МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждениевысшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических работ
по дисциплине «Информационная безопасность»
Для студентов направления подготовки 09.03.02 Информационные системы и технологии, направленность (профиль) Информационные системы и технологии в бизнесе

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Методические указания предназначены для студентов очной и заочной формы обучения направления 09.03.02 Информационные системы и технологии и других технических специальностей. Они содержат основы теории, порядок проведения лабораторных работ и обработки экспериментальных данных, перечень контрольных вопросов для самоподготовки и список рекомендуемой литературы. Работы подобраны и расположены в соответствии с методикой изучения дисциплины «Информационная безопасность». Объем и последовательность выполнения работ определяются преподавателем в зависимости от количества часов, предусмотренных учебным планом дисциплины, как для очной, так и для заочной форм обучения.

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников по направлению 09.03.02 Информационные системы и технологии.

Содержание

Лабораторная работа № 1

Разработка программного средства для реализации алгоритма интерактивного блочного шифрования и дешифрования

Цель работы: изучить принципы построения и функционирования генераторов псевдослучайных последовательностей для криптографических приложений.

Программа работы

1. Изучить принципы построения и функционирования генераторов псевдослучайных последовательностей для криптографических приложений, основные статистические тесты на случайность данных последовательностей.

2. Разработка программы для реализации заданного генератора псевдослучайной последовательности и оценки ее статистических характеристик.

3. Составить и защитить отчет по результатам работы.

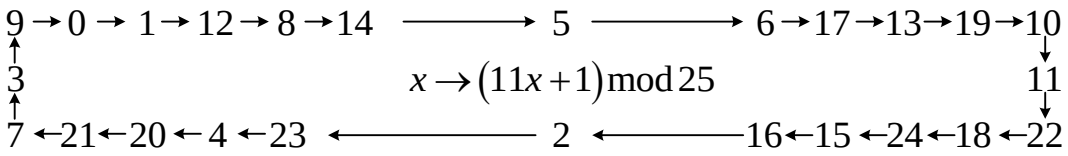
Краткие сведения из теории

Линейные конгруэнтные генераторы (ЛКГ) являются генераторами следующей структуры

$$x_{n+1} = \left(a \cdot x_n + b\right) \bmod m, \qquad (1)$$

где x_{n+1} – (n + 1)-ый член последовательности, x_n – предыдущий член последовательности; константы: a – множитель, b – инкремент, m – модуль.

Период данного генератора не может быть больше m. Если a, b и m выбраны правильно (b обратимо по модулю m; для любого простого p, делящего m, выполняется $a \equiv 1 \left(\bmod p\right)$; если 4 делит m, то $a \equiv 1 \left(\bmod 4\right)$), то генератор будет генератором с максимальным периодом равным m. Например, цикл максимальной длины для ЛКГ по модулю 25:



Преобразование $x \to (3x+1) \bmod 16$ не является циклическим:

Генератор (1) является одношаговым генератором, т.е. имеющий вид $x_{n+1} = f(x_n)$. Данный генератор нельзя использовать в криптографических приложениях, поскольку он предсказуем. Предсказуемыми являются все полиномиальные генераторы, в том числе квадратичные

$$x_{n+1} = \left(ax_n^2 + bx_n + c\right) \bmod m$$

и кубические генераторы

$$x_{n+1} = \left(ax_n^3 + bx_n^2 + cx_n + d\right) \bmod m .$$

ЛКГ сохраняют свою полезность для некриптографических приложений (моделирование, используются в большинстве эмпирических тестов).

Сдвиговые регистры с линейной обратной связью

Сдвиговый регистр с обратной связью состоит из двух частей: сдвигового регистра и функции обратной связи (рисунок 1.1). Сдвиговый регистр представляет собой последовательность битов. Количество битов определяется длиной сдвигового регистра. Если длина равна *n* битам, то регистр называется *n*-битовым сдвиговым регистром.

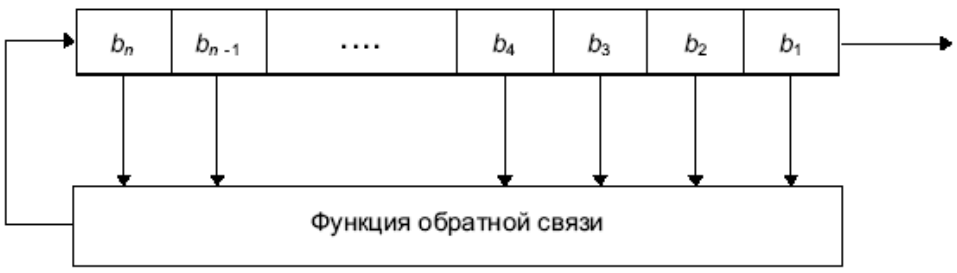


Рисунок 1.1 – Сдвиговый регистр с обратной связью

Всякий раз, когда необходимо 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 один, обычно младший значащий, бит. Периодом сдвигового регистра называется длина получаемой последовательности до начала ее повторения.

Простейшим видом сдвигового регистра с обратной связью является линейный сдвиговый регистр с обратной связью (linear feedback shift register, LFSR). Обратная связь представляет собой функцию XOR некоторых битов регистра, перечень которых называется отводной последовательностью (рисунок 1.2). LFSR чаще других сдвиговых регистров используется в криптографии.

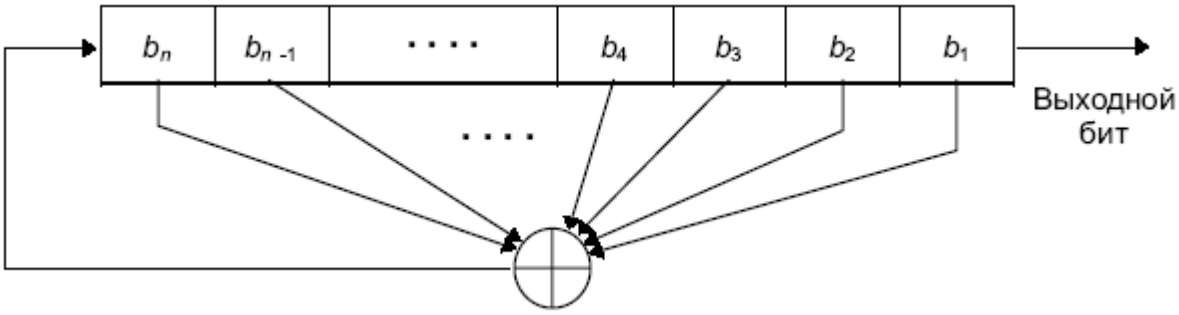


Рисунок 1.2 – Сдвиговый регистр с линейной обратной связью

Например, 4-битовый LFSR с отводом от первого и четвертого битов (рисунок 1.3). Если его проинициализировать значением 1111, то до повторения регистр будет принимать следующие внутренние состояния: 1111, 0111, 1011, 0101, 1010, 1101, 0110, 0011, 1001, 0100, 0010, 0001, 1000, 1100, 1110. Выходной последовательностью будет строка младших значащих битов: 111101011001000.

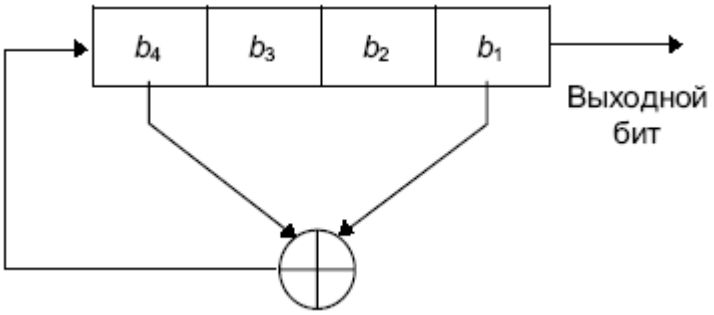


Рисунок 1.3 – 4-битовый LFSR

n-битовый LFSR может находиться в одном из $2^n - 1$ внутренних состояний, т.е. теоретически такой регистр может генерировать последовательность с периодом $2^n - 1$ битов. Только при определенных отводных последовательностях LFSR циклически пройдет через все $2^n - 1$ внутренних состояния (LFSR с максимальным периодом). Для того, чтобы конкретный LFSR имел максимальный период, многочлен, образованный из отводной последовательности и константы 1, должен быть примитивным по модулю 2. Степень многочлена является длиной сдвигового регистра. Примитивный многочлен степени n – это неприводимый многочлен, который является делителем $x^{2^{n-1}} + 1$, но не является делителем $x^d + 1$ для всех d, являющихся делителями $2^n - 1$. Например, примитивный многочлен по модулю 2: $x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1$ или (32, 7, 5, 3, 2, 1, 0), где первое число равно длине LFSR и все числа, за исключением последнего всегда равного 0, задают отводную последовательность, которая отсчитывается от левого края сдвигового регистра. Таким образом, новый бит генерируется с помощью XOR тридцать второго, седьмого, пятого, третьего, второго и первого битов (рисунок 4), циклически проходя до повторения через $2^{32} - 1$ значений.

