Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра радиотехники

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Методические указания по выполнению лабораторных работ для студентов специальности
5В100200 – Системы информационной безопасности

Алматы 2014

СОСТАВИТЕЛИ: Куликов А.А., Суйеубаев О.Б. Теоретические основы защиты информации. Методические указания по выполнению лабораторных работ для студентов специальности 5В100200 – Системы информационной безопасности. – Алматы: АИЭС, 2014. – 32 с.

Методические указания содержат материалы по подготовке к проведению лабораторных работ, в них приведены подробные описания алгоритмов шифрования и дешифрования, необходимых для выполнения работ, задание и методика выполнения, перечень рекомендуемой литературы и контрольные вопросы.

Табл. 7 ; Ил. 12; Библиогр. – 4 назв.

Рецензент: доцент Аманбаев А.А.

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2014 г.

© НАО «Алматинский университет энергетики и связи», 2014 г.

1 Лабораторная работа №1. Изучение информационно-статистических характеристик текста и частотного метода криптоанализа симметричных криптосистем

Цель работы: определить информационно-статистических характеристик текста и закрепить теоретических знания по методам криптоанализа симметричных криптосистем, а так же частотного метода криптоанализа на примере криптосистемы Цезаря.

1.1 Порядок выполнения работы

1.1.1 Используя литературу [1, 2, 3, 4] и материалы лекций углубить свои знания по криптосистеме Цезаря и частотному методу криптоанализа простейших шифров.

Подготовить русский алфавит со значениями вероятностей встречаемости букв.

1.1.2 Написать программу для определения вероятности встречающегося символа $p_1(j)$ для русского алфавита:

$$p_1(j) = \sum_{i=0}^{n} (x_i \subset j) \Big/ n .$$

1.1.3 Для текста длинной 1000 знаков используя написанную программу рассчитать $p_1(j)$ и $\log(p_1(j))$.

1.1.4 Сравнить полученные значения $p_1(j)$ и $\log(p_1(j))$ с приведенными в литературе значениями.

1.1.5 Написать короткое сообщение длиной 20-25 знаков и закодировать его, используя криптосистему Цезаря.

1.1.6 Для закодированного сообщения определить частотные характеристики криптограммы, для чего рассчитать значение частоты встречаемости символов $j \subset A_m$ в криптограмме.

1.1.7 Определить вероятностные характеристики алфавита в закодированном сообщении, для чего определить частоту встречаемости буквы $\nu_j(Y)$ и полученные значения свести в таблицу 1.1.

Т а б л и ц а 1.1

Буква	А	Б	…	Ю	Я
$j \subset A_m$					
$\log(p_1(j))$					
$\nu_j(Y)$					

1.1.8 В соответствии с выражением (1) определить значение логарифма функции правдоподобия l(K) и построить соответствующую графическую зависимость.

1.1.9 Определить в соответствии с выражением (1) оценку ключа k*.

1.1.10 Дешифровать заданную криптограмму, используя оценку ключа k*. При получении осмысленного текста подготовить отчет и представить его преподавателю.

1.2 Основные теоретические сведения

Криптосистема Цезаря определяется выражением:

$y_i=(x_i+k)\bmod m,\ i=\overline{1,n},$

где y_i - буква криптограммы;

x_i - буква открытого сообщения;

k - ключ шифра;

n - длина криптограммы (открытого текста);

m - мощность алфавита.

Выражение для дешифрования имеет вид:

$x_i=(y_i-k)\bmod m.$

Метод частотного криптоанализа базируется на реализации методов теории статистических решений, а именно, на методе максимального правдоподобия [1]. В соответствии с этим методом оценкой ключа шифра k* является такое его значение, которое доставляет максимальное значение логарифму функции правдоподобия l(k). Для криптосистемы Цезаря оценка формируется в соответствии с выражением:

$$k^* = \arg\max_k l(k), \quad l(k) = \sum_{j=0}^{m-1} \nu_{(j+k)\bmod m} \cdot \log\left(p_1(j)\right) ,$$

где $p_1(j)$ - оценка вероятности встречаемости j-й буквы алфавита мощности m в открытых текстах;

ν - частота встречаемости j-й буквы в криптограмме.

Это выражение справедливо, если источник открытых сообщений представляет собой стационарный источник дискретных сообщений без памяти.

1.3 Содержание отчета

Отчет должен включать в себя следующие пункты:

1) Задание на выполнение лабораторной работы (исходную криптограмму).

2) Основные расчетные соотношения.

3) Результаты расчетов, сведенные в таблицу 1.1.

4) Графическую зависимость l(K) и значение оценки ключа k*.

5) Полученный дешифрованием открытый текст.

1.4 Контрольные вопросы

1. Основные понятия криптографии и криптоанализа.

2. Понятие симметричной криптосистемы.
3. Шифры перестановки.
4. Шифры замены.
5. Основные характеристики открытых сообщений.
6. Модели источников открытых сообщений.
7. Частотный метод криптоанализа.
8. Функция правдоподобия.

2 Лабораторная работа № 2. Криптографические хэш-функции

Цель работы: изучить алгоритм SHA–256 вычисления дайджестов сообщений и написать программу, реализующую заданный алгоритм хэширования.

2.1 Порядок выполнения работы

2.1.1 Реализовать приложение с графическим интерфейсом, позволяющее выполнять следующие действия:

а) вычислять значение хэш-функции при этом:

– текст сообщения должен считываться из файла;

– полученное значение хэш-функции должно представляться в шестнадцатеричном виде и сохраняться в файл;

– при работе программы должна быть возможность просмотра и изменения считанного из файла сообщения и вычисленного значения хэш-функции;

б) исследовать лавинный эффект на сообщении, состоящем из одного блока:

– для бита, который будет изменяться, приложение должно позволять задавать его позицию (номер) в сообщении;

– приложение должно уметь после каждого раунда (итерации цикла) вычисления хэш-функции подсчитывать число бит, изменившихся в хэше при изменении одного бита в тексте сообщения;

– приложение может строить графики зависимости числа бит, изменившихся в хэше, от раунда вычисления хэш-функции, либо графики можно строить в стороннем программном обеспечении, но тогда приложение должно сохранять в файл необходимую для построения графиков информацию.

2.1.2 С помощью реализованного приложения выполнить следующие задания:

а) протестировать правильность работы разработанного приложения;

б) исследовать лавинный эффект при изменении одного бита в сообщении: для различных позиций изменяемого бита в сообщении построить графики зависимостей числа бит, изменившихся в хэше, от раунда вычисления хэш-функции (всего в отчёте должно быть 2–3 графика);

в) сделать выводы о проделанной работе.

2.2 Основные теоретические сведения

Хэширование (иногда *хеширование*, англ. *hashing*) – преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Такие преобразования также называются *хэш-функциями* или *функциями свёртки*, а их результаты называют *хэшем*, *хэш-кодом* или *дайджестом сообщения* (англ. *message digest*) [5].

Хэширование применяется для сравнения данных: если у двух массивов хэш-коды разные, массивы гарантированно различаются; если одинаковые – массивы, скорее всего, одинаковы. В общем случае, однозначного соответствия между исходными данными и хэш-кодом нет в силу того, что количество значений хэш-функций меньше, чем вариантов входного массива. Также существует множество массивов, дающих одинаковые хэш-коды – так называемые *коллизии*. Вероятность возникновения коллизий играет немаловажную роль в оценке качества хэш-функций.

Существует множество алгоритмов хэширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкость и т.п.). Выбор той или иной хэш-функции определяется спецификой решаемой задачи. Простейшими примерами хэш-функций могут служить контрольные суммы.

Контрольные суммы – это несложные, крайне быстрые и легко реализуемые аппаратные алгоритмы, используемые для защиты от непреднамеренных искажений, в том числе от ошибок аппаратуры. По скорости вычисления они в десятки и сотни раз быстрее, чем криптографические хэш-функции, и значительно проще в аппаратной реализации.

Платой за столь высокую скорость является отсутствие криптостойкости – очень просто подобрать сообщение под заранее известную сумму. Разрядность контрольных сумм (обычно 32 бита) ниже, чем криптографических хэшей (типичные значения – 128, 160, 256 бит и более), что означает возможность возникновения непреднамеренных коллизий.

Простейшим случаем такого алгоритма является деление сообщения на 32- или 16-битные слова и их суммирование, что применяется, например, в TCP/IP. Как правило, к такому алгоритму предъявляются требования отслеживания типичных аппаратных ошибок, таких, как несколько подряд идущих ошибочных бит до заданной длины.

