Лабораторная работа № 1

Гаммирование. Моделирование работы скремблера

Цель работы

Освоить на практике применение режима однократного гаммирования. Исследовать побитное непрерывное шифрование данных. Ознакомиться с шифрованием информации при помощи скремблера.

Гаммирование

Простейшей и в то же время наиболее надёжной из всех схем шифрования является так называемая ***схема однократного использования*** (см. рисунок 1), изобретение, которое чаще всего связывают с именем Г.С. Вернама.

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

С точки зрения теории криптоанализа, метод шифрования случайной однократной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым (далее для краткости будем употреблять термин "***однократное гаммирование***", держа в уме всё сказанное выше). Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение – информация о вскрытом участке гаммы не даёт информации об остальных её частях [3].

Допустим, в тайной деловой переписке используется метод однократного наложения гаммы на открытый текст. "Наложение" гаммы – не что иное, как выполнение операции сложения по модулю 2 (xor) её элементов с элементами открытого текста. Эта операция в языке программирования C++ обозначается знаком ^, а в математике – знаком $\oplus$.

Стандартные операции над битами:

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0.$$

Гаммирование является симметричным алгоритмом. Поскольку двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, шифрование и дешифрование выполняется одной и той же программой.

Режим шифрования однократного гаммирования реализуется следующим образом:

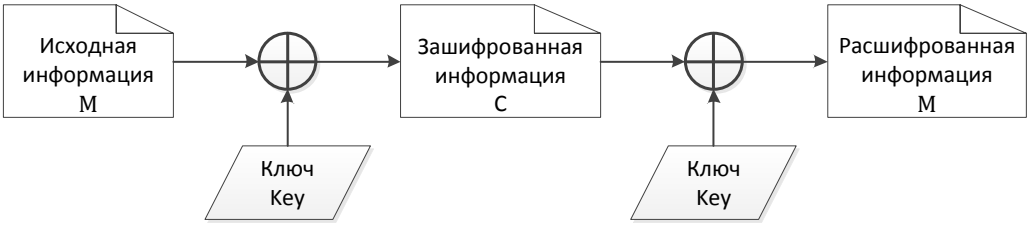

Рисунок 1 – Схема однократного гаммирования Вернама

Задача нахождения шифротекста при известном ключе и открытом тексте состоит в применении следующего правила к каждому символу открытого текста:

$$C_i = P_i \oplus Key_i, \tag{1}$$

где C_i – i-й символ получившегося зашифрованного текста, P_i – i-й символ открытого текста, Key_i – i-й символ ключа, где $i = \overline{1,m}$. Размерности открытого текста и ключа должны совпадать, тогда полученный шифротекст будет такой же длины.

Задача нахождения ключа по известному шифротексту и открытому тексту может быть решена, исходя из (1). Для этого нужно обе части равенства сложить с P_i по модулю 2:

$$C_i \oplus P_i = P_i \oplus Key_i \oplus P_i = Key_i, \tag{2}$$

$$Key_i = C_i \oplus P_i. \tag{3}$$

Таким образом получаются формулы для решения обеих поставленных задач. Т.к. открытый текст представлен в символьном виде, а ключ – в своём шестнадцатеричном представлении, то в соответствие с таблицей ASCII-кодов можно представить ключ в символьном виде. Тогда уже будут возможны операции (1), (3), необходимые для решения поставленных задач.

К. Шенноном было доказано, что если ключ является фрагментом истинно случайной двоичной последовательности с равномерным законом распределением, причём его длина равна длине исходного сообщения, и используется этот ключ только один раз, после чего уничтожается, то такой шифр является ***абсолютно стойким***, даже если криптоаналитик располагает неограниченным ресурсом времени и неограниченным набором вычислительных ресурсов. Действительно, противнику известно только зашифрованное сообщение C, при этом все различные ключевые последовательности Key возможны и равновероятны, а значит, возможны и любые сообщения P, т.е. **криптоалгоритм не даёт никакой информации об открытом тексте** [1].

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.

