АБДРАХМАНОВ Р.Г.[1)], ЖУКОВ А.Е.[2)]

УПРАВЛЯЕМЫЙ ПОДСТАНОВОЧНО-ПЕРЕСТАНОВОЧНЫЙ ШИФР “GRINDER”

[1)] Московский Инженерно-Физический Институт, кафедра «Защита информации», ruslan.abd@gmail.com
[2)] Московский Государственный Технический Университет им. Н.Э.Баумана, кафедра «Информационная безопасность», azhukov@iu8.bmstu.ru

Оглавление

Введение

На конференции РусКрипто-2000 было объявлено, что Криптологическая Ассоциация (РКА) организует конкурс криптографических алгоритмов. В настоящей работе приводится алгоритм GRINDER, представленный на конкурс РусКрипто, а также новый подход к построению шифрующих алгоритмов, лежащий в его основе.

Разработанный алгоритм является алгоритмом шифрования, который работает с блоком данных и ключом произвольной длины. В алгоритме используется перестановка m-битных блоков внутри обрабатываемого массива данных большого размера (сравнимого с объемом системной памяти компьютера), бинарная операция XOR m-битных блоков и их циклический сдвиг, причем каждое последующее преобразование зависит от предыдущего. Шифр сочетает в себе элементы блочных шифров, поточных шифров и шифров перестановки. Алгоритм может оперировать m-битными блоками любого размера. От размера блока зависят скорость и перемешивающие характеристики алгоритма. На данный момент написаны реализации алгоритма, работающие с 16-ти и 32-х битными блоками, что, по-видимому, является оптимальным выбором.

Для уменьшения корреляции между открытым текстом и шифртекстом используется гамма, для инициализации которой используется дополнительный входной массив произвольной длины. Инициализация проводится однократно на начальном этапе работы алгоритма, для инициализации используются однонаправленные функции. Полученный в итоге шифртекст зависит от ключа, массива гаммы и от открытого текста.

Особенность данного алгоритма заключается в том, что в процессе работы алгоритма *ключевая информация постоянно изменяется в процессе ее использования в зависимости от шифруемых данных*: изменение секретного ключа происходит в зависимости от открытого текста, самого ключа и массива гаммы.. Вместе с шифртекстом нужно хранить (или передавать) информацию, необходимую для восстановления ключа расшифрования, роль которого выпол-

няет значение ключа в конце работы алгоритма. В качестве такой информации можно использовать $K' \oplus K$ – сумму секретного ключа и ключа расшифрования. Величина $K' \oplus K$ может также рассматриваться как хэш-функция и использоваться для построения системы электронной цифровой подписи.

1. Подстановочно-перестановочные шифры

Цель данного исследования – получение способа шифрования данных, легко реализуемого программным способом и обеспечивающего рассеивание и перемешивание информации в пределах всего шифруемого массива данных, размер которого ограничивается, вообще говоря, лишь размерами системной памяти используемой ЭВМ.

В основу метода положена идея соединения двух типов преобразований: псевдослучайных преобразований отдельных блоков шифруемого массива данных (информационного массива) и псевдослучайных перестановок преобразованных блоков в пределах всего информационного массива. Поставленная задача достигается тем, что при преобразовании очередного блока данных вначале вычисляется новое значение рабочего ключа, которое зависит как от значения рабочего ключа на предыдущем этапе работы, так и от значения информационного блока, преобразуемого в данный момент. Затем по вычисленному значению рабочего ключа выбирается еще один блок массива данных, номер которого является псевдослучайной величиной. Оба блока подвергаются совместным псевдослучайным преобразованиям, после чего их образы меняются местами в информационного массива данных. Под *псевдослучайностью* понимается то, что без знания текущего значения рабочего ключа невозможно определить совместно с каким блоком будет преобразовываться данный информационный блок и каким преобразованиям они будут подвержены.

1.1. Формальное описание подстановочно-перестановочных шифров

Предлагаемый способ криптографического преобразования может быть описан следующим формальным образом.

Будем обозначать через $f(x|y_1,...,y_n)$ функцию (отображение) множества $X \times Y_1 \times \ldots \times Y_n$ в множество Z, обладающую тем свойством, что при любых фиксированных значениях элементов $y_1,\ldots, y_n$, где $y_i \in Y_i$, соответствующее отображение множества X в множество Z обратимо. Если зафиксировать значения элементов $y_1,\ldots, y_n$, $(y_i \in Y_i)$, то соответствующее обратное отображение будем обозначать через $f^{-1}(z|y_1,...,y_n)$, т.е. если $f(x|y_1,...,y_n)=z$, то $f^{-1}(z|y_1,...,y_n)=x$.