Код для LFSR(32, 7, 5, 3, 2, 1, 0) на языке C:

```
int LFSR ( ) {
```

```
    static unsigned long ShiftRegister = 1;
    ShiftRegister = ((((ShiftRegister >> 31)
            ^(ShiftRegister >> 6)
            ^(ShiftRegister >> 4)
            ^(ShiftRegister >> 2)
            ^(ShiftRegister >> 1)
            ^ ShiftRegister))
            &0x00000001)
            <<31)
            | (ShiftRegister >> 1);
    return ShiftRegister & 0x00000001;
}
```

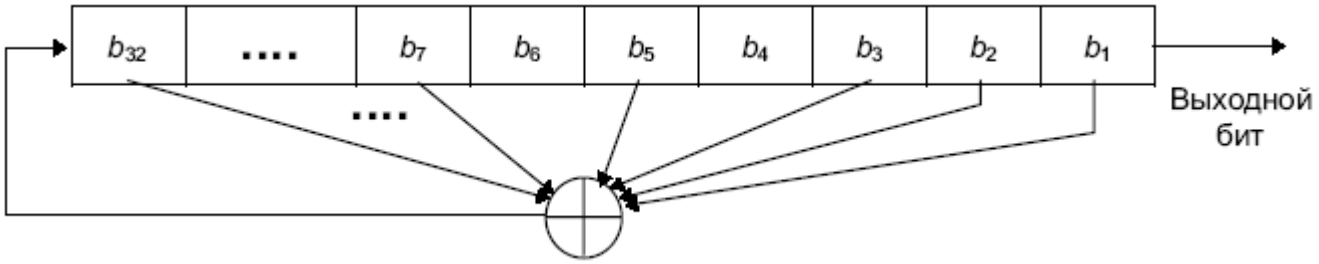


Рисунок 1.4 – 32-битовый LFSR с максимальной длиной

При выборе примитивного многочлена необходимо руководствоваться следующим.

1. Если $p(x)$ примитивен, то примитивен и $x^n p\left(1/x\right)$. Например, если примитивен $(a, b, 0)$, то примитивен и $(a, a - b, 0)$.
2. Быстрее всего программно реализуются примитивные трехчлены (т.е. разряженные многочлены), т.к. для генерации нового бита нужно выполнять XOR только двух битов сдвигового регистра. Однако, разреженность является источником слабости генератора, которой достаточно для вскрытия алгоритма. Для криптографических алгоритмов лучше использовать плотные примитивные многочлены, т.е. у которых много многочленов.

LFSR являются хорошими генераторами псевдослучайных последовательностей, но они обладают нежелательным свойством – последовательные биты линейны, что делает их бесполезными для шифрования. Для LFSR длины *n* внутреннее состояние представляет собой предыдущие *n* выходных битов генератора. Поэтому без знания схемы обратной связи она может быть определена по $2n$ выходным битам генератора. Кроме того, большие случайные числа, генерируемые с использованием идущих подряд битов этой последовательности, сильно коррелированны. Несмотря на это LFSR часто используется для создания алгоритмов шифрования.

Статистические тесты псевдослучайных генераторов

Статистические тесты реализуются на основе точно определенных статистик случайной выборки, которые являются функциями элементов случайной выборки (например, число нулей в двоичной последовательности). Статистики выбираются такими, чтобы они были эффективно вычислимыми, и такими, чтобы они соответствовали нормальному или χ^2-распределению.

Предположим, что статистика *X* случайной последовательности имеет χ^2-ое распределение с *v* степенями свободы. В таблице 2 представлены значения пороговой величины x_α для заданного уровня значимости α и степени свободы *v*, такие что выполняется $P\left(X > x_\alpha\right) = \alpha$. Это означает, что если величина X_S статистики выборки выходной последовательности удовлетворяет неравенству $X_S > x_\alpha$, то тест считается проваленным, в противном случае тест считается пройденным.

Если статистика *X* случайная последовательность, распределенная по нормальному закону $N\left(0,1\right)$, тогда в соответствии с таблицей 1 по за-

данному уровню значимости выбирается пороговая величина x_α, которая удовлетворяет выражению $P(X > x_\alpha) = P(X < -x_\alpha) = \alpha/2$. В этом случае если статистка удовлетворяет неравенству $X_S > x_\alpha$, то тест считается проваленным, в противном случае тест считается пройденным.

Таблица 1 – Процентные точки стандартного нормального распределения

α	0.1	0.05	0.025	0.01	0.005	0.0025	0.001	0.0005
x	1.2816	1.6449	1.9600	2.3263	2.5758	2.8070	3.0902	3.2905

Пусть задана двоичная последовательность $s = s_0, s_1, s_2, ..., s_{n-1}$ длины *n*. Рассмотрим основные пять статистических тестов:

1. Частотный тест (одноразрядный тест).

В соответствии с этим тестом определяется число нулей n_0 и единиц n_1 в *s*. Тогда статистика

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

должна подчиняться χ^2-распределению со степенью свободы 1 и $n \geq 10$.

2. Тест на серии (двухразрядный тест)

В соответствии с тестом определяется числом пар 00, 01, 10, 11 в последовательности *s*. Определим n_0 и n_1 как количество нулей и единиц в *s*, соответственно, а через n_{00}, n_{01}, n_{10}, n_{11} определим число пар 00, 01, 10, 11, соответственно. Кроме того $n_{00} + n_{01} + n_{10} + n_{11} = (n-1)$. Используемая статистка в данном тесте

$$X_2 = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

подчинена χ^2-распределению со степенью свободы 2 и $n \geq 21$.

3. Обобщенный тест

Пусть *m* положительное целое такое, что $\left\lfloor \frac{n}{m} \right\rfloor \geq 5 \cdot \left(2^m\right)$, и пусть $k = \left\lfloor \frac{n}{m} \right\rfloor$.

Разделим последовательность *s* на *k* непересекающихся частей каждая длиной *m*, и пусть n_i будет число последовательностей *i*-ого типа длины *m*, $1 \leq i \leq 2^m$. Данный тест учитывает последовательности длины *m*, которые повторяются приблизительное число раз в *s*. Статистика

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

подчинена χ^2-распределению со степенью свободы $2^m - 1$.

4. Тест на последовательности

Данный тест учитывает последовательности произвольной длины ожидаемые в *s*. Ожидаемое число промежутков (или блоков) длины *i* в случайной последовательности длины *n* равно $e_i = \left(n - i + 3\right) \Big/ 2^{i+2}$. Пусть k будет равно наибольшему целому числу *i* для которой $e_i \geq 5$. Пусть B_i, G_i будет число блоков и промежутков, соответственно, длины *i* в *s* для каждой *i*, $1 \leq i \leq k$. Статистика

$$X_4 = \sum_{i=1}^{k} \frac{\left(B_i - e_i\right)^2}{e_i} + \sum_{i=1}^{k} \frac{\left(G_i - e_i\right)^2}{e_i}$$

подчинена χ^2-распределению со степенью свободы $2k - 2$.

5. Автокорреляционный тест

Целью данного теста является проверка корреляции между последовательностью *s* и не циклически сдвинутой ее самой. Пусть *d* будет целое число из интервала $1 \leq d \leq \left\lfloor n \Big/ 2 \right\rfloor$. Число разрядов в *s* не равных разрядам в

тех же позициях после сдвига на d равно $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$, где $\oplus$ суть оператор XOR.

Таблица 2 – Процентные точки распределения χ^2

v	α					
	0.100	0.050	0.025	0.010	0.005	0.001
1	2.7055	3.8415	5.0239	6.6349	7.8794	10.8276
2	4.6052	5.9915	7.3778	9.2103	10.5966	13.8155
3	6.2514	7.8147	9.3484	11.3449	12.8382	16.2662
4	7.7794	9.4877	11.1433	13.2767	14.8603	18.4668
5	9.2364	11.0705	12.8325	15.0863	16.7496	20.5150
6	10.6446	12.5916	14.4494	16.8119	18.5476	22.4577
7	12.0170	14.0671	16.0128	18.4753	20.2777	24.3219
8	13.3616	15.5073	17.5345	20.0902	21.9550	26.1245
9	14.6837	16.9190	19.0228	21.6660	23.5894	27.8772
10	15.9872	18.3070	20.4832	23.2093	25.1882	29.5883
11	17.2750	19.6751	21.9200	24.7250	26.7568	31.2641
12	18.5493	21.0261	23.3367	26.2170	28.2995	32.9095
13	19.8119	22.3620	24.7356	27.6882	29.8195	34.5282
14	21.0641	23.6848	26.1189	29.1412	31.3193	36.1233
15	22.3071	24.9958	27.4884	30.5779	32.8013	37.6973
16	23.5418	26.2962	28.8454	31.9999	34.2672	39.2524
17	24.7690	27.5871	30.1910	33.4087	35.7185	40.7902
18	25.9894	28.8693	31.5264	34.8053	37.1565	42.3124
19	27.2036	30.1435	32.8523	36.1909	38.5823	43.8202
20	28.4120	31.4104	34.1696	37.5662	39.9968	45.3147
21	29.6151	32.6706	35.4789	38.9322	41.4011	46.7970
22	30.8133	33.9244	36.7807	40.2894	42.7957	48.2679
23	32.0069	35.1725	38.0756	41.6384	44.1813	49.7282
24	33.1962	36.4150	39.3641	42.9798	45.5585	51.1786
25	34.3816	37.6525	40.6465	44.3141	46.9279	52.6197
26	35.5632	38.8851	41.9232	45.6417	48.2899	54.0520
27	36.7412	40.1133	43.1945	46.9629	49.6449	55.4760
28	37.9159	41.3371	44.4608	48.2782	50.9934	56.8923
29	39.0875	42.5570	45.7223	49.5879	52.3356	58.3012
30	40.2560	43.7730	46.9792	50.8922	53.6720	59.7031
31	41.4217	44.9853	48.2319	52.1914	55.0027	61.0983
63	77.7454	82.5287	86.8296	92.0100	95.6493	103.4424
127	147.8048	154.3015	160.0858	166.9874	171.7961	181.9930
255	284.3359	293.2478	301.1250	310.4574	316.9194	330.5197
511	552.3739	564.6961	575.5298	588.2978	597.0978	615.5149
1023	1081.3794	1098.5208	1113.5334	1131.1587	1143.2653	1168.4972

Статистика

$$X_5 = 2\left(A(d) - \frac{n-d}{2} \right) \Big/ \sqrt{n-d}$$

подчинен стандартному нормальному распределению, если $n - d \geq 10$.