Среди множества существующих хэш-функций выделяют криптографически стойкие, применяемые в криптографии. Для того чтобы хэш-функция считалась *криптографически стойкой*, она должна удовлетворять трём основным требованиям, на которых основано

большинство применений хэш-функций в криптографии:

– *необратимость*: для заданного значения хэш-функции должно быть вычислительно неосуществимо найти блок данных X, для которого H(X)=m;

– *стойкость к коллизиям первого рода*: для заданного сообщения должно быть вычислительно неосуществимо подобрать другое сообщение N, для которого H(N)=H(M);

– *стойкость к коллизиям второго рода*: должно быть вычислительно неосуществимо подобрать пару сообщений (M, M'), имеющих одинаковый хэш.

Следует отметить, что не доказано существование необратимых хэш-функций, для которых вычисление какого-либо прообраза заданного значения хэш-функции теоретически невозможно. Обычно нахождение обратного значения является лишь вычислительно сложной задачей.

Атака «дней рождения» позволяет находить коллизии для хэш-функции с длиной значений бит в среднем за $2^{n/2}$ вычислений хэш-функции. Поэтому *n*-битная хэш-функция считается криптостойкой, если вычислительная сложность нахождения коллизий для неё близка к $2^{n/2}$.

Для криптографических хэш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось (это свойство называется *лавинным эффектом*). В частности, значение хэша не должно давать утечки информации даже об отдельных битах аргумента. Это требование служит залогом криптостойкости алгоритмов, хэширующих пользовательский пароль для получения ключа.

Семейство алгоритмов SHA (Secure hash standard) включает в себя 5 алгоритмов вычисления хэш-функции: SHA–1, SHA–224, SHA–256, SHA–384, SHA–512. Четыре последние хэш-функции объединяются в подсемейство SHA–2. Алгоритм SHA–1 разработан Агентством национальной безопасности США (NSA) в 1995 году. Алгоритмы подсемейства SHA–2 также разработаны Агентством национальной безопасности США и опубликованы Национальным институтом стандартов и технологий в федеральном стандарте обработки информации FIPS PUB 180–2 в августе 2002 года. Эти алгоритмы используются в SSL, SSH, S/MIME, DNSSEC, X.509, PGP, IPSec, при передаче файлов по сети (BitTorrent) [5, 6, 7, 8, 9].

Между собой алгоритмы отличаются криптостойкостью, которая обеспечивается для хэшируемых данных, а также размерами блоков и слов данных, используемых при хэшировании.

При преобразовании последовательности бит в байты и слова используется порядок расположения бит big-endian. В частности, при преобразовании последовательности бит в число наиболее незначимые 4 бита числа представляются четырьмя самыми правыми битами.

В алгоритме используются побитные операции '$\wedge$', 'V', '$\neg$', '$\oplus$', '»'и '«'. Под операцией '+' понимается сложение по модулю 2^{32} (или 2^{64} в зависимости от длины слова), т.е. $(X+Y)(\mathrm{mod}2^{32})$.

Также используются следующие обозначения для операций сдвига:

– сдвиг вправо $SHR^n(x) \equiv x»n$;

– циклический сдвиг вправо $ROTR^n(x) \equiv (x»n)V(x«w-n)$, где w – длина слова;

– циклический сдвиг влево $ROTL^n(x) \equiv (x«n)V(x»w-n)$, где w – длина слова;

В алгоритме используются следующие функции:

$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$;

$Parity(x,y,z) = x \oplus y \oplus z$;

$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$;

$$f(x,y,z) = \begin{cases} Ch(x,y,z), & 0 \le t \le 19, \\ Parity(x,y,z), & 20 \le t \le 39, \\ Maj(x,y,z), & 40 \le t \le 59, \\ Parity(x,y,z), & 60 \le t \le 79. \end{cases};$$

$\sum_0^{(256)}(X) = PORT^2(x) \oplus PORT^{13}(x) \oplus PORT^{22}(x)$;

$\sum_1^{(256)}(X) = PORT^6(x) \oplus PORT^{11}(x) \oplus PORT^{25}(x)$;

$\sigma_0^{(256)}(x) = PORT^7(x) \oplus PORT^{18}(x) \oplus SHR^3(x)$;

$\sigma_1^{(256)}(x) = PORT^{17}(x) \oplus PORT^{19}(x) \oplus SHR^{10}(x)$;

$\sum_0^{(256)}(X) = PORT^{28}(x) \oplus PORT^{34}(x) \oplus PORT^{39}(x)$;

$\sum_1^{(256)}(X) = PORT^{14}(x) \oplus PORT^{18}(x) \oplus PORT^{41}(x)$;

$\sigma_0^{(256)}(x) = PORT^1(x) \oplus PORT^8(x) \oplus SHR^7(x)$;

$\sigma_1^{(256)}(x) = PORT^{19}(x) \oplus PORT^{61}(x) \oplus SHR^6(x)$;

Описание алгоритма SHA–256.

Пусть имеется сообщение M длиной в L бит, хэш-функцию которого нужно вычислить.

Шаг 1. *Добавление дополнительных бит.*

К концу сообщения добавляется бит «1», вслед за которым добавляются нулевые биты («0») до тех пор, пока длина L дополненного сообщения не станет удовлетворять условию:

$L \equiv 448 \pmod{512}$.

Таким образом, может быть добавлено от 1 до 512 бит.

Шаг 2. *Добавление исходной длины сообщения.*

К результату предыдущего шага добавляются младшие 64 бита, взятые из побитного представления числа L (исходная длина сообщения). При этом для конвертирования последовательности бит в число используется порядок расположения бит big-endian. В итоге получается сообщение, длина которого кратна 512 битам (1024 битам). Например, если длина слова – 32 бита, то сообщение "abc", имеющее длину 24 бита, будет дополнено до следующего сообщения (см. рисунок 2.1):

$$\underbrace{01100001}_{"a"}\ \underbrace{01100010}_{"b"}\ \underbrace{01100011}_{"c"}\ 1\ \overbrace{00\ldots00}^{423}\ \overbrace{00\ldots0\underbrace{11000}_{l=24}}^{64}$$

Рисунок 2.1 – Сообщение «abc», дополненное битами '1' и '0' и исходной длиной сообщения

Полученное сообщение разбивается на блоки длиной 512 (или 1024) бита, которые обозначаются $M^{(1)}$, $M^{(2)}$, … , $M^{(N)}$. Каждый из блоков подразделяется на 16 32-битных (или 64-битных) слов, обозначаемых $M_0^{(i)}$, $M_1^{(i)}$, … , $M_{15}^{(i)}$.

Дальнейшие шаги специфичны для каждого алгоритма и приводятся далее в соответствующих разделах.

Шаг 3. *Определение констант.*

SHA–256 использует последовательность 64 32-битных слов ($K_0^{(256)}$, $K_1^{(256)}$, ... , $K_{63}^{(256)}$). Эти слова представляют собой первые 32 бита дробной части кубических корней, взятых от первых 64 простых чисел. В 16-ричном виде эти константы выглядят так:

$K_0^{(256)}$=428a2f98 $K_1^{(256)}$=71374491 $K_2^{(256)}$=b5c0fbcf $K_3^{(256)}$=e9b5dba5
$K_4^{(256)}$=3956c25b $K_5^{(256)}$=59f111f1 $K_6^{(256)}$=923f82a4 $K_7^{(256)}$=ab1c5ed5
$K_8^{(256)}$=d807aa98 $K_9^{(256)}$=12835b01 $K_{10}^{(256)}$=243185be $K_{11}^{(256)}$=550c7dc3
$K_{12}^{(256)}$=72be5d74 $K_{13}^{(256)}$=80deb1fe $K_{14}^{(256)}$=9bdc06a7 $K_{15}^{(256)}$=c19bf174
$K_{16}^{(256)}$=e49b69c1 $K_{17}^{(256)}$=efbe4786 $K_{18}^{(256)}$=0fc19dc6 $K_{19}^{(256)}$=240ca1cc
$K_{20}^{(256)}$=2de92c6f $K_{21}^{(256)}$=4a7484aa $K_{22}^{(256)}$=5cb0a9dc $K_{23}^{(256)}$=76f988da
$K_{24}^{(256)}$=983e5152 $K_{25}^{(256)}$=a831c66d $K_{26}^{(256)}$=b00327c8 $K_{27}^{(256)}$=bf597fc7
$K_{28}^{(256)}$=c6e00bf3 $K_{29}^{(256)}$=d5a79147 $K_{30}^{(256)}$=06ca6351 $K_{31}^{(256)}$=14292967
$K_{32}^{(256)}$=27b70a85 $K_{33}^{(256)}$=2e1b2138 $K_{34}^{(256)}$=4d2c6dfc $K_{35}^{(256)}$=53380d13
$K_{36}^{(256)}$=650a7354 $K_{37}^{(256)}$=766a0abb $K_{38}^{(256)}$=81c2c92e $K_{39}^{(256)}$=92722c85
$K_{40}^{(256)}$=a2bfe8a1 $K_{41}^{(256)}$=a81a664b $K_{42}^{(256)}$=c24b8b70 $K_{43}^{(256)}$=c76c51a3
$K_{44}^{(256)}$=d192e819 $K_{45}^{(256)}$=d6990624 $K_{46}^{(256)}$=f40e3585 $K_{47}^{(256)}$=106aa070
$K_{48}^{(256)}$=19a4c116 $K_{49}^{(256)}$=1e376c08 $K_{50}^{(256)}$=2748774c $K_{51}^{(256)}$=34b0bcb5
$K_{52}^{(256)}$=391c0cb3 $K_{53}^{(256)}$=4ed8aa4a $K_{54}^{(256)}$=5b9cca4f $K_{55}^{(256)}$=682e6ff3
$K_{56}^{(256)}$=748f82ee $K_{57}^{(256)}$=78a5636f $K_{58}^{(256)}$=84c87814 $K_{59}^{(256)}$=8cc70208
$K_{60}^{(256)}$=90befffa $K_{61}^{(256)}$=a4506ceb $K_{62}^{(256)}$=bef9a3f7 $K_{63}^{(256)}$=c67178f2