Рассмотрим пример:

Ключ Центра:

```
05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 57 FF C8 0B B2 70 54
```

Сообщение Центра:

```
Штирлиц – Вы Герой!!
D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C3 E5 F0 EE E9 21 21
```

Зашифрованный текст, находящийся у Мюллера:

```
DD FE FF 8F E5 A6 C1 F2 B9 30 CB D5 02 94 1A 38 E5 5B 51 75
```

Дешифровальщики попробовали ключ:

```
05 0C 17 7F 0E 4E 37 D2 94 10 09 2E 22 55 F4 D3 07 BB BC 54
```

и получили текст:

```
D8 F2 E8 F0 EB E8 F6 20 2D 20 C2 FB 20 C1 EE EB E2 E0 ED 21
Штирлиц - Вы Болван!
```

Пробуя новые ключи, они будут видеть всевозможные тексты заданной длины. ■

Режим шифрования однократного гаммирования одним ключом двух видов открытого текста реализуется следующим образом:

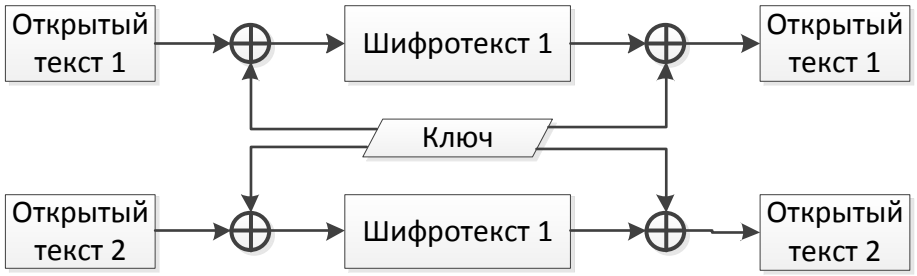

Рисунок 2 – Общая схема шифрования двух различных текстов одним ключом

С помощью формул режима однократного гаммирования получаются шифротексты обеих телеграмм:

$$Y_1 = P_1 \oplus K, \quad Y_2 = P_2 \oplus K \tag{4}$$

Задача нахождения открытого текста по известному шифротексту двух телеграмм, зашифрованных одним ключом, может быть решена, используя (4). Для этого нужно сложить по модулю 2 оба равенства (4). Учитывая такие свойства операции xor, как

$$1 \oplus 1 = 0, \quad 1 \oplus 0 = 1, \tag{5}$$

можно получить

$$Y_1 \oplus Y_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2. \quad (6)$$

Предположим, что одна из телеграмм является "рыбой", т.е. имеет фиксированный формат, в который вписываются значения полей, и злоумышленнику доподлинно известен этот формат. Тогда он получает достаточно много пар $Y_1 \oplus Y_2$ (известен вид обоих шифротекстов) и, предположим, P_1. Тогда с учётом (5) получается:

$$Y_1 \oplus Y_2 \oplus P_1 = P_1 \oplus P_2 \oplus P_1 = P_2. \quad (7)$$

Таким образом, злоумышленник получает возможность определить те символы сообщения P_2, которые находятся на позициях известной "рыбы" сообщения P_1. Догадываясь по логике сообщения P_2, злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения P_2. Затем он может использовать (7), вместо P_1 подставляя ранее определённые символы сообщения P_2. И так далее. Действуя подобным образом, злоумышленник если даже не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

Скремблер

Скремблером называется программная или аппаратная реализация алгоритма, позволяющего шифровать побитно непрерывные потоки информации.

Рассмотрим ***сдвиговый регистр с обратной связью*** (*Linear Feedback Shift Register*, сокращённо *LFSR*) – логическое устройство, схема которого показана на рисунке 3.

Сдвиговый регистр представляет собой последовательность бит. Количество бит определяется длиной сдвигового регистра. Если длина равна n бит, то регистр называется ***n-битным сдвиговым регистром***. Всякий раз, когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо на 1 позицию. Новый крайний левый бит является функцией всех остальных битов регистра. На выходе сдвигового регистра оказывается младший значащий бит [2, 3].