Пример. Рассмотрим неслучайную последовательность *s* длиною $n = 160$, полученную повторением следующей последовательности 4 раза: 11100 01100 01000 10100 11101 11100 10010 01001.

1. Частотный тест. $n_0 = 84$, $n_1 = 76$ и величина статистики $X_1 = 0.4$.

2. Тест на серии. $n_{00} = 44$, $n_{01} = 40$, $n_{10} = 40$, $n_{11} = 35$ и величина статистики $X_2 = 0.6252$.

3. Обобщенный тест. Имеется *m* = 3 и *k* = 53. Блоки 000, 001, 010, 011, 100, 101, 110, 111 встречаются 5, 10, 6, 4, 12, 3, 6 и 7 раз, соответственно, и величина статистики $X_3 = 9.6415$.

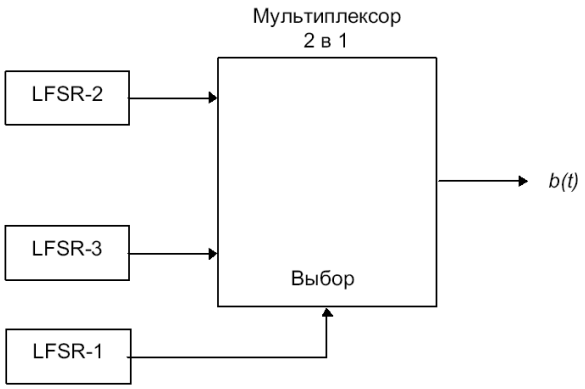
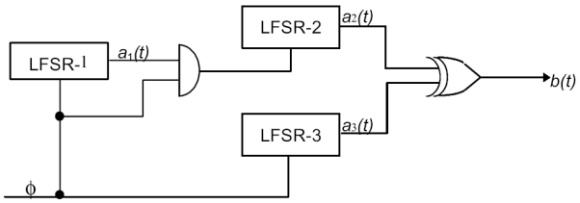
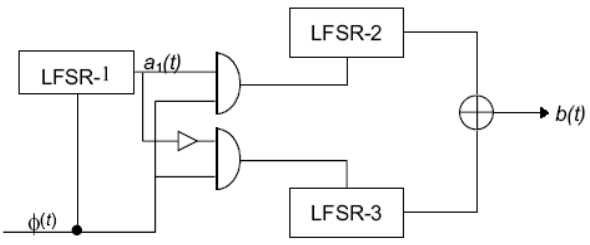
4. Тест на последовательности. $e_1 = 20.25$, $e_2 = 10.0625$, $e_3 = 5$ и *k* = 3. Тогда имеются 25, 4, 5 блоков длины 1, 2, 3, соответственно, и 8, 20, 12 промежутков длины 1, 2, 3, соответственно. Величина статистики $X_4 = 31.7913$.

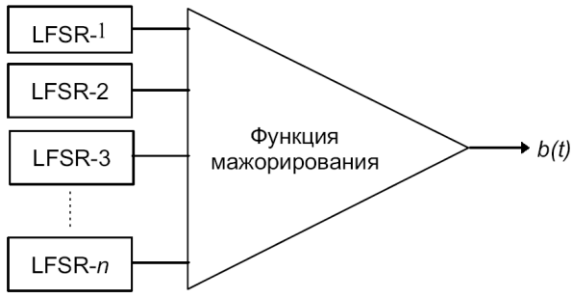
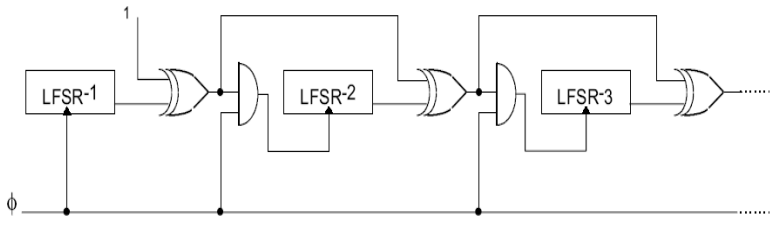
5. Автокорреляционный тест. Если *d* = 8, тогда *A(*8*)* = 100. Величина статистики $X_5 = 3.8933$.

Для уровня значимости $\alpha = 0.05$ пороговые величины для X_1, X_2, X_3, X_4 и X_5 равны 3.8415, 5.9915, 14.0671, 9.4877 и 1.96, соответственно. Как видно заданная последовательность успешно прошла частотный, тест на серии и обобщенный тест, но провалила тест на последовательности и автокорреляционный тест.

Задания на лабораторную работу

1. Написать программу для генератора псевдослучайной последовательности и сгенерировать последовательность равную периоду генератора. Записать полученную последовательность в файл.

Генератор	№ вв	LFSR
Генератор Геффа  $b = (a_1 \wedge a_2) \oplus ((\neg a_1) \wedge a_3)$ Если длины LFSR равны n_1, n_2, n_3, соответственно, то период генератора равен наименьшему общему делителю периодов трех генераторов	1	(16,5,3,2,0) (8,4,3,2,0) (32,7,6,2,0)
	2	(24,4,3,1,0) (20,3,0) (12,6,4,1,0)
	3	(32,7,6,2,0) (28,3,0) (16,5,3,2,0)
Генератор «стоп-пошел» Beth-Piper  	4	(32,7,5,3,2,1,0) (12,6,4,1,0) (28,3,0)
	5	(28,3,0) (16,5,3,2,0) (12,6,4,1,0)
	6	(20,3,0) (28,3,0) (16,5,3,2,0)
Чередующийся генератор «стоп-пошел»  	7	(12,6,4,1,0) (16,5,3,2,0) (24,4,3,1,0)
	8	(8,4,3,2,0) (32,7,6,2,0) (16,5,3,2,0)
	9	(28,3,0) (32,7,5,3,2,1,0) (16,5,3,2,0)

Пороговый генератор  $b = (a_1 \wedge a_2) \oplus (a_1 \wedge a_3) \oplus (a_2 \wedge a_3)$	10	n = 3, (32,7,6,2,0) (28,3,0) (16,5,3,2,0)
	11	n = 5, (32,7,5,3,2,1,0) (16,5,3,2,0) (24,4,3,1,0) (8,4,3,2,0) (20,3,0)
	12	n = 5, (16,5,3,2,0) (12,6,4,1,0) (32,7,6,2,0) (24,4,3,1,0) (28,3,0)
Каскад Голлманна  	13	K = 3, (20,3,0) (24,4,3,1,0) (28,3,0)
	14	K = 3, (8,4,3,2,0) (32,7,6,2,0) (16,5,3,2,0)
	15	K = 3, (32,7,6,2,0) (28,3,0) (16,5,3,2,0)

2. Написать программу для тестирования заданного в генератора по пяти основным тестам.

Контрольные вопросы

1. Что такое LFSR?

2. Как построить псевдослучайный генератор на основе регистра сдвига?

3. Какие тесты на случайность вам известны?

Лабораторная работа №2

Разработка программного средства для реализации алгоритма шифрования и дешифрования по ГОСТ 28147-89

Цель работы: изучить принципы построения и функционирования генераторов псевдослучайных последовательностей для криптографических приложений, получить навыки разработки программ для работы с длинными числами по модулю простого числа.

Программа работы

1. Изучить принципы построения и функционирования генераторов псевдослучайных последовательностей для криптографических приложений, основные статистические тесты на случайность данных последовательностей.

2. Разработка программы для реализации заданного генератора псевдослучайной последовательности и оценки ее статистических характеристик.

3. Составить и защитить отчет по результатам работы.

Краткие сведения из теории

В данном подходе используется в качестве базиса для криптосистемы некоторая известная и сложная проблема, например, теоретико-числовая: факторизация чисел или нахождение дискретного логарифма. Также как и алгоритмы с открытыми ключами, данные генераторы являются медленными и громоздкими.

RSA генератор

Алгоритм функционирования RSA генератора псевдослучайной последовательности $z_1, z_2, ..., z_l$ длины l:

1. Задаются начальные параметры: генерируется два секретных больших простых числа p и q, вычисляется $n = pq$ и $\phi = (p-1)(q-1)$. Выбирается случайное целое число e из диапазона $1 < e < \phi$, такое что $\gcd(e, \phi) = 1$.

2. Выбирается случайное стартовое целое число x_0 из интервала $[1, n-1]$.

3. Для i от 1 до l выполняется:

   3.1. $x_i \leftarrow x_{i-1}^e \bmod n$.

   3.2. $z_i \leftarrow$ младший значащий бит x_i.

4. Выходная последовательность есть $z_1, z_2, ..., z_l$.

Безопасность RSA генератора опирается на сложность вскрытия RSA. Если n достаточно велико (разрядность 1024 бит), то генератор безопасен.

Модификация RSA генератора (Micali-Schnorr RSA генератор):

---

1. Задаются начальные параметры: генерируется два секретных больших простых числа p и q, вычисляется $n = pq$ и $\phi = (p-1)(q-1)$. Принимается $N = \lfloor \log_2 n \rfloor + 1$. Выбирается случайное целое число e из диапазона $1 < e < \phi$, такое что $\gcd(e, \phi) = 1$ и $80e \le N$. Принимается $k = \left\lfloor N\left(1 - \frac{2}{e}\right) \right\rfloor$ и $r = N - k$.