Пусть M – двоичный массив данных, подлежащий криптографическому преобразованию; пусть объем этого массива – $n*m$ бит, где m – некоторая фиксированная величина. Размер массива данных может быть произвольным и ограничивается лишь размером системной памяти ЭВМ.

Вначале весь массив данных M размера $n*m$ бит разбивается на n блоков $(M_0, M_1, \ldots, M_{n-1})$ по m бит в каждом блоке.

Алгоритм шифрования состоит из $T \geq 1$ циклов шифрования: сначала выполняется 1-й цикл шифрования, затем 2-й и т.д. до T-го цикла. Каждый цикл шифрования состоит из n итераций: при выполнении t-го цикла шифрования в начале выполняется 0-я итерация, затем 1-я итерация и т.д. до $(n-1)$-й итерации. После выполнения $(n-1)$-й итерации t-го цикла шифрования алгоритм начинает выполнять 0-ю итерацию $(t+1)$-го цикла. После выполнения $(n-1)$-й итерации T-го цикла шифрования алгоритм заканчивает работу.

Пусть K – секретный ключ. В начале работы алгоритма ключевая информация помещается в рабочий массив, содержимое которого в процессе работы алгоритма постоянно изменяется. Будем обозначать заполнение этого массива при i-ой итерации t-го цикла работы алгоритма через $K_i^{(t)}$ и называть его значением ***рабочего ключа*** при i-ой итерации t-го цикла. Преобразование, которому алгоритм шифрования подвергает массив данных M в результате i-ой итерации t-го цикла задается (определяется) следующим образом:

1. Вычисляется значение рабочего ключа для i-ой итерации t-го цикла шифрования:

$K_i^{(t)} = f(K_{i-1}^{(t)} \mid M_i)$ для всех $i \geq 1$,

При этом: для $i = 0,\ t \geq 2$, полагаем $K_{-1}^{(t)} = K_{n-1}^{(t-1)}$;

для $i = 0,\ t = 1$, полагаем $K_{-1}^{(1)} = K$

2. Определяется номер блока, который будет преобразовываться совместно с блоком M_i:

$$j = \varphi(K_i^{(t)})$$

3. Вычисляются новые значения блоков M_i и M_j:

$$A = F_1(M_j \mid M_i, K_i^{(t)})$$

$$B = F_2(M_i \mid K_i^{(t)})$$

$$M_i = A$$

$$M_j = B$$

4. Если $i < n-1$, то осуществляется переход к итерации $i+1$ того же цикла t;

Если $i = n\text{-}1$ и $t < T$, то осуществляется переход к итерации 0 цикла $t+1$;

Если $i = n\text{-}1$, $t = T$, то работа алгоритма заканчивается.

Значения блоков M_0, M_1, …, $M_{n\text{-}1}$ после завершения последней итерации последнего цикла шифрования образуют массив шифртекста, который передается (хранится) вместе со значением хэш-функции

$K' = k(K_{n\text{-}1}{}^{(T)} \| K)$, которая одновременно служит для получения ключа расшифрования $K_{n\text{-}1}{}^{(t\text{-}1)}$.

Алгоритм расшифрования преобразует массив данных M, который также разбит на блоки $M_0, M_1, \ldots, M_{n\text{-}1}$ по m бит в каждом и начинается с определения значения рабочего ключа $K_{n\text{-}1}{}^{(T)}$:

$$K_{n\text{-}1}{}^{(T)} = k^{-1}(K' \| K).$$

Затем проделывается T циклов расшифрования, начиная с T-го цикла расшифрования и заканчивая 1-м циклом расшифрования. Каждый цикл расшифрования состоит из n итераций, начиная с выполнения $(n\text{-}1)$-й итерации и заканчивая выполнением 0-й итерации. После 0-й итерации t-го цикла расшифрования выполняется $(n\text{-}1)$-я итерация $(t\text{-}1)$-го цикла расшифрования. После выполнения 0-й итерации 1-го цикла расшифрования алгоритм расшифрования заканчивает работу.