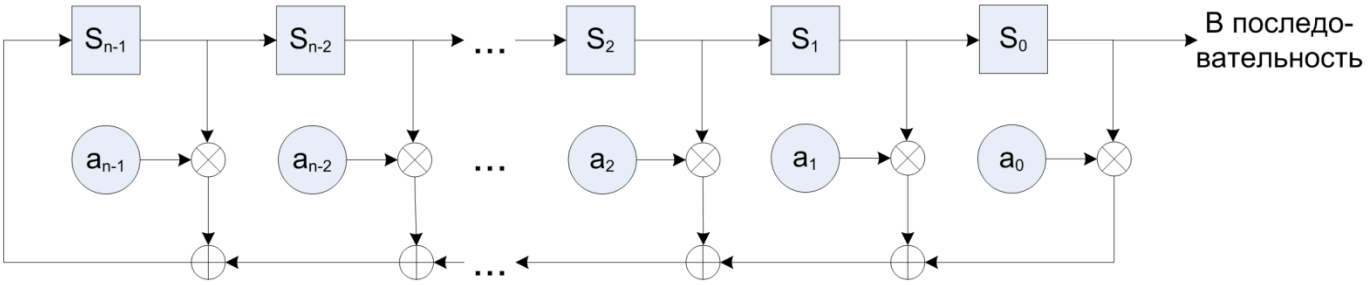

Рисунок 3 – Схема LFSR

LFSR состоит из n ячеек памяти, двоичные состояния которых в моменты времени $t = 0, 1,$ ... характеризуются значениями $S_0(t), S_1(t), \dots, S_{n-1}(t) \in A = \{0, 1\}$. Выходы ячеек памяти связаны не только последовательно друг с другом, но и с сумматорами $\oplus$ в соответствии с коэффициентами передачи $a_0, a_1, \dots, a_{n-1} \in A$: если $a_i = 1$, то значение $S_i(t)$ i-й ячейки передаётся на один из входов i-го сумматора; если же $a_i = 0$, то такая передача отсутствует. Обычно коэффициенты передачи задаются с помощью полинома:

$$f(x) = x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_2 \cdot x^2 + a_1 \cdot x + a_0.$$

Состояние LFSR в текущий момент времени t задаётся двоичным n-вектор-столбцом $S(t) = \left(S_{n-1}(t), \dots, S_0(t)\right)'$.

Содержание ячеек LFSR с течением времени изменяется следующим образом, определяя тем самым динамику состояний LFSR:

$$S_i(t+1) = \begin{cases} S_{i+1}(t), & \text{если } i \in \overline{0, n-2}, \\ \sum_{j=0}^{n-1} a_j S_j(t), & \text{если } i = n-1. \end{cases}$$

Текущие значения нулевой ячейки регистра используются в качестве элементов порождаемой LFSR двоичной псевдослучайной последовательности $s_y = S_0(t)$ (см. рисунок 3).

Данная последовательность извлеченных бит должна обладать тремя свойствами:

1. ***Сбалансированность***: для каждого интервала последовательности количество двоичных единиц должно отличаться от числа двоичных нулей не больше, чем на несколько процентов от их общего количества на интервале.
2. ***Цикличность***: непрерывную последовательность одинаковых двоичных чисел называют ***циклом***. Появление иной двоичной цифры автоматически начинает новый цикл. Длина цикла равна количеству одинаковых цифр в нём. Необходимо, чтобы половина всех «полосок» (подряд идущих идентичных компонентов последовательности) имела длину 1, одна четвертая – длину 2, одна восьмая – длину 3, и т.д.
3. ***Корреляция***: если часть последовательности и её циклично сдвинутая копия поэлементно сравниваются, желательно, чтобы число совпадений отличалось от числа несовпадений не более, чем на несколько процентов от длины последовательности.

При достаточно долгой работе скремблера неизбежно возникает его зацикливание. По выполнении определённого числа тактов в ячейках скремблера создастся комбинация бит, которая в нем уже однажды оказывалась, и с этого момента шифрующая последовательность начнёт циклически повторяться с фиксированным периодом. Данная проблема неустранима по своей природе, т.к. в N разрядах скремблера не может пребывать более 2^N комбинаций бит, и, следовательно, максимум через $2^N - 1$ итерации цикла повтор комбинации непременно произойдёт. Последовательность бит, генерируемая таким скремблером, называется ***последовательностью наибольшей длины***.