2. Выбирается случайное стартовое целое число x_0 разрядностью r.

3. Генерируется псевдослучайная последовательность длины $k \cdot l$. Для i от 1 до l выполняется:

   3.1. $y_i \leftarrow x_{i-1}^e \bmod n$.

   3.2. $x_i \leftarrow r$ старших значащих разрядов y_i.

3.3. $z_i \leftarrow k$ младших значащих разрядов y_i.

4. Выходная последовательность есть $z_1 \| z_2 \| ... \| z_l$, где || – оператор конкатенации.

Micali-Schnorr RSA генератор более эффективен чем простой RSA генератор, потому что посредством возведения в степень генерируются сразу $\left\lfloor N\left(1-\frac{2}{e}\right)\right\rfloor$ разрядов выходной последовательности. Например, e = 3 и N = 1024, тогда k = 341. Кроме того, каждое возведение в степень включает только одно возведение в квадрат по модулю r = 683-разрядное число и одно модулярное умножение.

Blum-Micali генератор

Безопасность этого генератора определяется трудностью вычисления дискретных логарифмов. Пусть g и p – простые числа. Ключ x_0 начинает процесс:

$$x_i \leftarrow g_{i-1}^{e} \bmod p.$$

Выходом генератора является 1 если $x_i < {(p-1)}/{2}$ и 0 в противном случае. Если p достаточно велико, чтобы вычисление дискретных логарифмов $\bmod p$ стало физически невозможным, то этот генератор безопасен.

Blum-Blum-Shub генератор псевдослучайных разрядов

Генератор с квадратичным остатком (BBS генератор) основывается на сложности факторизации числа.

Алгоритм BBS генератора:

---

1. Задаются начальные параметры: генерируется два секретных простых числа p и q, каждый из которых конгруэнтен 3 по модулю 4,вычисляется $n = pq$.

2. Выбирается случайное целое число s из интервала $[1, n-1]$ такое что $\gcd(s,n)=1$ и вычисляется $x_0 \leftarrow s^2 \bmod n$.

3. Для *i* от 1 до *l* выполняется:

3.1. $x_i \leftarrow x_{i-1}^2 \bmod n$.

3.2. $z_i \leftarrow$ младший значащий бит x_i.

4. Выходная последовательность есть $z_1, z_2, ..., z_l$.

Генерация каждого псевдослучайного бита z_i включает одно возведение в квадрат по модулю. Если *n* достаточно велико (разрядность 1024 бит), то генератор безопасен.

Задание на лабораторную работу

1. Написать программу для генераторов и провести их тестирование

№ вв	Генератор	№ вв	
1	RSA генератор p = 1019, q = 1021	8	Blum-Micali генератор p = 103483
2	RSA генератор p = 1031, q = 1033	9	Blum-Micali генератор p = 103591
3	RSA генератор p = 1039, q = 1049	10	Blum-Micali генератор p = 103703
4	Micali-Schnorr RSA генератор p = 1051, q = 1061	11	Blum-Micali генератор p = 104399
5	Micali-Schnorr RSA генератор p = 1063, q = 1069	12	Blum-Blum-Shub генератор p = 1103, q = 1109
6	Micali-Schnorr RSA генератор p = 1087, q = 1091	13	Blum-Blum-Shub генератор p = 1117, q = 1123
7	Micali-Schnorr RSA генератор p = 1093, q = 1097	14	Blum-Blum-Shub генератор p = 1129, q = 1151
		15	Blum-Blum-Shub генератор p = 1153, q = 1163

2. Написать программу для тестирования заданного в генератора по пяти основным тестам.

Алгоритм вычисления $a^d \bmod m$

Представим d в двоичной системе счисления $d = \sum_{i=0}^{r} d_i \cdot 2^{r-i}$. Положим $a_0 = a$ и затем для i = 1, …, r вычислим $a_i \equiv a_{i-1}^2 \cdot a^{d_i} \left(\bmod m\right)$. a_r есть искомый вычет $a^d \bmod m$.

Алгоритм модульного умножения

```
F = 0
G = 0
for i from n to 0 do
begin
  F = 2*F + A*b_i
  Q = |F/n|
  F = F – Q*n
end
return (F)
```

Алгоритм модулярной редукции через сокращение разрядности

$$A(j+1) = \sum_{i=0}^{k-1} \left|2^i\right|_p^+ \left\{A(j)\right\}^{[i]}, \text{ для j = 0, 1, 2 …,}$$

где $\{\bullet\}^{[i]}$ – оператор извлечения i-го разряда двоичного представления числа; $\left|\bullet\right|_p^+$ – вычетная функция, возвращающая наименьший неотрицательный вычет данного числа по модулю p.

Контрольные вопросы

1. На чем базируется стойкость генераторов псевдослучайных чисел, исследованных в лабораторной работе?

2. Как реализовать возведение в степень чисел большой разрядности по большому модулю?

3. Сравните результаты тестов генераторов из первой лабораторной работы с тестами второй работы.

Лабораторная работа № 3

Разработка программного средства для реализации алгоритма шифрования и дешифрования криптосистемы RC5

Цель работы: изучить принципы блочного шифрования, получить навыки разработки программ для реализации блочного шифрования и дешифрования.

Программа работы

1. Изучить режимы шифрования блочных шифров и стандарты блочного шифрования.
2. Разработка программы для реализации блочного шифрования и дешифрования.
3. Составить и защитить отчет по результатам работы.

Краткие сведения из теории

Криптографическое преобразование составляет основу любого блочного шифра. Прямое криптографическое преобразование (шифрование) переводит блок открытого текста в блок шифротекста той же длины. Обратное криптографическое преобразование (дешифрование) переводит блок шифротекста в исходный блок открытого текста. Необходимое условие выполнения как прямого, так и обратного криптографического преобразования – наличие секретного ключа. Шифры, в которых прямое и обратное преобразования выполняются над блоками фиксированной длины, называются блочными. Для многих блочных шифров разрядность блока составляет 64 бита. Прямое криптографическое преобразование обладает следующим свойством: различные блоки открытого текста отображаются в различные блоки шифротекста. При обратном преобразовании соответствие сохраняется. Прямое преобразование можно рассматривать как перестановку на множестве сообще-

ний с фиксированным размером блока. Результат перестановки носит секретный характер, что обеспечивается секретным компонентом – ключом.

Принцип итерирования является основным при разработке криптографических преобразований и заключается в многократной, состоящей из нескольких циклов обработке одного блока открытого текста. На каждом цикле данные подвергаются специальному преобразованию при участии вспомогательного ключа, полученного из заданного секретного ключа. Выбор числа циклов определяется требованиями криптостойкости и эффективности реализации блочного шифра. Как правило, чем больше циклов, тем выше криптостойкость и ниже эффективность реализации (больше задержка при шифровании/дешифровании) блочного шифра, и наоборот. Так, например, в случае DES (федеральный криптостандарт США) для того, чтобы все биты шифротекста зависели от всех битов ключа и всех битов открытого текста, необходимо 5 циклов криптографического преобразования. DES с 16 циклами обладает высокой криптостойкостью по отношению к ряду криптоаналитических атак.

Конструкция Фейстеля

Конструкция Фейстеля (H. Feistel), или сеть Фейстеля, представляет собой разновидность итерированного блочного шифра. При шифровании блок открытого текста разбивается на две равные части – правую и левую. Очевидно, что длина блока при этом должна быть четной. На каждом цикле одна из частей подвергается преобразованию при помощи функции f и вспомогательного ключа k_i, полученного из исходного секретного ключа. Результат операции суммируется по модулю 2 (операция XOR) с другой частью. Затем левая и правая части меняются местами. Схема конструкции Фейстеля представлена на рисунке 1. Преобразования

на каждом цикле идентичны, но на последнем не выполняется перестановка. Процедура дешифрования аналогична процедуре шифрования, однако k_i, выбираются в обратном порядке. Конструкция Фейстеля хороша тем, что прямое и обратное криптографические преобразования для такого блочного шифра имеют идентичную структуру.

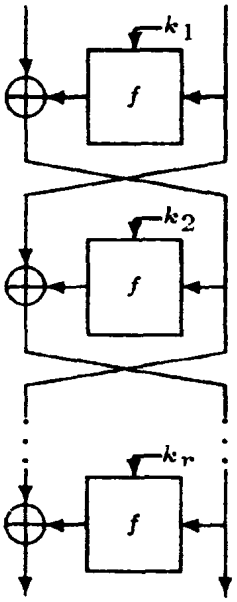


Рисунок 3.1 – Схема конструкции Фейстеля

Конструкция Фейстеля применяется в криптоалгоритмах DES, ГОСТ 28147-89, Lucifer, FEAL, Khufu, Khafre, LOKI, COST, CAST, Blowfish, и др. Блочный шифр, использующий такую конструкцию, является обратимым и гарантирует возможность восстановления входных данных функции f на каждом цикле. Сама функция f не обязательно должна быть обратимой. При задании произвольной функции f не потребуется реализовывать две различные процедуры – одну для шифрования, а другую для дешифрования. Структура сети Фейстеля автоматически позаботится об этом.