Начальное значение хэш-функции устанавливается следующими константами:

$H_0^{(0)}$= 6a09e667, $H_1^{(0)}$= bb67ae85, $H_2^{(0)}$= 3c6ef372, $H_3^{(0)}$= a54ff53a,
$H_4^{(0)}$= 510e527f, $H_5^{(0)}$= 9b05688c, $H_6^{(0)}$= 1f83d9ab, $H_7^{(0)}$= 5be0cd19.

Шаг 4. *Основной цикл.*

В следующем цикле последовательно обрабатываются все блоки, на которые было разделено дополненное сообщение.

```
For i=1 to N
   {
     // 1. Подготовка списка преобразованных слов сообщения
```

$$W_t = \begin{cases} M_t, & 0 \leqslant t \leqslant 15, \\ \sigma_1^{(256)}(W_{t-2}) + W_{t-7} + \sigma_0^{(256)}(W_{t-15}) + W_{t-16}, & 16 \leqslant t \leqslant 63. \end{cases}$$

// 2. Инициализация рабочих переменных
$a=H_0^{(i-1)}$ $b=H_1^{(i-1)}$ $c=H_2^{(i-1)}$ $d=H_3^{(i-1)}$
$e=H_4^{(i-1)}$ $f=H_5^{(i-1)}$ $g=H_6^{(i-1)}$ $h=H_7^{(i-1)}$

// 3. Внутренний цикл
For t=0 to 63
{
$T_1=h+\Sigma_1^{(256)}(e)+Ch(e, f, g)+K_t^{(256)}+W_t$
$T_2=\Sigma_0^{(256)}(a)+Maj(a, b, c)$
h=g
g=f
f=e
$e=d+T_1$
d=c
c=b
b=a
$a=T_1+T_2$
}

// 4. Вычисление промежуточного значения хэш-функции
$H_0^{(i)}=a+H_0^{(i-1)}$ $H_1^{(i)}=b+H_1^{(i-1)}$ $H_2^{(i)}=c+H_2^{(i-1)}$ $H_3^{(i)}=d+H_3^{(i-1)}$
$H_4^{(i)}=e+H_4^{(i-1)}$ $H_5^{(i)}=f+H_5^{(i-1)}$ $H_6^{(i)}=g+H_6^{(i-1)}$ $H_7^{(i)}=h+H_7^{(i-1)}$
}

Шаг 5. *Результат.*
Результатом вычисления хэш-функции от всего сообщения является:

$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$.

Примеры значений хэш-функции (в 16-ричном виде):

SHA256 ("") = e3b0c442 98fc1c14 9afbf4c8 996fb924 27ae41e4 649b934c a495991b 7852b855

SHA256 ("abc") = ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad

SHA256 («abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq») = 248d6a61 d20638b8 e5c02693 0c3e6039 a33ce459 64ff2167 f6ecedd4 19db06c1

2.3 Содержание отчета

Отчет должен включать в себя следующие пункты:
1) Исходный текст, для которого вычисляется хэш-функция.

2) Вычисленная для исходного текста хэш-функция.

3) Результаты исследования лавинного эффекта, выполненные в соответствие со всеми требованиями, изложенными в пунктах 2.2.1 и 2.2.2.

4) Выводы о проделанной работе.

2.4 Контрольные вопросы

1. Что такое хэш-функция?
2. Какие требования предъявляются к хэш-функциям?
3. Когда хэш-функция является криптографически стойкой?
4. Что такое лавинный эффект?
5. Семейство алгоритмов SHA.
6. Какие ещё существуют алгоритмы расчета хэш-функций кроме SHA?
7. Из каких условий выбирается размер (длина в битах) хэш-функции?
8. Преимущества и недостатки хэш-функции SHA по сравнению с другими хэш-функциями.

3 Лабораторная работа № 3. Стандарт симметричного шифрования ГОСТ 28147-89

Цель работы: изучить алгоритм симметричного шифрования информации ГОСТ 28147-89, познакомиться с критериями оценки свойств лавинного эффекта.

3.1 Порядок выполнения работы

3.1.1 Реализовать приложение для шифрования, позволяющее выполнять следующие действия:

– шифровать данные по алгоритму ГОСТ 28147-89:

а) шифруемый текст должен храниться в одном файле, а ключ шифрования – в другом;

б) зашифрованный текст должен сохраняться в файл;

в) в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде;

г) программа должна показывать время шифрования;

– исследовать лавинный эффект (исследования проводить на одном блоке текста):

а) для бита, который будет изменяться, приложение должно позволять задавать его позицию (номер) в открытом тексте или в ключе;

б) приложение должно уметь после каждого раунда шифрования подсчитывать число бит, изменившихся в зашифрованном тексте при изменении одного бита в открытом тексте либо в ключе;

в) приложение может строить графики зависимости числа бит, изменившихся в зашифрованном тексте, от раунда шифрования, либо графики можно строить в стороннем программном обеспечении, но тогда приложение для шифрования должно сохранять в файл необходимую для построения графиков информацию.

3.1.2 Реализовать приложение для дешифрования с учетом следующих требований:

– зашифрованный текст должен храниться в одном файле, ключ – в другом;

– расшифрованный текст должен сохраняться в файл;

– в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.

3.1.3 Реализовать приложение, вычисляющее значения 1–4 критериев оценивающих рассеивающие свойства алгоритмов приведенных в пункте 3.2 для алгоритма ГОСТ 28147-89.

3.1.4 С помощью реализованных приложений выполнить следующие задания:

– протестировать правильность работы разработанных приложений;

– исследовать лавинный эффект при изменении одного бита в открытом тексте и в ключе: построить графики зависимостей числа бит, изменившихся в зашифрованном сообщении, от раунда шифрования (всего должно быть построено 2 графика);

– для алгоритма ГОСТ 28147-89 сравнить значения критериев 1–4 оценивающих рассеивающие свойства алгоритмов при изменении одного бита в блоке открытого текста и одного бита в ключе при использовании одного и того же сообщения. Сообщение должно состоять хотя бы из пяти блоков (чем больше, тем точнее будут оценки критериев 1–4);

– сделать выводы о проделанной работе.

3.2 Основные теоретические сведения

ГОСТ 28147-89 – это стандарт симметричного шифрования, принятый в 1989 году в Советском Союзе и установивший алгоритм шифрования данных, составляющих государственную тайну [10]. В начале 90-х годов алгоритм стал полностью открытым. В Российской Федерации установлен единый стандарт криптографического преобразования текста для информационных систем. Он рекомендован к использованию для защиты любых данных, представленных в виде двоичного кода, хотя не исключаются и другие методы шифрования. Данный стандарт формировался с учётом мирового опыта, и, в частности, были приняты во внимание недостатки и нереализованные возможности алгоритма DES, поэтому использование алгоритма ГОСТ 28147-89 предпочтительнее.

В алгоритме ГОСТ 28147-89 блок текста, подлежащий шифрованию, сначала разбивается на левую половину L и правую половину R. На этапе i используется подключ K_i. На этапе i алгоритма ГОСТ 28147-89 выполняется следующее:

$L_i=R_{i-1}$,

$R_i=L_{i-1}\oplus f(R_{i-1}, K_{i)}$.