Преобразование, которому алгоритм расшифрования подвергает массив данных M в результате i-ой итерации t-го цикла расшифрования задается следующим образом:

1. Определяется номер блока, который будет преобразовываться совместно с блоком M_i:

$$j = \varphi(K_i^{(t)})$$

2. Вычисляются новые значения блоков M_i и M_j:

$$A = F_2^{-1}(M_j \mid K_i^{(t)})$$

$$B = F_1^{-1}(M_i \mid A, K_i^{(t)})$$

$$M_i = A$$

$$M_j = B$$

3. Вычисляется значение рабочего ключа для $(i\text{-}1)$-ой итерации t-го цикла шифрования:

$K_{i-1}^{(t)} = f^{-1}(K_i^{(t)} \mid M_i)$ для всех $i \geq 1$;

При этом для $i = 0,\ t < T,$ полагаем $K_{n-1}^{(t)} = K_{-1}^{(t+1)}$;

для $i = n\text{-}1,\ t = T,$ полагаем $K_{n-1}^{(T)} = k^{-1}(K' \mid K)$

5. Если $i < 0$, то осуществляется переход к итерации $i\text{-}1$ того же цикла t;

Если $i = 0$ и $t > 1$, то осуществляется переход к $(n\text{-}1)$-й итерации цикла $t\text{-}1$;

Если $i = 0, t = 1$, то работа алгоритма расшифрования заканчивается.

Значения блоков $M_0, M_1, \ldots, M_{n-1}$ после завершения 0-й итерации 1-го цикла расшифрования образуют массив исходного открытого текста.

Отметим еще раз следующие особенности предлагаемого алгоритма:

- рабочий ключ для каждой итерации алгоритма вычисляется по значению преобразуемого блока и значению рабочего ключа на предыдущей итерации;

- информационный блок, преобразуемый во время очередной итерации, преобразуется совместно с другим блоком, выбор которого из общего массива данных зависит от значения рабочего ключа на данной итерации;
- два преобразованных на данной итерации блока переставляются в общем массиве данных.

Благодаря такому решению обеспечивается рассеивание и перемешивание информации в пределах всего преобразуемого массива данных, что делает невозможным применение статистических методов криптоанализа, в том числе методов разностного криптоанализа и линейного криптоанализа.

1.2. Выбор функций

При конкретной реализации предлагаемого способа криптографического преобразования информации, необходимо задать легко реализуемые обратимые функции

$$f(K_{i-1}^{(t)} \mid M_i), \qquad F_1(M_j \mid M_i, K_i^{(t)}),$$
$$F_2(M_i \mid K_i^{(t)}), \qquad k(K_{N-1}^{(T)} \mid K),$$

а также легко реализуемую функцию выбора номера блока $\varphi(K_i^{(t)})$.

Одним из вариантов выбора указанных выше функций, является следующий.

1.2.1. Входные данные

Пусть M – двоичный массив данных, подлежащий криптографическому преобразованию, в дальнейшем этот массив будет называться ***информационным массивом***; пусть объем этого массива – $n*m$ бит, где m – некоторая фиксированная величина. Размер информационного массива может быть произвольным и ограничивается лишь размером системной памяти ЭВМ. Весь массив данных M размера $n*m$ бит разбивается на n блоков $(M_0, M_1, \ldots, M_{n-1})$ по m

бит в каждом блоке, в дальнейшем эти блоки будут называться ***информационными блоками***.

Пусть K – двоичный массив данных, образующих секретный ключ, в дальнейшем этот массив будет называться ***ключевым массивом***; пусть объем этого массива – $s*m$ бит, где m – та же величина, что и выше. Весь массив K разбивается на s блоков $(K_0, K_1, \ldots, K_{s-1})$ по m бит в каждом блоке, в дальнейшем эти блоки будут называться ***подключами***. В процессе работы алгоритма заполнение массива K меняется при переходе к каждой следующей итерации; его заполнение при выполнении i-ой итерации t-го цикла работы алгоритма является значением рабочего ключа $K_i^{(t)}$. ***Рабочим подключем*** называется блок массива K, используемый в преобразованиях на i-ой итерации t-го цикла работы алгоритма.

Для изменения информационных блоков и подключей в процессе работы алгоритма используются дополнительные блоки, которые выбираются псевдослучайным образом из проинициализированного в начале работы алгоритма дополнительного массива G, называемого ***массивом гаммы***. Пусть объем этого массива – $p*m$ бит, где m – та же величина, что и выше. Весь массив G разбивается на p блоков $(G_0, G_1, \ldots, G_{p-1})$ по m бит в каждом блоке, в дальнейшем эти блоки будут называться ***блоками гаммы***.