Идею конструкции Фейстеля можно объяснить с помощью инволютивного отображения. Так, некоторая функция f является инволюцией,

если $f\left(f\left(x\right)\right)=x$ для всех x. Для такой функции область определения (множество аргументов x) и область значений (множество значений $f\left(x\right)$) совпадают. Например, функция $f\left(x\right)=\neg x$ является инволюцией, так как $f\left(f\left(x\right)\right)=f\left(\neg x\right)=\neg\left(\neg x\right)=x$. Другой пример инволюции: $f\left(x\right)=x\oplus c$, где c — некоторая константа. Действительно, $f\left(f\left(x\right)\right)=f\left(x\oplus c\right)=x\oplus c\oplus c=x$.

Лабораторная работа № 4

Разработка программного средства для реализации алгоритма шифрования и дешифрования BLOWFISH

Режимы шифрования блочных шифров

При использовании блочных шифров применяются различные схемы шифрования, известные под названием рабочих режимов шифрования для блочных шифров. Очевидно, что применение того или иного режима шифрования не должно отрицательно сказываться на эффективности и тем более криптостойкости блочного шифра. Режимы шифрования позволяют реализовать дополнительные, отсутствующие в исходной конструкции блочного шифра функции.

Стандарт режимов шифрования для блочных шифров (применительно к криптоалгоритму DES) опубликован в материалах Национального института стандартов США и ANSI X3.106. Стандарт включает шифрование в следующих режимах: Электронной кодовой книги (Electronic Code Book, ECB), Сцепления блоков шифра (Cipher Block Chaining, CBC), Обратной связи по шифротексту (Cipher Feedback, CFB) и Обратной связи по выходу (Output Feedback, OFB). В режиме ECB (рисунок 2) шифрова-

ние/дешифрование i-го блока открытого текста/шифротекста выполняется независимо: $m_i = D_k(c_i)$, $c_i = E_k(m_i)$, где через E_k и D_k обозначены процедуры шифрования/дешифрования на секретном ключе *k*.

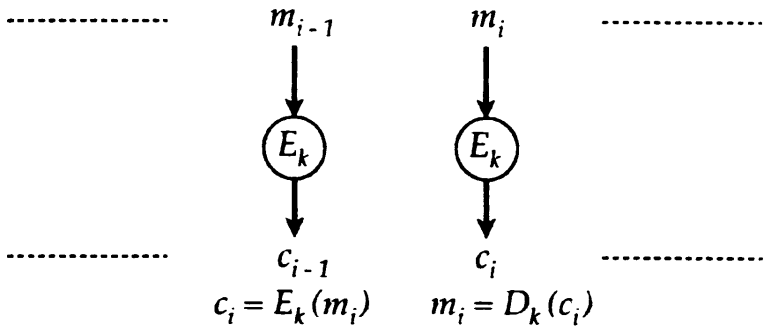


Рисунок 3.2 – Шифрование в режиме ECB

Криптостойкость режима ECB не ниже, чем криптостойкость используемого блочного шифра. Недостаток заключается в том, что фиксированные блоки открытого текста (например, последовательность нулей длины $l = nb$ бит, где b длина блока) будут соответствовать фиксированным блокам шифротекста. Следовательно, открытый текст может быть легко. Скорость обработки блоков в режиме ECB фиксирована и определяется эффективностью реализации блочного шифра. Режим ECB допускает эффективное распараллеливание вычислений. Однако конвейерная обработка блоков в данном режиме невозможна.

В режиме CBC каждый *i*-й блок открытого текста суммируется по модулю 2 (операция XOR) с (i – 1)-м блоком шифротекста и затем шифруется (рисунок 3). Начальное значение задается вектором инициализации.

Криптостойкость режима CBC определяется криптостойкостью используемого блочного шифра. Применение режима CBC позволяет устранить недостаток режима ECB: каждый блок открытого текста «маскируется» блоком шифротекста, полученным на предыдущем этапе. Та-

ким образом, возможность изменения открытого текста при использовании режима CBC весьма ограничена – любые манипуляции с блоками шифротекста, за исключением удаления первого и последнего блоков, будут обнаружены. Скорость обработки в данном режиме не ниже производительности блочного шифра – задержка при выполнении операции XOR пренебрежимо мала. Процедура шифрования в режиме CBC с трудом поддается распараллеливанию, процедуру дешифрования распараллелить значительно проще.

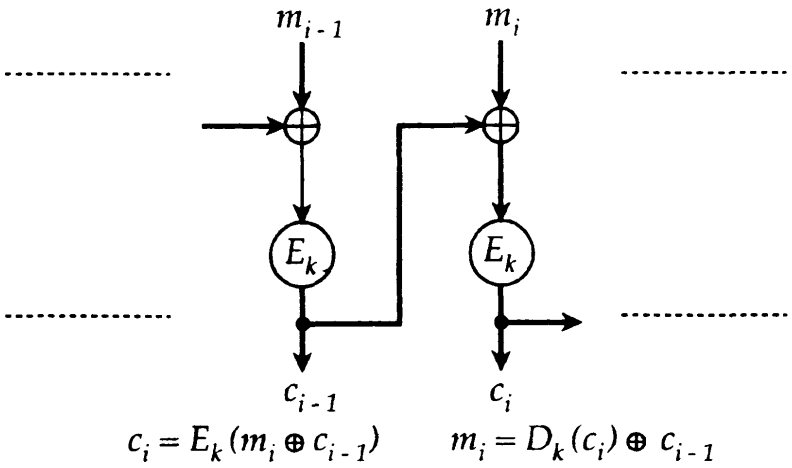


Рисунок 3.3 – Шифрование в режиме CBC

В режиме CFB *i*-й блок шифротекста формируется путем шифрования (*i* – 1)-го блока шифротекста и его суммированием (операция XOR) с *i*-м блоком открытого текста (рисунок 4).

Режим CFB можно задать таким образом, что обратная связь будет захватывать не целый *n*-битный блок, а только *k* бит предыдущего блока, $k < n$. Начальное значение c_0 так же, как в режиме CBC, задается при помощи вектора инициализации.

Криптостойкость CFB определяется криптостойкостью используемого шифра. Фиксированные блоки открытого текста «маскируются» блоками шифротекста. Возможности изменения открытого текста те же,

что и в режиме CBC. Если в режиме CFB с полноблочной обратной связью имеется два идентичных блока шифротекста, результат, например, DES-шифрования на следующем шаге будет тем же. Скорость шифрования CFB-режима с полноблочной обратной связью та же, что и у блочного шифра, причем возможности распараллеливания процедуры шифрования ограничены.

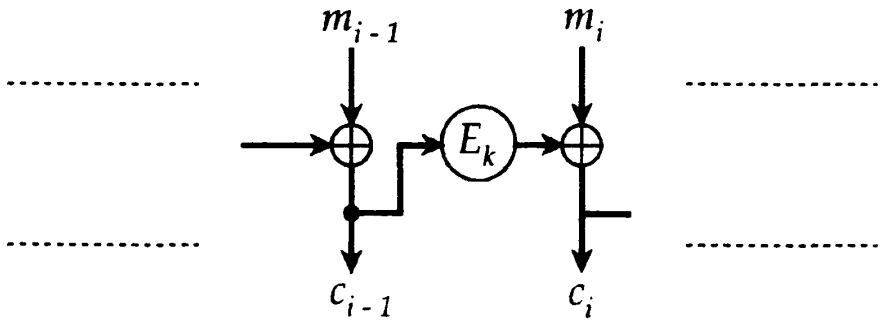


Рисунок 3.4 – Шифрование режима CFB

Режим OFB аналогичен CFB, за исключением того, что суммируемые с открытым текстом биты генерируются независимо от открытого текста и шифротекста. Вектор инициализации s_0 задает начальное значение последовательности блоков s_i, и каждый блок s_i получается путем шифрования предыдущего блока s_{i-1}. Открытый текст шифруется суммированием (операция XOR) *i*-го блока открытого текста с s_i из независимой последовательности блоков (рисунок 5).

Обратная связь по выходу на *k* разрядов не рекомендуется из соображений криптостойкости. Режим OFB имеет следующее преимущество по сравнению с режимом CFB: ошибки, возникающие в результате передачи по каналу с шумом, при дешифровании не «размазываются» по всему шифротексту, а локализуются в пределах одного блока. Однако открытый текст может быть изменен путем определенных манипуляций с

блоками шифротекста. Скорость шифрования в режиме OFB та же, что и у блочного шифра. Несмотря на то, что OFB-шифрование не поддается распараллеливанию, эффективность процедуры может быть повышена за счет предварительной генерации независимой последовательности блоков.

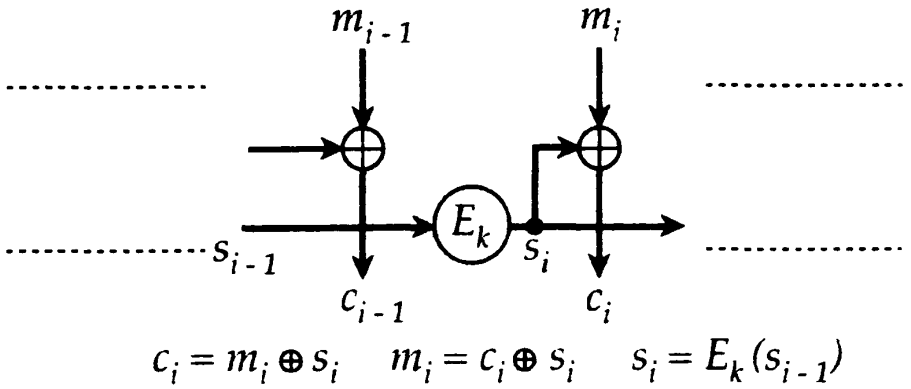


Рисунок 3.5 – Шифрование в режиме OFB

Известные недостатки привели к появлению усовершенствованного варианта шифрования в режиме OFB. Основные изменения касаются метода генерации независимой последовательности блоков: для получения очередного блока предлагается шифровать не s_i, а $s_i + IV \left(\mathrm{mod}\, 2^{64}\right)$, где IV — некоторый вектор инициализации.