После 32 раунда выполняется перестановка левой и правой частей полученного шифротекста: $L_{32}R_{32}\rightarrow R_{32}L_{32}$.

Один такт алгоритма ГОСТ 28147-89 показан на рисунке 3.1. Один такт алгоритма осуществляет одно преобразование Фейстеля [11].

Формирование функции f() начинается с того, что сначала правый блок R_{i-1} складывается по модулю 2^{32}с подключом K_i . Полученное 32-битное сообщение делится на восемь 4-битных чисел. Каждое из этих 4-битных чисел преобразуется соответствующим S-блоком в другое 4-битное число. Поэтому любой S-блок определяется некоторой 16-битной перестановкой на множестве из 16 элементов 0, 1, … , 15. В ГОСТ использовались следующие S-блоки:

S1 = (4 10 9 2 13 8 0 14 6 11 1 12 7 15 5 3),
S2 = (14 11 4 12 6 13 15 10 2 3 8 1 0 7 5 9),
S3 = (5 8 1 13 10 3 4 2 14 15 12 7 6 0 9 11),
S4 = (7 13 10 1 0 8 9 15 14 4 6 12 11 2 5 3),
S5 = (6 12 7 1 5 15 13 8 4 10 9 14 0 3 11 2),
S6 = (4 11 10 0 7 2 1 13 3 6 8 5 9 12 15 14),
S7 = (13 11 4 1 3 15 5 9 0 10 14 7 6 8 2 12),
S8 = (l 15 13 0 5 7 10 4 9 2 3 14 6 11 8 12).

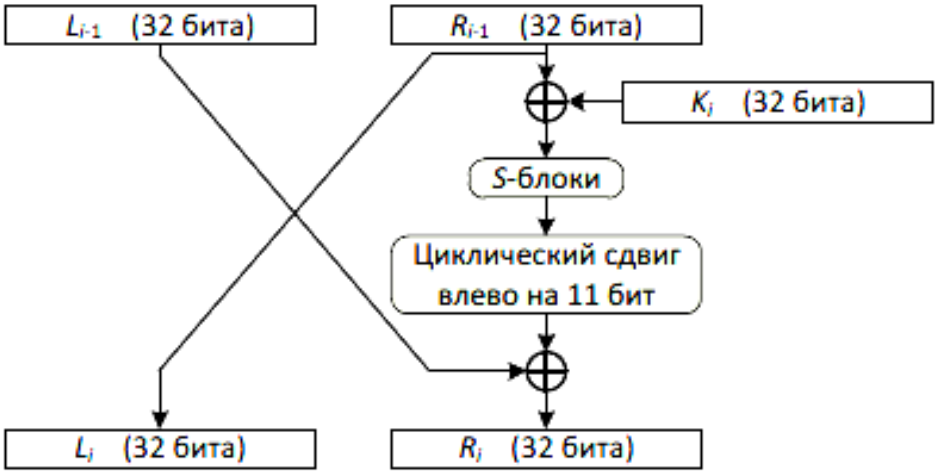

Рисунок 3.1 – Один такт алгоритма ГОСТ 28147-89

После преобразования S-блоками полученное 32-битное сообщение сдвигается циклически влево на 11 бит.

Для формирования подключей ГОСТ 28147-89 исходный 256-битный ключ делится на восемь 32-битных подключей K_i, $i=\overline{1,8}$. Они используются в 32 тактах в следующем порядке:

1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1.

При дешифровании порядок использования подключей меняется на противоположный.

Критерии оценки свойств «лавинного эффекта».

Пусть $X^{(i)}=X\oplus E_i$, т.е. бинарный вектор, полученный инвертированием *i*-го бита в векторе $X \in U$. Тогда бинарный вектор

$Y^{(i)}\oplus Y=F(X^{(i)}, K)\oplus F(X, K)$

называется *лавинным вектором по компоненту* (здесь F – функция шифрования). Пусть для рассматриваемых критериев X – это 1 блок открытого текста, размерность вектора X равна n, а Y – m и пусть U – это множество, состоящее из всех блоков открытого текста.

Введём следующее обозначение *мощности множества A*: #A, т.е. #A – это количество элементов в множестве.

Матрица зависимостей имеет следующий вид: $a_{ij}=\#\{X|Y_j^{(i)}\neq Y(j)\}$. Эта матрица отражает зависимость *j*-го разряда выходного вектора от *i*-го разряда входного вектора.

Матрица расстояний имеет вид: $b_{ij}=\#\{Y^{(i)}|w(Y^{(i)}\oplus Y)=j\}$, где w– функция веса Хэмминга (число неравных нулю элементов вектора).

Существует 4 критерия, по которым предлагается проверять рассеивающие свойства блочных алгоритмов:

1) Среднее число бит выхода, изменяющихся при изменении одного бита входного вектора. Это число оценивается по формуле:

$$d_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{\sum_{j=1}^{m}(j\cdot b_{ij})}{\#U}.$$

2) Степень полноты преобразования:

$$d_2 = 1 - \frac{\#\{(i,j)|a_{ij}=0\}}{n\cdot m}.$$

3) Степень лавинного эффекта:

$$d_3 = 1 - \frac{\sum_{i=1}^{n}\left|\sum_{j=1}^{m}2jb_{ij}-m\right|}{n\cdot m}, \text{ где } N=\#U.$$

4) Степень соответствия строгому лавинному критерию:

$$d_4 = 1 - \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left|\frac{2a_{ij}}{N}-1\right|}{n\cdot m}, \text{ где } N=\#U.$$

При исследовании диффузии, т.е. влияния бит исходного (открытого) текста на преобразованный (зашифрованный) текст, элементы матриц зависимостей и расстояний имеют вид:

$a_{ij}=\#\{X|(f(X^{(i)}, K))_j\neq(f(X, K))_j\}$,

$b_{ij}=\#\{X|w(f(X^{(i)}, K)\oplus f(X, K))=j\}$.

При исследовании конфузии, т.е. влияния бит ключа на преобразованный (зашифрованный) текст, элементы матриц зависимостей и расстояний имеют вид:

$a_{ij}=\#\{X|(f(X, K^{(i)}))_j \neq (f(X, K))_j\}$,

$b_{ij}=\#\{X|w(f(X, K^{(i)}) \oplus f(X, K))=j\}$.

3.3 Содержание отчета

Отчет должен включать в себя следующие пункты:

1) Исходный текст, для шифрования.

2) Зашифрованный текст по алгоритму ГОСТ 28147-89 и результат дешифрования.

3) Результаты исследования лавинного эффекта, выполненные в соответствие со всеми требованиями, изложенными в пункте 3.2.1.

4) Вычисленные значения критериев оценивающих рассеивающие свойства алгоритмов.

5) Выводы о проделанной работе.

3.4 Контрольные вопросы

1. Что представляет собой функция f() в алгоритме ГОСТ?
2. Как используется заданный ключ при шифровании в алгоритме ГОСТ?
3. Общие черты и отличия алгоритмов ГОСТ и DES.
4. Критерии оценки свойств лавинного эффекта.
5. Критерии оценки рассеивающих свойств алгоритмов.
6. Преобразование Фейстеля.
7. Назначение S-блока.
8. Как размер блока шифрования, длина ключа и число раундов влияют на криптостойкость алгоритмов?

4 Лабораторная работа № 4. Исследование алгоритма шифрования информации RIJNDAEL и стандарт AES

Цель работы: изучить криптографический стандарт шифрования AES и его особенности, познакомиться с различными режимами блочного шифрования.

4.1 Порядок выполнения работы

4.1.1 Реализовать приложение для шифрования, позволяющее выполнять следующие действия:

– шифровать данные по алгоритму AES:

а) шифруемый текст должен храниться в одном файле, а ключ шифрования – в другом;

б) зашифрованный текст должен сохраняться в файл;

в) в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде;

г) программа должна показывать время шифрования.

– исследовать лавинный эффект (исследования проводить на одном блоке текста):

а) для бита, который будет изменяться, приложение должно позволять задавать его позицию (номер) в открытом тексте или в ключе;

б) приложение должно уметь после каждого раунда шифрования подсчитывать число бит, изменившихся в зашифрованном тексте при изменении одного бита в открытом тексте либо в ключе;

в) приложение может строить графики зависимости числа бит, изменившихся в зашифрованном тексте, от раунда шифрования, либо графики можно строить в стороннем программном обеспечении, но тогда приложение для шифрования должно сохранять в файл необходимую для построения графиков информацию.