1.2.2. Инициализация гаммы

На начальном этапе происходит инициализация массива гаммы. Данная процедура необходима для того, чтобы обеспечить выработку качественной гаммы в процессе шифрования. Изначально массив гаммы заполняется любыми данными, например, текстом или псевдослучайной последовательностью. Инициали-

зация представляет собой многократное однонаправленное изменение начального заполнения массива гаммы (например операциями циклического сдвига, векторного сложения, поразрядного XOR и т.д.) с использованием секретного ключа. Инициализация снимает жесткие ограничения на длину и начальное заполнение массива гаммы, что позволяет использовать в качестве начального заполнения как текстовые, так и бинарные фрагменты.

Инициализация гаммы: $G' = I_K(G)$ – однократная инициализация дополнительного массива в зависимости от ключа шифрования.

1.2.3. Зашифрование

Зашифрование информационного массива производится последовательным считыванием информационных блоков и подключей (из соответствующих массивов) с их последующим преобразованием. При преобразовании очередного блока данных вначале вычисляется новое значение рабочего ключа, которое зависит как от значения рабочего ключа на предыдущем этапе работы, так и от значения информационного блока, преобразуемого в данный момент. Затем по вычисленному значению рабочего ключа выбирается еще один блок массива данных, номер которого является псевдослучайной величиной. Оба блока подвергаются совместным псевдослучайным преобразованиям, после чего их образы меняются местами в информационном массиве данных.

Будем пользоваться следующими обозначениями:

- X' – преобразованный массив X;
- $(X)RRot(Y)$ – циклический сдвиг X вправо на Y позиций
- $(X)LRot(Y)$ – циклический сдвиг X влево на Y позиций;
- Операция + это либо $\oplus$ (XOR), либо $\boxplus$ (сложение по модулю 2^m, где m – размер блоков в битах).
- $\nu_1,\ \nu_2,\ \pi_1,\ \pi_2 : (Z_2)^m \to Z_m$
- $\lambda_1,\ \mu_1 : (Z_2)^m \to Z_s$

- $\lambda_2,\ \mu_2,\ \psi_1,\ \psi_2 : (Z_2)^m \to Z_p$
- $\varphi : (Z_2)^m \to Z_n$

Преобразование массива *K*: Вычисляется значение рабочего ключа для очередной итерации очередного цикла шифрования:

$K_i^{'} = K_i + M_y$ – изменение текущего подключа в зависимости от текущего блока текста.

$K_{i+1}^{'} = K_{i+1} + K_{\lambda_1(K_i^{'})} + G_{\lambda_2(K_i^{'})}$ – изменение следующего по порядку подключа в зависимости от текущего подключа.

$K_i^{''} = K_i^{'} + K_{\mu_1(K_{i+1}^{'})} + G_{\mu_2(K_{i+1}^{'})}$ – изменение текущего подключа в зависимости от следующего по порядку использования подключа.

$K_{i+1}^{''} = K_{i+1}^{'} RRot(\nu_1(K_i^{''}))$ – циклический сдвиг следующего подключа вправо на количество разрядов, определяемое текущим подключом.

$K_i^{''} = K_i^{''} RRot(\nu_2(K_{i+1}^{''}))$ – циклический сдвиг текущего подключа вправо на количество разрядов, определяемое следующим подключом.

Преобразование массива M:

Сложение текущего и псевдослучайно выбранного (в зависимости от текущего подключа) блока текста с блоком, псевдослучайно выбранным (в зависимости от текущего подключа) из дополнительного массива, циклический сдвиг вправо на количество разрядов, определяемое текущим подключом, и их взаимная перестановка:

$$M_y = (M_{\varphi(K_i^{'''})} + G_{\psi_1(K_i^{'''})}) RRot(\pi_1(K_i^{'''}))$$

$$M_{\varphi(K_i^{'''})} = (M_y + G_{\psi_2(K_i^{'''})}) RRot(\pi_2(K_i^{'''}))$$

Заполнение ключевого массива в конце работы алгоритма является ключем расшифрования K' . Значение $K' \oplus K$ является значением хэш-функции для нашего текста.