Режим шифрования PCBC (Propagating Cipher Block Chaining) применяется в протоколе Kerberos (версия 4) и позволяет обнаруживать ошибки. Данный режим шифрования не является федеральным или международным стандартом. Режим PCBC – вариант режима CBC, обладающий специфическим свойством, в результате дешифрования единичная ошибка распространяется на весь шифротекст (решается обратная задача с точки зрения режима OFB). Данное свойство позволяет с высокой надежностью обнаруживать ошибки, возникающие при передаче сообще-

ний по каналам с шумом. Шифрование в режиме PCBC выполняется по правилу:

$$c_i = E_k\left(m_i \oplus m_{i-1} \oplus c_{i-1}\right),$$

дешифрование:

$$m_i = D_k\left(c_i\right) \oplus c_{i-1} \oplus m_{i-1}$$

где то $m_0 \oplus c_0$ – вектор инициализации.

Лабораторная работа № 5

Разработка программного средства для реализации алгоритма псевдовероятного 512-байтового блочного шифрования

Стандарты блочного шифрования

Федеральный стандарт США – DES, который ANSI называет Алгоритмом шифрования данных DEA (Data Encryption Algorithm), а ISO — DEA-1, за 20 лет стал мировым стандартом. DES представляет собой блочный шифр, шифрующий данные 64-битовыми блоками. С одного конца алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрования используются одинаковые алгоритм и ключ (за исключением небольших различий в использовании ключа). Длина ключа равна 56 битам. Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа. Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент времени.

Криптостойкость полностью определяется ключом. Фундаментальным строительным блоком DES является комбинация подстановок и перестановок. DES состоит из 16 циклов (рисунок 6). В общем виде цикл преобразования представлен на рисунке 7.

Если L_i и R_i – левая и правая половины, полученные в результате i-й итерации, K_i – 48-битный ключ для цикла i, а f – функция, выполняющая все подстановки, перестановки и XOR с ключом, то один цикл преобразования можно представить как

$$\left(L_i, R_i\right) = \left(L_{i-1}, R_{i-1} \oplus f\left(R_{i-1}, K_i\right)\right).$$

Учитывая подстановку $F(\square)$ и перестановку $T(\square)$, цикл преобразования можно представить так, как это сделано на рисунке 8. Из рисунка 8 видно, что каждый цикл DES представляет собой композиционный шифр с двумя последовательными преобразованиями – подстановкой $F(\square)$ и перестановкой $T(\square)$ (за исключением последнего, шестнадцатого цикла, где перестановка опускается).

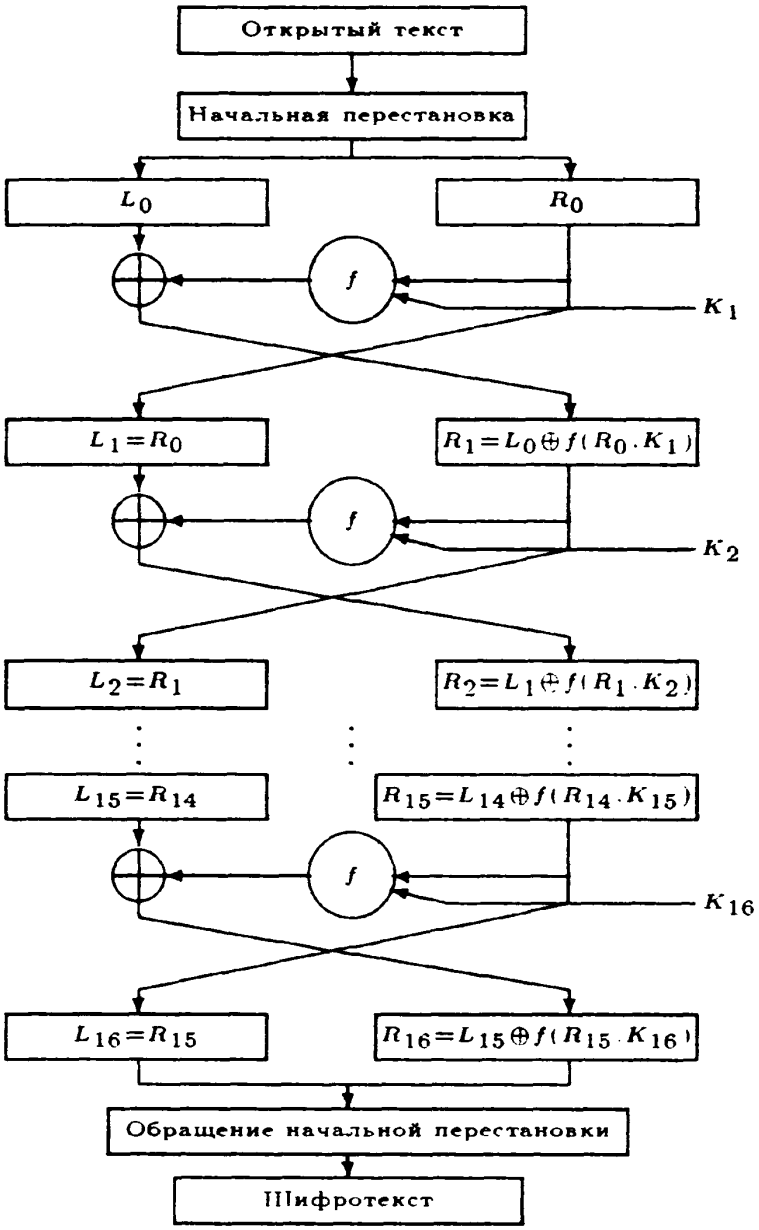


Рисунок 3.6 – Схема DES-преобразования

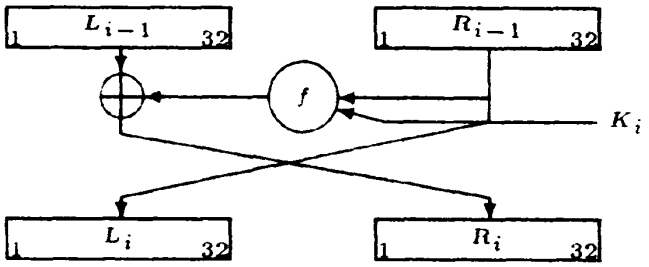


Рисунок 3.7 – Общий цикл DES-преобразования

Таким образом, DES является шифром Фейстеля и сконструирован так, чтобы выполнялось полезное свойство: для шифрования и дешифрования используется один и тот же алгоритм. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке.

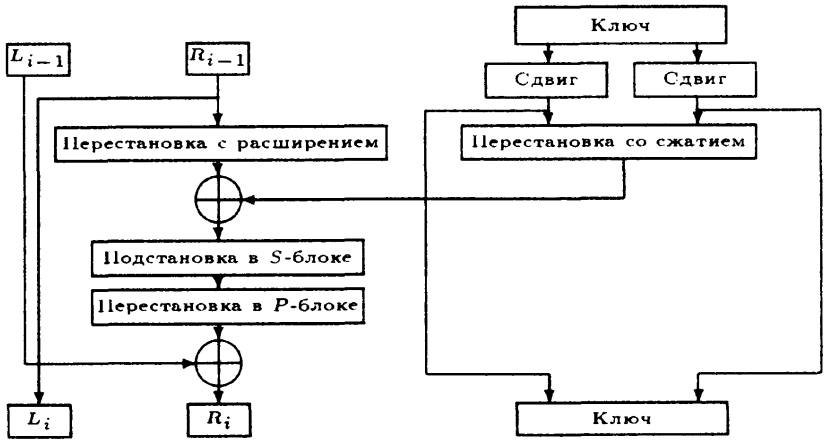


Рисунок 3.8 – Один цикл DES-преобразования

То есть если при шифровании использовались ключи $K_1, K_2, K_3, ..., K_{16}$, то ключами дешифрования будут $K_{16}, K_{15}, K_{14}, ..., K_1$. Алгоритм использует только стандартную арифметику 64-битовых чисел и логические операции, поэтому легко реализуется на аппаратном уровне.

DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 преобразований (функция *f*), в которых данные объединяются с ключом. После шестнадцатого цикла правая и левая половины объединяются, и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной). На каждом цикле (см. рисунок 8) биты ключа сдвигаются, и затем из 56 битов ключа выбираются 48 битов. Правая половина дан-

ных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией *f*.

Затем результат функции *f* объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая становится новой левой половиной. Эти действия повторяются 16 раз, образуя 16 циклов DES.

ГОСТ 28147-89 – это блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками. В криптоалгоритме также используется дополнительный ключ. Для шифрования открытый текст сначала разбивается на левую и правую половины *L* и *R*. На i-м цикле используется под ключ K_i:

$$L_i = R_{i-1},$$

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

Один цикл криптографического преобразования показан на рисунке 3.9.

