ” [1]. Решению этой задачи были посвящены ряд работ отечественных авторов [2, 3, 4]. Однако предложенные решения обладали

 ISBN 966-622-022-9
Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.