На основе предлагаемой схемы может быть построено множество алгоритмов, параметры которых будут зависеть от конкретных требований. Для достижения высокой скорости работы можно, например, упростить используемые функции, зафиксировать размер ключа, дополнительного массива и обрабатываемого текста, а для повышения криптостойкости наоборот – усложнить функции и увеличить число циклов шифрования.

2. Алгоритм шифрования “GRINDER”.

2.1. Используемые обозначения

2.1.1. Входные данные

M *– информационный массив – часть открытого текста или шифртекста, которая одновременно обрабатывается алгоритмом шифрования;*

K *– ключевой массив;*

G *– массив гаммы;*

n *– размер информационного массива (в словах по 32 бита);*

s *– размер ключевого массива (в словах по 32 бита);*

p *– размер массива гаммы (в словах по 32 бита);*

2.1.2. Операции и преобразования

Если X *– двоичный вектор, то* $|X|$ *– число, имеющее* X *своей двоичной записью;*

$(a)_b = a \bmod b;$ $\quad (X)_b = |X| \bmod b;$

$X \oplus Y = X$ **XOR** $Y;$

$X+Y$ *– вектор, соответствующий их арифметической сумме, т.е.* $|X|+|Y|;$

$(X)RRot(Y)$ *– циклический сдвиг* X *вправо на* $|Y|$ *позиций;*

$(X)LRot(Y)$ *– циклический сдвиг* X *влево на* $|Y|$ *позиций;*

A' *– преобразованный массив* $A;$

X' *– преобразованный блок* $X;$

2.2. Инициализация гаммы

На начальном этапе происходит инициализация массива гаммы. Данная операция необходима для того, чтобы обеспечить получение исходного массива для выработки качественной гаммы в процессе шифрования. Инициализация представляет собой многократное однонаправленное изменение начального заполнения массива гаммы с помощью секретного ключа, используя операции циклического сдвига, векторного и арифметического сложения. Процедура инициализации снимает жесткие ограничения на начальное заполнение и длину массива гаммы, что позволяет использовать в качестве таковых как текстовые, так и бинарные фрагменты. Желательно, чтобы этот фрагмент обладал небольшой избыточностью, поэтому автором настоятельно рекомендуется использовать для выработки таких данных современные технологии и алгоритмы для получения случайной или псевдослучайной последовательности.

Инициализация гаммы

От u = 1 до Число_Циклов_Инициализации_Гаммы делать

От y =1 до p делать

$$G'_y = (G_y \, RRot(G_y \oplus K_{(y+G_y)_s})_{32}) \oplus G_{(y+G_y \oplus K(y+G_y)_s))_p}$$

конец y

конец u

2.3. Зашифрование

Зашифрование файла производится последовательным считыванием из файла информационных массивов, образованных блоками данных в заданном количестве (n блоков по 32 бита в каждом блоке), преобразованием их и записью преобразованных массивов в выходной файл. Заполнение ключевого массива в конце работы алгоритма является ключем расшифрования K'. Значение $K' \oplus K$ является значением хэш-функции от открытого текста.

Зашифрование

Инициализация_гаммы(K, G)

i - номер слова в ключевом массиве, i = 0 ÷ s-1

y - номер слова в информационном массиве, y = 0 ÷ n-1

i=0

от и = 0 до Число_Главных_Циклов делать

от y=0 до n-1 делать

1. $K'_i = K_i \oplus M_y$ – изменение текущего подключа в зависимости от текущего информационного блока

2. $K'_{(i+K'_i)_S} = K'_i \oplus (K_{(i+K'_i)_S} RRot((i + K'_i)_S)_{32})$

– изменение второго фрагмента ключа в зависимости от текущего фрагмента ключа.

3. $M'_y = M_{(y+K'_i)_n}$

$$M'_{(y+K'_i)_n} = M_{(y+K'_i)_n} \oplus (M_y RRot((y + K'_i)_n)_{32})$$

– перестановка пары блоков данных, изменение одного из блоков путем

сложения со вторым операцией XOR и циклическим сдвигом в зависимости от его номера и текущего фрагмента ключа.

4. $K''_{(i+K'_i)_s} = (K'_{(i+K'_i)_s} \; RRot((K'_i)_p)_{32}) \oplus G(K'_i)_p$

– изменение второго фрагмента ключа путем циклического сдвига в зависимости от текущего фрагмента ключа и последующим сложением с блоком гаммы операцией XOR.