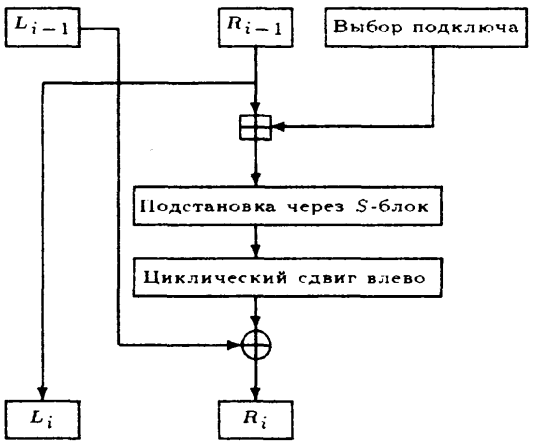


Рисунок 3.9 – Один цикл преобразования ГОСТа 28147-89

Функция f реализована следующим образом. Сначала правая половина и i-й подключ складываются по модулю 2^{32}. Результат разбиваетя на восемь 4-битовых подпоследовательностей, каждая из которых поступает на вход своего S-блока. ГОСТ использует восемь различных S-блоков, первые 4 бита попадают в первый S-блок, вторые 4 бита – во второй S-блок и т.д. Каждый S-блок представляет собой перестановку чисел от 0 до 15. Например, S-блок может выглядеть так: 7,10,2,4,15,9,0,3,6,12,5,13,1,8,11. В этом случае, если на входе S-блока 0, то на выходе 7, если на входе 1, на выходе 10 и т.д. Все восемь S-блоков различны, они фактически являются дополнительным ключевым материалом. Выходы всех восьми S-блоков объединяются в 32-битовое слово, затем все слово циклически сдвигается влево на 11 битов. Наконец, результат объединяется с помощью операции XOR с левой половиной, и получается новая правая половина, а правая половина становится новой левой половиной. Для генерации подключей исходный 256-битный ключ разбивается на восемь 2-битных блоков: $k_1, k_2, k_3, ..., k_8$. На каждом цикле используется свой подключ. Дешифрование выполняется так же, как и шифрование, но инвертируется порядок подключей k_i. Стандарт не определяет способ генерации S-блоков.

Набор S-блоков, указанный в таблице, рекомендуется стандартом ГОСТ Р 34.11-94.

Таблица 3.1 – S-блоки ГОСТа 28147-89

S-блок 1:	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
S-блок 2:	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
S-блок 3:	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
S-блок 4:	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
S-блок 5:	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
S-блок 6:	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
S-блок 7:	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
S-блок 8:	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Главные различия между DES и ГОСТом заключаются в следующем:

- DES использует сложную процедуру для генерации подключей из ключей. В ГОСТе эта процедура очень проста;
- в DES 56-битный ключ, а в ГОСТе — 256-битный. Если добавить секретные перестановки 5-блоков, то полный объем секретной информации ГОСТа составит примерно 610 бит;
- у S-блоков DES 6-битные входы и 4-битные выходы, а у 5-блоков ГОСТа 4-битные входы и выходы. В обоих алгоритмах используется по восемь S-блоков, но размер S-блока ГОСТа равен четверти размера S-блока DES;
- в DES используются нерегулярные перестановки, названные *P*-блоком, а в ГОСТе используется 11-битный циклический сдвиг влево;
- в DES 16 циклов, а в ГОСТе – 32.

Силовая атака на ГОСТ абсолютно бесперспективна. ГОСТ использует 256-битовый ключ, а если учитывать секретные 5-блоки, то длина ключа будет еще больше. ГОСТ, по-видимому, более устойчив к дифференциальному и линейному криптоанализу, чем DES. Хотя случайные S-блоки ГОСТа при некотором выборе не гарантируют высокой криптостойкости по сравнению с фиксированными 5-блоками DES, их

секретность увеличивает устойчивость ГОСТа к дифференциальному и линейному криптоанализу. К тому же эффективность этих криптоаналитических методов зависит от количества циклов преобразования – чем больше циклов, тем труднее криптоанализ. ГОСТ использует в два раза больше циклов, чем DES, что, возможно, приводит к несостоятельности дифференциального и линейного криптоанализа. С точки зрения криптостойкости операция арифметического сложения, используемая в ГОСТе, не хуже, чем операция XOR в DES. Основным различием представляется использование в ГОСТе циклического сдвига вместо перестановки. Перестановка DES увеличивает лавинный эффект. В ГОСТе изменение одного входного бита влияет на один S-блок одного цикла преобразования, который затем влияет на два S-блока ледующеего цикла, затем на три блока следующего цикла и т.д. Потребуется восемь циклов, прежде чем изменение одного входного бита повлияет на каждый бит результата; в DES для этого нужно только пять циклов. Однако ГОСТ состоит из 32 циклов, а DES только из 16.

Задания на лабораторную работу

Написать программу для реализации блочного шифрования файла:

№ вв	Алгоритм блочного шифрования
1,2,3,4	ГОСТ 28147-89
5,6,7,8	BlowFish
9,10,11,12	3-Way
13,14,15	RC5

Контрольные вопросы

1. Что такое симметричное шифрование?

2. В чем особенность блочных шифров?

3. Какова длина ключа блочного шифра?

4. На чем базируется криптостойкость блочного шифра?

5. Какие элементарные операции используются в симметричном шифровании?

Лабораторная работа №6

Разработка программного средства для реализации алгоритма асимметричного шифрования и дешифрования

Цель работы: изучить принципы построения асимметричных криптосистем, получить навыки разработки программ для реализации асимметричного шифрования и дешифрования.

Программа работы

1. Изучить принципы асимметричного шифрования и стандарты асимметричного шифрования.

2. Программная реализация асимметричного шифрования и дешифрования.

3. Составить и защитить отчет по результатам работы.

Краткие сведения из теории

Криптосистема RSA, предложенная в 1977 г. Ривестом (R. Rivest), Шамиром (A. Shamir) и Адлеманом (L. Adleman), предназначена для шифрования и цифровой подписи. Для генерации парных ключей используются два больших случайных простых числа, p и q. В целях максимальной криптостойкости p и q выбираются равной длины. Затем вычисляется произведение: $n = pq$.

Далее случайным образом выбирается ключ шифрования *e*, такой, что *e* и $\phi(n)=(p-1)(q-1)$ являются взаимно простыми числами. Наконец расширенный алгоритм Евклида используется для вычисления ключа дешифрования *d,* такого, что $ed \equiv 1 \bmod \phi(n)$. Другими словами,

$$d = e^{-1} \bmod \phi(n).$$

Заметим, что *d* и *n* – также взаимно простые числа. Числа *e* и *n* – открытый ключ, а *d* – секретный. Два простых числа *p* и *q* хранятся в секрете. Для шифрования сообщения *m* необходимо выполнить его разбивку на блоки, каждый из которых меньше *n* (для двоичных данных выбирается самая большая степень числа 2, меньшая *n).* То есть если *p* и *q* – 100-разрядные простые числа, то *n* будет содержать около 200 разрядов и каждый блок сообщения m_i должен иметь такое же число разрядов. Если нужно зашифровать фиксированное число блоков, их можно дополнить несколькими нулями слева, чтобы гарантировать, что блоки всегда будут меньше *n*. Зашифрованное сообщение *c* будет состоять из блоков *a* той же самой длины. Шифрование сводится к вычислению $c_i = m_i^e \bmod n$. При дешифровании для каждого зашифрованного блока c_i вычисляется $m_i = c_i^d \bmod n$. Все вычисления выполняются по mod *n.* Сообщение может быть зашифровано с помощью *d,* а дешифровано с помощью e, возможен любой выбор.

Численный пример

p = 47 и *q* = 71, то *n* = *pq* = 3337. Ключ *e* не должен иметь общих множителей с $\phi(n) = 46 \times 72 = 3220$. Выбираем (случайно) *e* равным 79. Тогда $d = 79^{-1} \bmod 3220 = 1019$. Публикуем *e* и *n*, сохранив в секрете *d*. Для шифрования сообщения $m = 6882326879666683$ сначала разделим его на блоки. Для выбранных параметров ограничимся блоками по три деся-

тичных разряда. Сообщение разбивается на шесть блоков m_i: $m_1 = 688$, $m_2 = 232$, $m_3 = 687$, $m_4 = 966$, $m_5 = 668$, $m_6 = 003$. Первый блок шифруется как $688^{79} \bmod 3337 = 1570 = c_1$. Выполняя те же операции для последующих блоков, создадим шифротекст сообщения:

$$c = 157027562091227624231 58.$$

Для дешифрования нужно выполнить возведение в степень, используя ключ дешифрования 1019:

$$1570^{1019} \bmod 3337 = 688 = m_1.$$

Аналогично восстанавливается оставшаяся часть сообщения.

Предполагается, что криптостойкость RSA зависит от проблемы разложения на множители больших чисел. Однако никогда не было доказано математически, что нужно разложить *n* на множители, чтобы восстановить *m* по *c* и *e*. Не исключено, что может быть открыт совсем иной способ криптоанализа RSA. Однако, если этот новый способ позволит криптоаналитику получить *d*, он также может быть использован для разложения на множители больших чисел. Также можно атаковать RSA, угадав значение (p – l)(q – 1). Однако этот метод не проще разложения *n* на множители. При использовании RSA раскрытие даже нескольких битов информации по шифротексту не легче, чем дешифрование всего сообщения. Самой очевидной атакой на RSA является разложение *n* на множители. Любой противник сможет получить открытый ключ *e* и модуль *n*. Чтобы найти ключ дешифрования *d*, противник должен разложить *n* на множители. Криптоаналитик может перебирать все возможные *d*, пока не подберет правильное значение. Но подобная силовая атака даже менее эффективна, чем попытка разложения *n* на множители.

Некоторые атаки используют уязвимость криптографического протокола. Важно понимать, что само по себе использование RSA не обеспечивает требуемого уровня безопасности системы. Рассмотрим несколько сценариев.