4.1.2 Реализовать приложение для дешифрования с учетом следующих требований:

– зашифрованный текст должен храниться в одном файле, ключ – в другом;

– расшифрованный текст должен сохраняться в файл;

– в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.

4.1.3 Реализовать приложение, вычисляющее значения 1–4 критериев оценивающих рассеивающие свойства алгоритмов приведенных в пункте 3.1.

4.1.4 С помощью реализованных приложений выполнить следующие задания:

– протестировать правильность работы разработанных приложений;

– исследовать лавинный эффект при изменении одного бита в открытом тексте и в ключе: построить графики зависимостей числа бит, изменившихся в зашифрованном сообщении, от раунда шифрования (всего должно быть построено 2 графика);

– для алгоритма AES сравнить значения критериев 1–4 оценивающих рассеивающие свойства алгоритмов при изменении одного бита в блоке открытого текста и одного бита в ключе при использовании одного и того же сообщения. Сообщение должно состоять хотя бы из пяти блоков (чем больше, тем точнее будут оценки критериев 1–4);

– сделать выводы о проделанной работе.

4.2 Основные теоретические сведения

26 ноября 2001 г. была опубликована окончательная версия стандарта FIPS-197, описывающего новый американский стандарт шифрования AES. Высочайшую надежность AES подтверждается астрономическими числами, так 128-битный ключ обеспечивает 340 андециллионов возможных комбинаций, а 256-битный ключ ещё увеличивает это число.

Стандарт шифрования AES базируется на алгоритме шифрования RIJNDAEL. RIJNDAEL - это итерационный блочный шифр, имеющий архитектуру «Квадрат». Шифр имеет переменную длину блоков и различные длины ключей. Длина ключа и длина блока могут быть равны независимо друг от друга 128, 192 или 256 битам. В стандарте AES определена длина блока данных, равная 128 битам.

Введем следующие обозначения:

а) Nb - число 32-битных слов содержащихся во входном блоке, Nb=4;

б) Nk - число 32-битных слов содержащихся в ключе шифрования, Nk=4,6,8;

в) Nr - число раундов шифрования, как функция от Nb и Nk, Nr=10,12,14.

Входные (input), промежуточные (state) и выходные (output) результаты преобразований, выполняемых в рамках криптоалгоритма, называются состояниями (State). Состояние можно представить в виде прямоугольного массива байтов (см. рисунок 4.1). При размере блока, равном 128 битам, этот 16-байтовый массив (см. рисунок 4.2, а) имеет 4 строки и 4 столбца (каждая строка и каждый столбец в этом случае могут рассматриваться как 32-разрядные слова). Входные данные для шифра обозначаются как байты состояния в порядке S_{00}, S_{10}, S_{20}, S_{30}, S_{01}, S_{11}, S_{21}, S_{31}.

State

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

Рисунок 4.1 – Пример представления 128-разрядного блока данных в виде массива State, где а – байт блока данных, а каждый столбец – одно 32-разрядное слово

s_{00}	s_{01}	s_{02}	s_{03}
s_{10}	s_{11}	s_{12}	s_{13}
s_{20}	s_{21}	s_{22}	s_{23}
s_{30}	s_{31}	s_{32}	s_{33}

а

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}

б

Рисунок 4.2 – Форматы данных: а – пример представления блока (Nb, = 4); б – ключа шифрования (Nk = 4), где s и k – соответственно байты массива State и ключа, находящиеся на пересечении i-й строки и j-то столбца

После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке. В общем случае число столбцов Nb равно длине блока, деленной на 32.

Рисунок 4.3 демонстрирует такое представление, носящее название архитектуры «Квадрат».

Input

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

State

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

output

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Рисунок 4.3 – Пример представления блока в виде матрицы 4Nb

Ключ шифрования также представлен в виде прямоугольного массива с четырьмя строками (см. рисунок 4.2, б). Число столбцов этого массива равно длине ключа, деленной на 32. В стандарте определены ключи всех трех размеров - 128 бит, 192 бита и 256 бит, то есть соответственно 4, 6 и 8 32-разрядных слова (или столбца – в табличной форме представления). В некоторых случаях ключ шифрования рассматривается как линейный массив 4-байтовых слов. Слова состоят из 4 байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

Ключ шифрования также представлен в виде прямоугольного массива с четырьмя строками (см. рисунок 4.2, б). Число столбцов этого массива равно длине ключа, деленной на 32. В стандарте определены ключи всех трех размеров - 128 бит, 192 бита и 256 бит, то есть соответственно 4, 6 и 8 32-разрядных слова (или столбца – в табличной форме представления). В некоторых случаях ключ шифрования рассматривается как линейный массив

4-байтовых слов. Слова состоят из 4 байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

Число раундов Nr в алгоритме RIJNDAEL зависит от значений Nb и Nk, как показано в таблице 4.1. В стандарте AES определено соответствие между размером ключа, размером блока данных и числом раундов шифрования, как показано в таблице 4.2.

Т а б л и ц а 4.1 – Число раундов Nr как функция от длины ключа Nk и длины блока Nb

Nr	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

Раунд состоит из четырех различных преобразований:

– замены байтов SubBytes() - побайтовой подстановки в S-блоках с фиксированной таблицей замен размерностью 8 x 256;

– сдвига строк ShiftRows() - побайтового сдвига строк массива State на различное количество байт;

– перемешивания столбцов MixColumns() — умножения столбцов состояния, рассматриваемых как многочлены над $GF(2^8)$, на многочлен третьей степени g(x) по модулю $x^4 + 1$;

– сложения с раундовым ключом AddRoundKey() - поразрядного XOR с текущим фрагментом развернутого ключа.

Т а б л и ц а 4.2 – Соответствие между длиной ключа, размером блока данных и числом раундов в стандарте AES

Стандарт	Длина ключа (Nk слов)	Размер блока данных (Nb слов)	Число раундов (Nr)
AES-128	10	12	14
AES-192	12	12	14
AES-256	14	14	14

Замена байтов (SubBytes). Преобразование SubBytes() представляет собой нелинейную замену байтов, выполняемую независимо с каждым байтом состояния. Таблицы замены S-блока являются инвертируемыми и построены из композиции следующих двух преобразований входного байта:

1) получение обратного элемента относительно умножения в поле $GF(2^8)$, нулевой элемент {00} переходит сам в себя;

2) применение преобразования над $GF(2^8)$, определенного следующим образом:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Другими словами суть преобразования может быть описана уравнениями:

$b'_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$,

где $c_0=c_1=c_5=c_6=0$, $c_2=c_3=c_4=c_7=1$, b'_i и b_i - соответственно преобразованное и исходное значение i-го бита, i = 0,1,2….7.

Применение описанного S-блока ко всем байтам состояния обозначается как SubBytes(State). Рисунок 4.4 иллюстрирует применение преобразования SubBytes() к состоянию. Логика работы S-блока при преобразовании байта {xy} отражена в таблице 4.3. Например, результат {ed} преобразования байта {53} находится на пересечении 5-й строки и 3-го столбца.

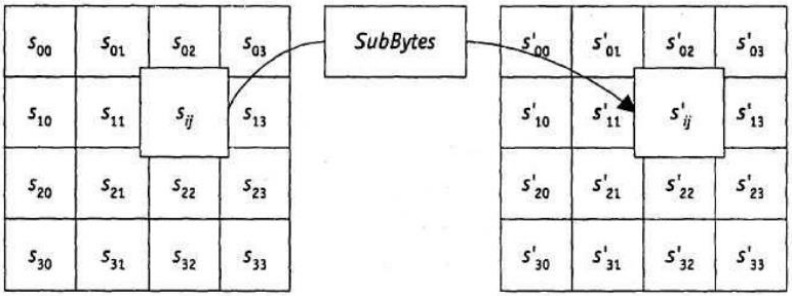

Рисунок 4.4 – SubBytes() действует на каждый байт состояния

Преобразование сдвига строк (ShiftRows). Последние 3 строки состояния циклически сдвигаются влево на различное число байтов. Строка 1 сдвигается 173 на C1 байт, строка 2 - на C2 байт, и строка 3 - на C3 байт. Значения сдвигов C1, C2 и C3 в RIJNDAEL зависят от длины блока Nb. Их величины приведены в таблице 4.4.

В стандарте AES, где определен единственный размер блока, равный 128 битам, C1 = 1, C2 = 2 и C3 = 3.

Операция сдвига последних трех строк состояния обозначена как ShiftRows(State). Рисунок 4.5 показывает влияние преобразования на состояние.