5. $M''_y = (M'_y \, RRot((K'_i)_p)_{32}) \oplus G(K'_i)_p \oplus$
$(G((K'_i)_p + G(K'_i)_p)_p \; RRot(((K'_i)_p + G_{(K'_i)_p})_p)_{32})$

– изменение текущего блока открытого текста циклическим сдвигом в зависимости от текущего фрагмента ключа и сложением с двумя блоками гаммы операцией XOR, один из которых изменен циклическим сдвигом в зависимости от текущего фрагмента ключа и второго блока гаммы.

i=i+1

если i > s-1 то i = 0

конец у

конец и

2.4. Расшифрование

Расшифрование производится чтением информационных массивов с конца шифрованного файла в обратном порядке, расшифрованием их и записью в выходной файл.

Расшифрование

Инициализация_гаммы(K, G)

$i = (n \cdot (Число\_Циклов))_S - 1$ – восстановление номера блока ключа, который использовался последним в операции зашифрования

Если i < 0 то i = s –1

От u = 0 до Число_Циклов делать

y = n -1

Пока y ≥ 0 делать

1. $M'y = (My \oplus G_{(K_i)_p} \oplus$

$\oplus (G((K_i)_p + G(K_i)_p)_p \; LRot(((K_i)_p + G_{(K_i)_p})_p)_{32}))RRot((K_i)_p)_{32}$

– изменение текущего блока открытого текста сложением с двумя блоками гаммы операцией XOR, один из которых изменен циклическим сдвигом в зависимости от текущего блока ключа и второго блока гаммы и циклическим сдвигом в зависимости от текущего блока ключа.

2. $K'_{(i+Ki)_S} = K_{(i+Ki)_S} \oplus G_{(Ki)_p}$

$K''_{(i+Ki)_S} = K'_{(i+Ki)_S} \; LRot \; ((K_i)_p)_{32}$

– изменение второго фрагмента ключа сложением с блоком гаммы операцией XOR и последующим циклическим сдвигом в зависимости от текущего фрагмента ключа.

3. $M''y = (M'_{(y+Ki)_N} \oplus M'y) LRot((y + K_i)_N)_{32}$

$M'_{(y+Ki)_N} = M'y$

– перестановка пары блоков данных, изменение одного из блоков путем сложения со вторым операцией XOR и циклическим сдвигом в зависимости от текущего фрагмента ключа.

4. $$K''_{(i+K_i)_S} = (K_i \oplus K'_{(i+K_i)_S}) LRot((i + K_i)_S)_{32})$$

– изменение второго фрагмента ключа в зависимости от текущего фрагмента ключа.

5. $$K'_i = K_i \oplus M''_y$$

– изменение ключа в зависимости от открытого текста

i = i – 1

если i < 0 то i = s - 1

y = y –1

конец y

конец u

3. Реализация и тестирование

3.1. Реализация и скорость

Алгоритм был реализован на языке C в среде разработки приложений Visual Studio 6.0 для работы под управлением операционной системы Windows 9x или Windows NT.

Реализованный вариант алгоритма не был оптимизирован, но даже в этом случае скорость алгоритма составила 10Мбайт/сек (80Мбит/сек) для процессора Intel Celeron с тактовой частотой 845MHz. Скорость определялась путем шифрования 1000 раз одного и того же файла объемом в 1Мбайт. Благодаря тому, что в алгоритме используются простые (быстрые) операции, возможно значительное увеличение скорости шифрования, путем упрощения используемых функций, при незначительной потере криптостойкости.

Для определения характеристик алгоритма был проведен ряд тестов.

3.2. Равномерности распределения байтов шифртекста

Проверка равномерности распределения байтов шифртекста производилась с помощью критерия согласия хи-квадрат К.Пирсона. Проведенные тесты показали, что при шифровании текстового и бинарного файлов размера 1 Мб каждый равномерность распределения байтов шифртекста достигается если:

- длина ключа – не менее 5 байтов,
- длина массива гаммы – не менее 45 байтов,
- число главных циклов – не менее 2-х,

причем увеличение параметров сверх указанных не ведет к заметному увеличению равномерности распределения.

3.3. Эффект размазывания

Другой важной характеристикой является эффект размазывания. Проверялось изменение шифртекта в зависимости от изменения 1 бита открытого текста или 1 бита ключа в любой позиции. В идеале должна измениться примерно половина битов шифр-текста. Тест показал ≈50% изменение. Тестировались данные различных типов. Позиция измененного бита значения не имеет.