Чтобы построить *N*-разрядный скремблер, создающий последовательность наибольшей длины, пользуются примитивными многочленами. ***Примитивный (базовый) многочлен степени*** n по модулю 2 – это неприводимый многочлен, который является делителем $x^{2^n-1} + 1$, но не является делителем $x^d + 1$ для всех d, на которые делится $2^n - 1$. ***Неприводимый многочлен степени*** n нельзя представить в виде умножения никаких других многочленов, кроме него самого и единичного.

Найденный примитивный многочлен степени n записывается в двоичном виде, затем отбрасывается единица, соответствующая самому старшему разряду.

Приведем пример 7-разрядного скремблера, генерирующего последовательность с периодом равным $T = 7$: $x^7 + x^6 + x^2$. Пусть начальное значение состояния будет равно $(79)_{10} = (1001111)_2$. Для этого сдвигового регистра новый бит генерируется по следующей схеме:

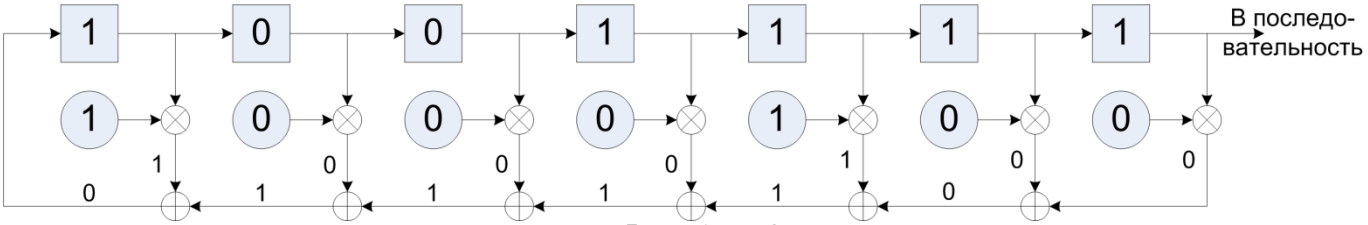

Рисунок 4 – Схема LFSR для многочлена $x^7 + x^6 + x^2$ при начальном состоянии $(1001111)_2$

Код для этого LFSR на языке C выглядит следующим образом:

```
int LFSR ()
{
 static unsigned long ShiftRegister = 79; /* Начальное состояние */
 ShiftRegister =
  ((((ShiftRegister >> 6) ^ (ShiftRegister >> 2)) & 0x01) << 6) | (ShiftRegister >> 1);
 return ShiftRegister & 0x01;
}
```

Последовательность изменения внутреннего состояния скремблера имеет вид:

1001111
0100111
1010011
1101001
1110100
0111010
0011101
1001110
0100111
1010011
...

Рисунок 5 – Изменение состояния скремблера $x^7 + x^6 + x^2$

Как видно из рисунка 5, период генерируемой скремблером последовательности равен 7. Псевдослучайная последовательность, которая используется в качестве гаммы: 1111001011 ...

Задание

I. Реализовать приложение для шифрования, позволяющее выполнять следующие действия:
   1. Шифровать данные в режиме однократного гаммирования:
      1) шифруемый текст должен храниться в файле;
      2) ключ шифрования должен задаваться случайным образом;
      3) зашифрованный текст должен сохраняться в один файл, а использовавшийся при шифровании ключ – в другой;
      4) в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в двоичном, шестнадцатеричном и символьном виде.
   2. Шифровать данные при помощи каждого заданного в варианте скремблера:
      1) шифруемый текст должен храниться в одном файле, начальное значение скремблера – в другом;
      2) зашифрованный текст должен сохраняться в файл;
      3) в процессе шифрования предусмотреть возможность просмотра и изменения начального значения скремблера, шифруемого и зашифрованного текстов в двоичном, шестнадцатеричном и символьном виде.
   3. Проводить исследование генерируемой каждым скремблером последовательности псевдослучайных чисел при заданном начальном ключе:
      1) получать период скремблера;
      2) проверять равномерность последовательности по критерию χ^2;
      3) исследовать последовательность на свойства сбалансированности, цикличности, корреляции.