Преобразование перемешивания столбцов (MixColumns). В этом преобразовании столбцы состояния рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен g(x), выглядящий следующим образом:

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

Т а б л и ц а 4.3 – Таблица замен S-блока

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

Т а б л и ц а 4.4 – Величина сдвига для разной длины блоков

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	3

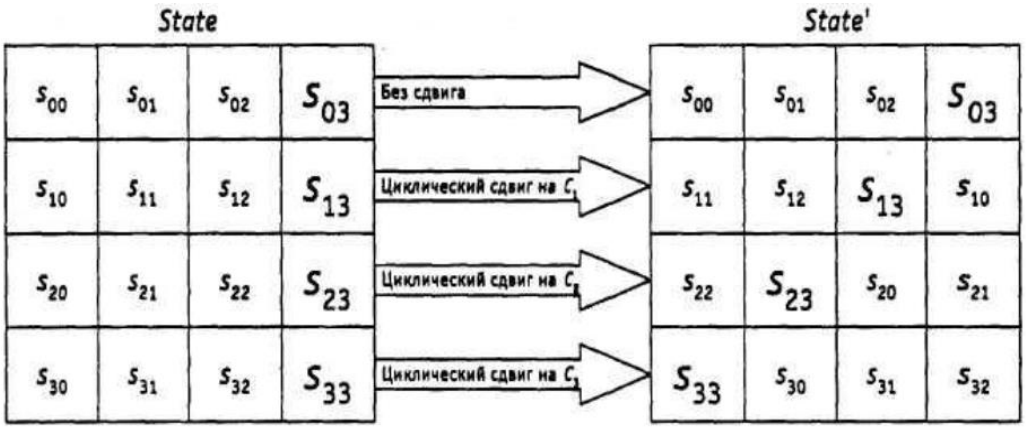

Рисунок 4.5 – ShiftRows() действует на строки состояния

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \leqslant c \leqslant 3,$$

где c - номер столбца массива State.

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

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

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

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

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

Применение этой операции ко всем четырем столбцам состояния обозначено как MixColumns(State). Рисунок 4.6 демонстрирует применение преобразования MixColumnsQ к столбцу состояния.

Добавление раундового ключа (AddRoundKey). В данной операции раундовый ключ добавляется к состоянию посредством простого поразрядного XOR. Раундовый ключ вырабатывается из ключа шифрования посредством алгоритма выработки ключей (key schedule). Длина раундового ключа (в 32-разрядных словах) равна длине блока Nb. Преобразование, содержащее добавление посредством XOR раундового ключа к состоянию (см. рисунок 4.7), обозначено как AddRoundKey (State, RoundKey).

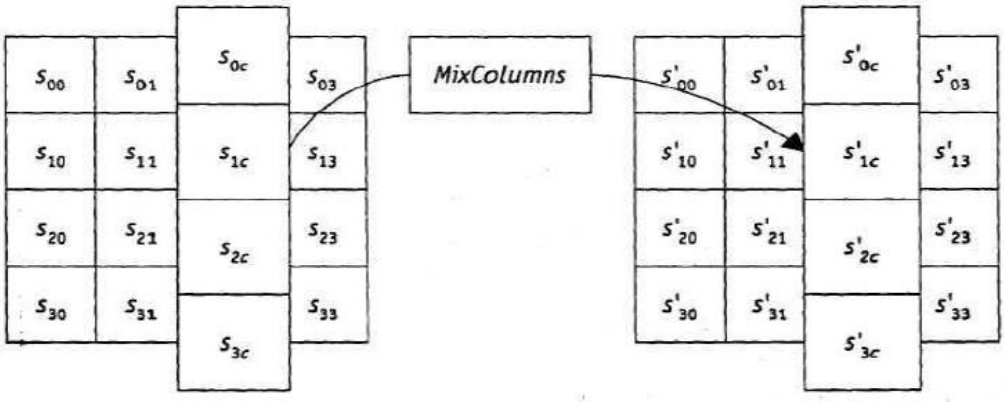

Рисунок 4.6 – MixColumnsQ действует на столбцы состояния

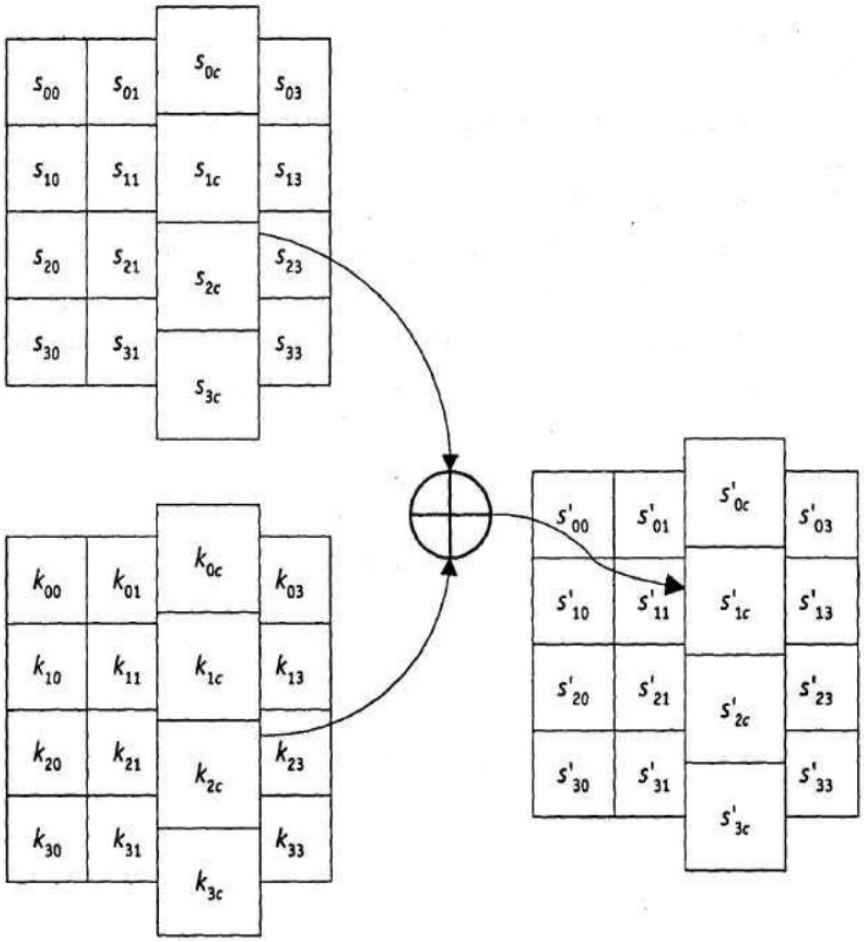

Рисунок 4.7 – При добавлении ключа раундовый ключ складывается посредством операции XOR с состоянием

Алгоритм выработки ключей (Key Schedule). Раундовые ключи получаются из ключа шифрования посредством алгоритма выработки ключей. Он содержит два компонента: расширение ключа (Key Expansion) и выбор раундового ключа (Round Key Selection). Основополагающие принципы алгоритма выглядят следующим образом:

– общее число битов раундовых ключей равно длине блока, умноженной на число раундов, плюс 1 (например, для длины блока 128 бит и 10 раундов требуется 1408 бит раундовых ключей);

– ключ шифрования расширяется в расширенный ключ (Expanded Key);

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

Расширение ключа (Key Expansion). Расширенный ключ в RIJNDAEL представляет собой линейный массив w[i] из Nb(Nr+1) 4- байтовых слов, i = 0,1…Nb(Nr +1). В AES массив w[i] состоит из 4(Nr+1) 4-байтовых слов, i = 0,1…4(Nr +1).

Первые Nk слов содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами. Алгоритм выработки ключей зависит от величины Nk.

Как можно заметить (см. рисунок 4.8, а), первые Nk слов заполняются ключом шифрования. Каждое последующее слово w[i] получается посредством операции XOR предыдущего слова w[i-l] и слова на Nk позиций ранее w[i-Nk]'.

$w[i] = w[i - 1] \oplus w[i - N_k]$.

Для слов, позиция которых кратна Nk, перед XOR применяется преобразование к w[i-1], а затем еще прибавляется раундовая константа Rcon. Преобразование реализуется с помощью двух дополнительных функций: RotWord(), осуществляющей побайтовый сдвиг 32-разрядного слова по формуле {a0 a1 a2 a3} → {a1 a2 a3 a0}, и SubWord() осуществляющей побайтовую замену с использованием 5-блока функции SubBytes(). Значение Rcon[j] равно 2^{j-l}. Значение w[i] в этом случае равно

$w[i] = SubWord(RotWord(w[i - 1])) \oplus Rcon[i/Nk] \oplus w[i - Nk]$.