3.4. Имитостойкость

Имитостойкость проверялась путем определения влияния изменения одного бита шифртекста на расшифрованный текст. Разница побитового сравнения файлов после расшифрования, в одном из которых был предварительно изменен 1 бит, составила 50%. Позиция измененного бита значения не имеет.

3.5. Анализ накладываемой гаммы

Так как на блоки текста накладываются блоки, псевдослучайным образом выбранные из массива гаммы (т.е. фактически на текст накладывается гамма), необходимо проанализировать накладываемую последовательность. Тестировалась последовательность размером 3 Кбайта, т.е. 24000 бит, что удовлетворяет требованиям тестов. Использовались пять основных тестов для проверки качества псевдослучайных последовательностей:

- Monobit test
- Two-bit test
- Poker test
- Runs test
- Autocorrelation test

Исследуемая последовательность удовлетворительно прошла все тесты. Это говорит о том, что исследуемая последовательность близка по своим свойствам к случайной и вполне удовлетворяет требованиям данного алгоритма.

Для определения линейной сложности исследуемой последовательности использовался алгоритм Берлекемпа-Месси. Тест показал, что математическое ожидание сложности исследуемой последовательности длины n равно $n/2$, как и у случайной последовательности.

Выводы

1. Впервые предложен метод шифрования, основанный на одновременном преобразовании двух блоков текста, причем вместе с текущим по очереди блоком преобразуется блок, номер которого выбран псевдослучайным образом (в зависимости от текущего состояния рабочего ключа). При этом преобразованные блоки переставляются внутри массива большего размера в зависимости от секретного ключа и шифруемой информации.
2. Рабочий ключ изменяется после обработки каждого блока текста, причем его изменение зависит от самого текста и предыдущего значения рабочего ключа. Раскрытие подключа на одном из тактов работы не определяет даже части ключевого массива, так как в дальнейшем он пересчитывается.
3. Изменение одного бита открытого текста ведет к изменению рабочего ключа и, следовательно, к изменению всех последующих преобразований.
4. Конечное значение рабочего ключа является ключом расшифрования. Ключ расшифрования зависит от ключа шифрования и открытого текста. Для каждого открытого текста он свой. Ключ расшифрования сло-

женный с ключом шифрования (или преобразованный иным способом) играет роль хэш-функции.

5. Предложенный метод шифрования с изменением рабочего ключа в зависимости от шифруемых данных, перестановкой блоков данных и последующим наложением гаммы из-за своих особенностей, отличных от методов большинства стандартных алгоритмов, значительно усложняет задачу криптоанализа. Поскольку алгоритм осуществляет перемешивание информации в пределах большого массива данных, становятся практически неприменимыми статистические подходы к анализу шифрующих алгоритмов, в частности дифференциальный и линейный криптоанализ.
6. Поскольку для опробования ключа потребуется расшифровать (или зашифровать) весь массив данных, возможность машинного перебора секретных ключей существенно осложнится, что служит защитой от атаки «грубой силой».
7. Частичное совпадение двух открытых текстов не дает совпадения соответствующих шифр-текстов. Изменение открытого текста даже в одном бите ведет к лавинному изменению шифр-текста (≈50%). Причем лавинный эффект происходит не на уровне одного блока, а на уровне всего текста.
8. Тестированием алгоритма шифрования установлено его соответствие криптографическим требованиям для применения в различных информационных системах. Однако новизна предлагаемого метода шифрования требует в будущем его всестороннего анализа (и, при необходимости, дальнейшего совершенствования).
9. Таким образом:

- Данный алгоритм может использоваться в качестве симметричного алгоритма шифрования. Благодаря своим качествам, а именно: универсальности, скорости, простоте реализации, он может использоваться для

построения широкого спектра защищенных информационных систем с различными требованиями к криптографическим средствам защиты.

- Этот алгоритм можно также использовать для организации симметричной криптосистемы с сеансовыми ключами, что повышает защищенность информационной системы. В качестве ключа зашифрования сеанса может использовать ключ расшифрования предыдущего сеанса. Для первой инициализации алгоритма участникам диалога необходимо по секретному каналу передать начальный ключ и, желательно так же по секретному каналу, синхропосылку для инициализации гаммы.
- Описанный алгоритм может использоваться для выработки хэш-функции. На его основе может быть создана система электронной цифровой подписи.