II. Реализовать приложение для дешифрования, позволяющее выполнять следующие действия:
   1. Дешифровать данные в режиме однократного гаммирования:
      1) зашифрованный текст должен храниться в одном файле, ключ – в другом;
      2) расшифрованный текст должен сохраняться в файл;
      3) в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в двоичном, шестнадцатеричном и символьном виде.
   2. Расшифровать данные при помощи каждого заданного в варианте скремблера:
      1) зашифрованный текст должен храниться в одном файле, начальное значение скремблера – в другом;
      2) зашифрованный текст должен сохраняться в файл;
      3) в процессе дешифрования предусмотреть возможность просмотра и изменения начального значения скремблера, зашифрованного и расшифрованного текстов в двоичном, шестнадцатеричном и символьном виде.

III. С помощью реализованных приложений выполнить следующие задания:
   1. Протестировать правильность работы разработанных приложений.
   2. Определить ключ, с помощью которого зашифрованный текст может быть преобразован в некоторый осмысленный фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.
   3. Определить и выразить аналитически, каким образом, имея зашифрованные тексты двух телеграмм, злоумышленник может получить обе телеграммы, не зная ключа и не стремясь его определить. Привести пример.
   4. Исследовать генерируемые каждым скремблером последовательности псевдослучайных чисел при различных начальных значениях скремблера.
   5. Сделать выводы о проделанной работе.

Дополнительные критерии оценивания качества работы

1. Наглядность приложений:
   1 – приложения позволяют просматривать и изменять ключи, шифруемый и зашифрованный тексты во всех предусмотренных заданием представлениях;
   0 – приложения позволяют просматривать ключи, шифруемый и зашифрованный тексты только в каком-то одном представлении;
   л.р. не принимается – иначе.

Варианты

Вариант	**Скремблеры**	
1	$x^8 + x^7 + x^6 + x^3 + x^2 + 1,$	$x^8 + x^5 + x^3 + x^2 + 1$
2	$x^9 + x^3 + 1,$	$x^9 + x^4 + 1$
3	$x^{10} + x^5 + x^4 + x^2 + 1,$	$x^{10} + x^7 + 1$
4	$x^5 + x^4 + x^2 + 1,$	$x^5 + x^2 + 1$
5	$x^{11} + x^5 + x^2 + 1,$	$x^{11} + x^2 + 1$
6	$x^7 + x^5 + x^2 + 1,$	$x^7 + x + 1$
7	$x^{12} + x^7 + x^3 + x + 1,$	$x^{12} + x^6 + x^4 + x + 1$
8	$x^8 + x^6 + x^2 + 1,$	$x^8 + x^4 + x^3 + x^2 + 1$
9	$x^{11} + x^3 + x^2 + 1,$	$x^{11} + x^{10} + x^9 + x^2 + 1$
10	$x^6 + x^5 + x + 1,$	$x^6 + x + 1$

Вопросы для защиты

1. Преимущества и недостатки однократного гаммирования.
2. Почему размерность открытого текста должна совпадать с ключом?
3. Как по открытому тексту и шифротексту получить ключ?
4. Необходимые и достаточные условия абсолютной стойкости шифра?
5. Как, зная текст одного из сообщений (P_1 или P_2), определить другое, не пытаясь определить ключ?
6. Преимущества и недостатки использования скремблера.
7. Свойства, которыми должна обладать псевдослучайная последовательность, генерируемая скремблером.

Список литературы

1. Столлингс, В. Криптография и защита сетей: принципы и практика : Пер. с англ. / В. Столлингс. – 2-е изд. – М. : Издательский дом "Вильямс", 2001. – 672 с.
2. 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. – 382 с.
3. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М. : Триумф, 2002. – 816 с.