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	…

а)

Раундовый ключ 0	Раундовый ключ 1	Раундовый ключ 2	……..

б)

а - расширения ключа (светло-серым цветом выделены слова расширенного ключа, которые формируются без использования функций SubWord() и RotWordQ; темно-серым цветом выделены слова расширенного ключа, при вычислении которых используются преобразования SubWordQ и RotWordQ); б - выбора раундового ключа для Nk – 4.

Рисунок 4.8 – Процедуры

Выбор раундового ключа (Round Key Selection). Раундовый ключ i получается из слов массива раундового ключа от W[Nb i] и до W[Nb (i + 1)], как показано на рисунке 4.8.

Функция шифрования шифра RIJNDAEL состоит (см. рисунок 4.9):

– из начального добавления раундового ключа;

– Nr - 1 раундов;

– заключительного раунда, в котором отсутствует операция MixColumns().

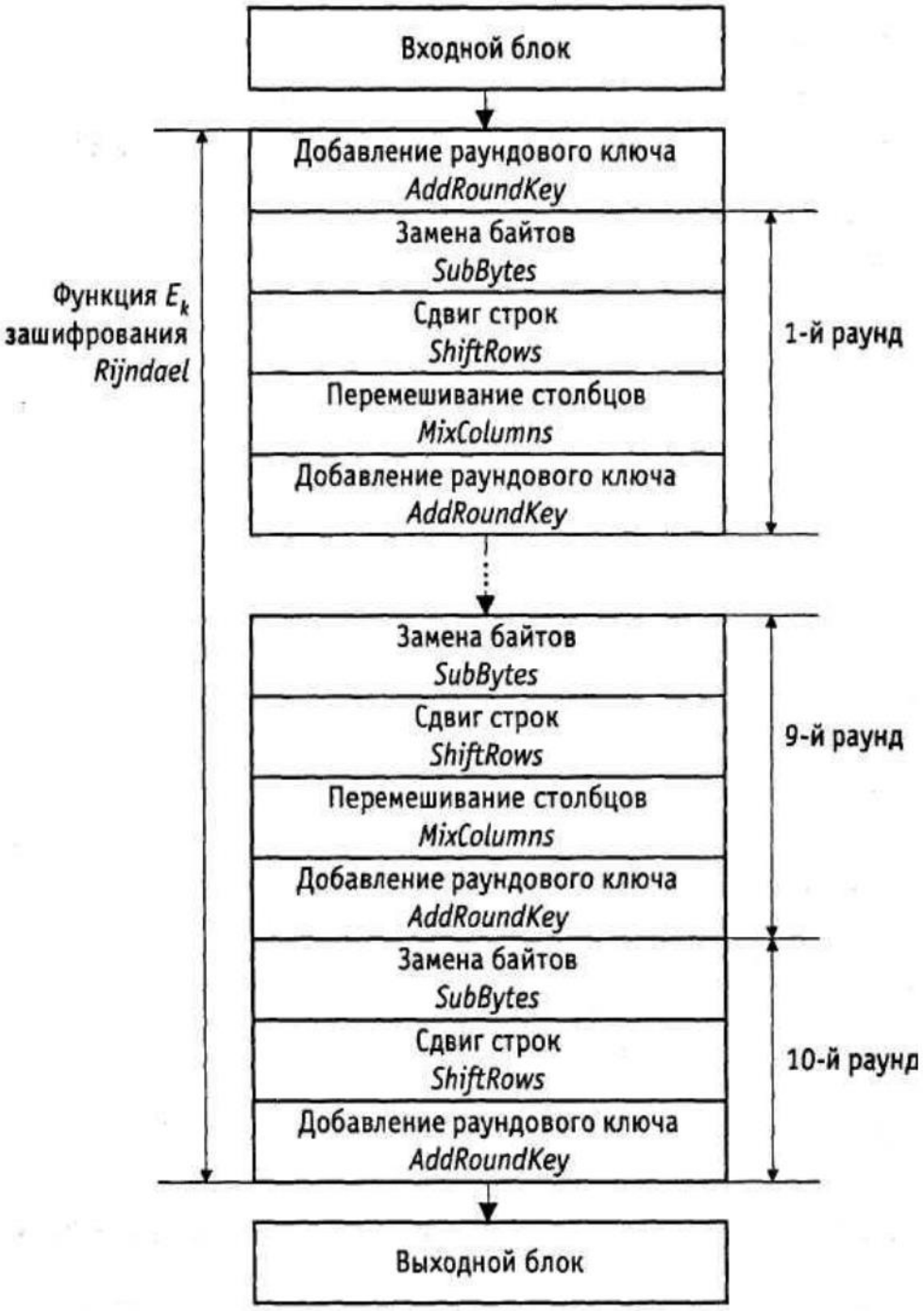

Рисунок 4.9 – Схема функции Ек зашифрования криптоалгоритма RIJNDAEL при Nk = Nb = 4

На вход алгоритма подаются блоки данных State, в ходе преобразований содержимое блока изменяется и на выходе образуется шифротекст, организованный опять же в виде блоков State, как показано на рисунке 4.2, б и 4.10, где Nb - 4, in_m и out_m – m-е байты соответственно входного и выходного блоков, m=0,1…15, s_{ij} – байт, находящийся на пересечении i-й строки и j-го столбца массива State, i = j = 0,1…3.

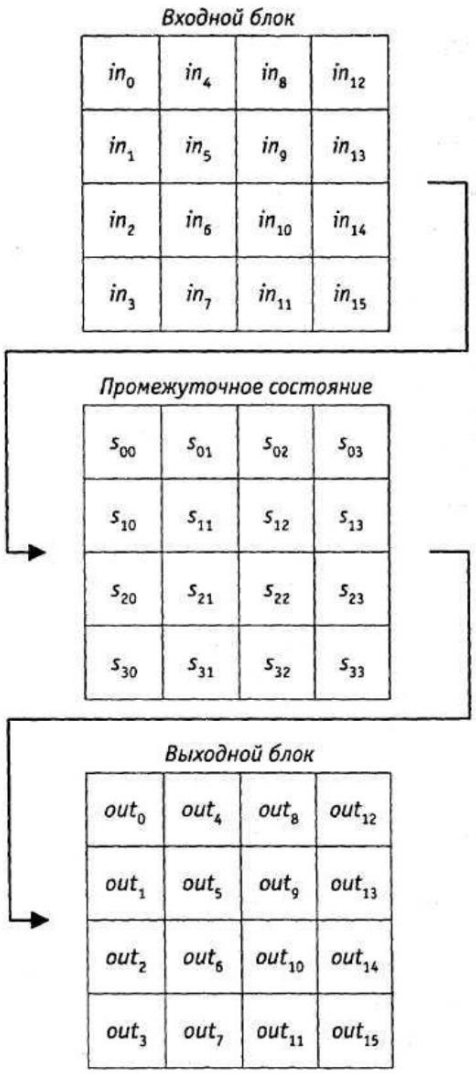

Рисунок 4.10 – Ход преобразования данных, организованных в виде блоков State

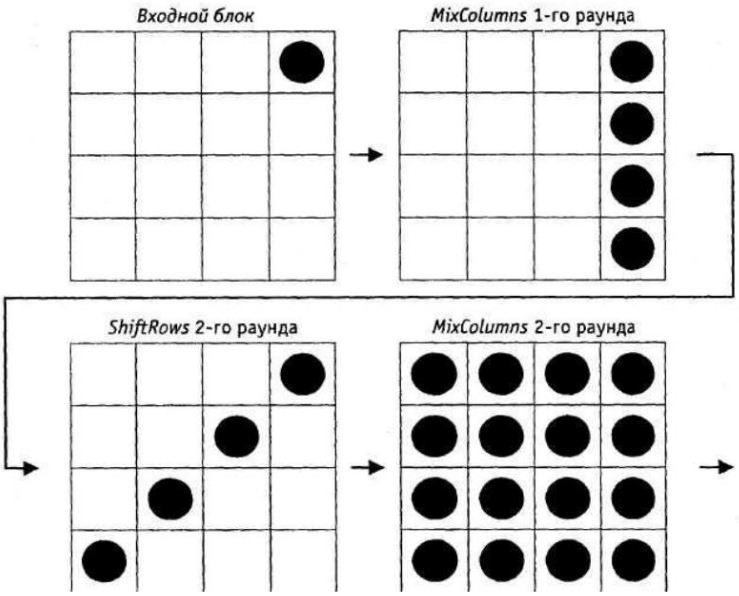

Рисунок 4.11 – Принцип действия криптоалгоритма RIJNDAEL;

- измененный байт

Перед началом первого раунда происходит суммирование по модулю 2 с начальным ключом шифрования, затем - преобразование массива байтов State в течение 10, 12 или 14 раундов в зависимости от длины ключа. Последний раунд несколько отличается от предыдущих тем, что не задействует функцию перемешивания байт в столбцах MixColumns().

Рисунок 4.11 демонстрирует и рассеивающие и перемешивающие свойства шифра. Видно, что два раунда обеспечивают полное рассеивание и перемешивание.

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

Далее приводится описание функций, обратных используемым при прямом шифровании.

Функция AddRoundKey() обратна сама себе, учитывая свойства используемой в ней операции XOR.

Преобразование InvSubBytes. Логика работы инверсного S-блока при преобразовании байта {xy} отражена в таблице 4.5.

Т а б л и ц а 4.5 – Таблица замен инверсного S-блока

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

Преобразование InvShiftRows. Последние 3 строки состояния циклически сдвигаются вправо на различное число байтов. Строка 1

сдвигается на C1 байт, строка 2 - на C2 байт, и строка 3 - на C3 байт. Значения сдвигов C1, C2, C3 зависят от длины блока Nb. Их величины приведены в таблице 4.4.

Преобразование InvMixColumns. В этом преобразовании столбцы состояния рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен $g^{-1}(x)$, выглядящий следующим образом:

$g^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \leqslant c \leqslant 3.$$

В результате на выходе получаются следующие байты

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

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

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

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

Функция прямого дешифрования. Алгоритм обратного дешифрования, описанный выше, имеет порядок приложения операций-функций, обратный порядку операций в алгоритме прямого шифрования, но использует те же параметры (развернутый ключ). Однако некоторые свойства алгоритма шифрования RIJNDAEL позволяют применить для дешифрования тот же порядок приложения функций (обратных используемым для шифрования) за счет изменения некоторых параметров, а именно - развернутого ключа.

Два следующих свойства позволяют применить алгоритм прямого дешифрования.

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

Операция MixColumns() является линейной относительно входных данных, что означает

InvMixColumns(State XOR RoundKey) =

= lnvMixColumns(State) XOR InvMixColumns(RoundKey)

Эти свойства функций алгоритма шифрования позволяют изменить порядок применения функций InvSubBytes() и InvShiftRows(). Функции AddRounKey() и InvMixColwnns() также могут быть применены в обратном порядке, но при условии, что столбцы (32-битные слова) развернутого ключа дешифрования предварительно пропущены через функцию invMixColumns().

Таким образом, можно реализовать более эффективный способ дешифрования с тем же порядком приложения функций, как и в алгоритме шифрования.

Примечание. В последнем операторе (в функции InvMixColumn()) происходит преобразование типа, так как развернутый ключ хранится в виде линейного массива 32-разрядных слов, в то время как входной параметр функции - двумерный массив байтов.

В таблице 4.6 приведена процедура шифрования, а также два эквивалентных варианта процедуры дешифрования при использовании двухраундового варианта RIJNDAEL. Первый вариант функции дешифрования суть обычная инверсия функции шифрования. Второй вариант функции дешифрования получен из первого после изменения порядка следования операций в трех парах преобразований:

invShiftRows — InvSubBytes (дважды)

и AddRoundKey — InvMixColumns.

Таблица 4.6 – Последовательность преобразований в двухраундовом варианте RIJNDAEL

Функция шифрования двухраундового варианта RIJNDAEL	Функция дешифрования двухраундового варианта RIJNDAEL	Эквивалентная функция прямого дешифрования двухраундового варианта RIJNDAEL
AddRoundKey	AddRoundKey	AddRoundKey
SubBytes	invShiftRows	InvSubBytes
ShiftRows	InvSubBytes	invShiftRows
MixColumns	AddRoundKey	InvMixColumns
AddRoundKey	InvMixColumns	AddRoundKey
SubBytes	invShiftRows	InvSubBytes
ShiftRows	InvSubBytes	invShiftRows
AddRoundKey	AddRoundKey	AddRoundKey

Очевидно, что результат преобразования при переходе от исходной к обратной последовательности выполнения операций в указанных парах не изменится.

Видно, что процедура шифрования и второй вариант процедуры дешифрования совпадают с точностью до порядка использования раундовых ключей (при выполнении операций AddRoundKey), таблиц замен (при выполнении операций SubBytes и InvSubBytes) и матриц преобразования (при выполнении операций MixColumns и invMixColumns). Данный результат легко обобщить и на любое другое число раундов.

4.3 Содержание отчета

Отчет должен включать в себя следующие пункты:

1) Исходный текст, для шифрования.

2) Зашифрованный текст по алгоритму AES и результат дешифрования.

3) Результаты исследования лавинного эффекта, выполненные в

соответствие со всеми требованиями, изложенными в пункте 4.2.1.

4) Вычисленные значения критериев оценивающих рассеивающие свойства алгоритмов.

5) Выводы о проделанной работе.

4.4 Контрольные вопросы

1. Что такое симметричное шифрование?
2. Что представляет собой стандарт AES (длина ключа, размер входного блока, количество раундов)?
3. Какой алгоритм выбран в качестве стандарта AES?
4. Что собой представляет архитектура данного стандарта?
5. Из чего состоит один раунд?
6. Что быстрее шифрование или дешифрование? Почему?
7. Какие режимы шифрования применяются в стандарте AES?
8. Какие режимы быстрее при дешифровании? Почему?
9. Какие режимы лучше восстанавливают шифрованную информацию при ошибке в одном символе? Почему?
10. С какой целью используется синхропосылка или вектор инициализации?

Список литературы

1. Харин Ю.С., Беник В.И, Матвеев Г.В., Агиевич С.В. Математические и компьютерные основы криптологии: Учеб. пособие. – Минск: Новое знание, 2003.
2. Рябко Б.Я., Фионов А.Н. Криптографические методы защиты информации: Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2005.
3. Баричев С.Г, Гончаров В.В., Серов Р.Е. Основы современной криптографии: Учебный курс. – 2-е изд. – М.: Горячая линия – Телеком, 2002.
4. Осипян В.О, Осипян К.В. Криптография в задачах и упражнениях. – М.: Гелиос АРВ, 2004.
5. Столлингс В. Криптография и защита сетей: принципы и практика: Пер. с англ. / В. Столлингс. – 2-е изд. – М. : Издательский дом «Вильямс», 2001.
6. Eastlake, D.E. US Secure Hash Algorithm 1 (SHA1) : RFC 3174 / D.E. Eastlake, P.E. Jones. – Mil-ford: Motorola; Research Triangle Park : Cisco Systems, Inc., 2001. – 22 p.
7. Eastlake, D.E. US Secure Hash Algorithms (SHA and HMAC–SHA): RFC 4634 / D.E. Eastlake, T. Hansen. – Milford: Motorola Laboratories; Middletown: AT&T Laboratories, 2006.
8. Secure hash standard: FIPS Publication 180-2. – Springfield: National Technical Information Ser-vice, 2002.
9. Secure hash standard: FIPS Publication 180-3. – Gaithersburg: National Institute of Standards and Technology, 2008.
10. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования ГОСТ 28147-89. – М.: Издательство стандартов, 1989.
11. Харин, Ю.С. Математические и компьютерные основы криптологии: Учебное пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. – Мн.: Новое знание, 2003.
12. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2002.
13. Мао, В. Современная криптография: теория и практика: Пер. с англ. / В. Мао. – М.: Издательский дом «Вильямс», 2005.
14. Харин, Ю.С. Математические и компьютерные основы криптологии: Учебное пособие / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. – Мн.: Новое знание, 2003.

Содержание

1 Лабораторная работа №1. Изучение информационно-статистических характеристик текста и частотного метода криптоанализа симметричных криптосистем 3

2 Лабораторная работа №2. Криптографические хэш-функции 5

3 Лабораторная работа №3. Стандарт симметричного шифрования ГОСТ 28147-89 11

4 Лабораторная работа №4. Исследование алгоритма шифрования информации RIJNDAEL и стандарт AES 15

Список литературы 30

Сводн. план 2014 г. поз 153

Куликов Андрей Александрович
Суйеубаев Олжас Билалович

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Методические указания по выполнению лабораторных работ
для студентов специальности
5В100200 – Системы информационной безопасности

Редактор Н.М. Голева
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать
Тираж 50 экз.
Объем 2,1 уч.- изд. л.

Формат 60х84 1/16.
Бумага типографская №1.
Заказ №__ Цена 1050 тенге.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, Байтурсынова, 126