Сценарий 1

Злоумышленнику удалось перехватить сообщение *c*, зашифрованное с помощью открытого RSA-ключа абонента А. Он хочет прочитать сообщение. Для раскрытия $m = c^d$ он сначала выбирает первое случайное число r, меньшее *n,* и затем, воспользовавшись открытым ключом Алисы e, вычисляет $x = r^e \bmod n$, $y = xc \bmod n$, $t = r^{-1} \bmod n$. Если $x = r^e \bmod n$, то $r = x^d \bmod n$.

Далее злоумышленник вынуждает абонента А подписать сообщение *y.* Таким образом, процедура вычисления подписи на секретном ключе соответствует процедуре дешифрования сообщения *y.* Абонент должен подписать сообщение, а не значение хэш-функции. Такой обман вполне реален, так как абонент А никогда раньше не видел *y.* Абонент А посылает злоумышленнику $u = y^d \bmod n$. Теперь злоумышленник раскрывает *m,* вычисляя

$$tu \bmod n = r^{-1} y^d \bmod n = r^{-1} x^d c^d \bmod n = m.$$

Сценарий 2

Если абонент А хочет заверить документ, он посылает его нотариусу. Нотариус подписывает его цифровой подписью и отправляет обратно. При этом хэш-функции не используются, нотариус шифрует все сообщение на своем секретном ключе. Злоумышленник хочет, чтобы нотариус подписал такое сообщение, которое в обычном случае тот никогда не подпишет. Это может быть фальшивая временная метка, либо автором этого сообщения может являться другое лицо. Какой бы ни

была причина, нотариус никогда не подпишет это сообщение, если у него будет возможность выбора. Назовем это сообщение m'. Сначала злоумышленник выбирает произвольное значение x и вычисляет $y = x^e \bmod n$. Параметр e он может получить без труда – это открытый ключ нотариуса, и должен быть опубликован для проверки подписи последнего. Теперь злоумышленник вычисляет $m = ym' \bmod n$ и посылает m нотариусу на подпись. Нотариус возвращает $m^d \bmod n$. Далее злоумышленник вычисляет $\left(m^d \bmod n\right)x^{-1} \bmod n$, которое равно $m'^d \bmod n$ и является подписью m'.

Сценарий 3

Злоумышленник хочет, чтобы абонент подписал некое сообщение m_3. Для этого он создает два сообщения, m_1 и m_2, такие, что $m_3 = m_1 m_2 \bmod n$. Если злоумышленник заставит абонента подписать m_1 и m_2, то сможет вычислить подпись для m_3:

$$m_3^d = \left(m_1^d \bmod n\right)\left(m_2^d \bmod n\right) \bmod n.$$

Вывод – никогда нельзя использовать RSA для подписи случайных документов. Применение хэш-функций в технологии RSA-подписи строго обязательно.

Сценарий 4

При реализации RSA можно попробовать раздать всем абонентам криптосети одинаковый модуль n, но каждому – свои значения показателей степени e и d. При этом наиболее очевидная проблема заключается в том, что если одно и то же сообщение когда-нибудь шифровалось разными показателями степени (при фиксированном модуле) и эти два показателя – взаимно-простые числа (как обычно и бывает), то открытый текст может быть раскрыт даже при неизвестных ключах дешиф-

рования. Пусть заданы: m – открытый текст, e_1 и e_2 – два ключа шифрования, n – общий модуль. Шифротекстами сообщения являются:

$$c_1 = m^{e_1} \bmod n,\ c_2 = m^{e_2} \bmod n.$$

Криптоаналитик знает n, e_1, e_2, c_1 и c_2. Так как e_1 и e_2 – взаимно-простые числа, то, воспользовавшись расширенным алгоритмом Евклида, можно найти такие числа r и s, что

$$re_1 + se_2 = 1.$$

Полагая r отрицательным (или r, или s должно быть отрицательным), можно снова воспользоваться расширенным алгоритмом Евклида для вычисления c_1^{-1}. Тогда

$$\left(c_1^{-1}\right)^{-r} c_2^s = m \bmod n.$$

Сценарий 5

Известно, что криптосистема RSA обладает низкой криптостойкостью при зашифрованном на малом e коротком сообщении. Действительно, при $c = m^e < n$ открытый текст m может быть восстановлен по шифротексту c при помощи процедуры извлечения корня. Фактически подобная атака возможна и тогда, когда в процессе возведения в степень выполнялось некоторое количество приведений по модулю. При $c > n$ трудоемкость такой атаки ниже трудоемкости исчерпывающего перебора для m. Однако меры противодействия также очевидны, – либо открытый ключ e должен быть достаточно большим, либо открытый текст не должен быть коротким. Выбор малого e обусловлен соображениями вычислительной эффективности шифрования и проверки подписи. Таким образом, разумный подход заключается в искусственном наращивании коротких открытых текстов («набивки»). При этом необходимо сле-

дить за тем, чтобы удлиненный открытый текст при числовом отображении не превращался в набор множителей некоторого известного числа P, например $P = 2^l$, что происходит при дополнении открытого текста последовательностью нулей справа (со стороны младших разрядов).

На основании перечисленных атак можно сформулировать следующие ограничения при использовании RSA :

- знание одной пары показателей шифрования/дешифрования для данного модуля позволяет злоумышленнику разложить модуль на множители;

- знание одной пары показателей шифрования/дешифрования для данного модуля позволяет злоумышленнику вычислить другие пары показателей, не раскладывая модуль на множители;

- в криптографических протоколах с использованием RSA общий модуль использоваться не должен;

- для предотвращения раскрытия малого показателя шифрования сообщения должны быть дополнены («набиты») случайными значениями;

- показатель дешифрования должен быть большим.

Недостаточно использовать криптостойкий алгоритм, безопасной должна быть вся криптосистема, включая криптографический протокол. Слабое место любого из трех этих компонентов сделает небезопасной всю систему.

Задания на лабораторную работу

№ варианта	Задание № 1 Написать программу для реализации RSA шифрования/ дешифрования	Задание № 2 Произвести моделирование атаки на RSA криптосистему
1	607 613	Сценарий 1

2	617	619	Сценарий 2
3	631	641	Сценарий 3
4	643	647	Сценарий 4
5	653	659	Сценарий 5
6	661	673	Сценарий 1
7	677	683	Сценарий 2
8	691	701	Сценарий 3
9	709	719	Сценарий 4
10	727	733	Сценарий 5
11	739	743	Сценарий 1
12	751	757	Сценарий 2
13	761	769	Сценарий 3
14	773	787	Сценарий 4
15	797	809	Сценарий 5

Контрольные вопросы

1. В чем особенность асимметричных систем шифрования?
2. На чем базируется криптостойкость RSA?
3. Как увеличить производительность системы шифрования RSA?
4. Какие атаки на систему RSA вам известны?
5. Как противодействовать атакам на систему RSA?

Лабораторная работа №7

Разработка программного средства для реализации алгоритма шифрования на основе управляемых подстановок

Цель работы: изучить принципы построения асимметричных криптосистем, получить навыки разработки программ для реализации асимметричного шифрования и дешифрования.

Программа работы:

1. Изучить принципы асимметричного шифрования и стандарты асимметричного шифрования.

2. Программная реализация асимметричного шифрования и дешифрования.

3. Составить и защитить отчет по результатам работы.

Краткие сведения из теории

Криптосистему, предложенную ЭльГамалем (T. ElGamal) в 1985 г., можно использовать как для цифровых подписей, так и для шифрования. Криптостойкость определяется трудоемкостью вычисления дискретного логарифма над конечным полем.

Для генерации пары ключей сначала выбираются простое число p и два случайных числа, g и x; оба этих числа должны быть меньше p. Затем вычисляется

$$y = g^x \bmod p .$$

Открытым ключом являются y, g и p. И g, и p можно сделать общими для группы пользователей. Секретным ключом является x.

Вычисление и проверка подписи.

Чтобы подписать сообщение M, сначала выбирается случайное число k, взаимно простое с $(p-1)$. Затем вычисляется $a = g^k \bmod p$, и с

помощью расширенного алгоритма Евклида из уравнения $M = (xa + kb) \bmod (p-1)$ находится b. Подписью является пара чисел: a и b. Случайное значение k должно храниться в секрете. Для проверки подписи необходимо убедиться, что

$$y^a x^b \bmod p = g^M \bmod p.$$

Каждая новая подпись требует нового значения k, и это значение должно выбираться случайным образом. Если злоумышленник раскроет k, используемое абонентом, он сможет раскрыть секретный ключ x. Если злоумышленник сможет получить два сообщения, подписанные при помощи одного и того же k, он сможет раскрыть x, даже не зная k.

Численный пример

Выберем $p = 11$ и $g = 2$. Пусть секретный ключ $x = 8$. Вычислим

$$y = g^x \bmod p = 28 \bmod 11 = 3.$$

Открытым ключом являются $y = 3$, $g = 2$ и $p = 11$. Чтобы подписать $M = 5$, сначала выберем случайное число $k = 9$. Убедимся, что gcd(9,10) = 1. Далее вычислим $a = g^x \bmod p = 29 \bmod 11 = 6$. Затем с помощью расширенного алгоритма Евклида найдем b из уравнения $M = (xa + kb) \bmod (p-1)$:

$$5 = (8 \cdot 6 + 9 \cdot b) \bmod 10.$$

Решение: $b = 3$, а подпись представляет собой пару: $a = 6$ и $b = 3$. Для проверки подписи убедимся, что $y^a x^b \bmod p = g^M \bmod p$:

$$3663 \bmod 11 = 25 \bmod 11.$$

Шифрование/дешифрование.

Для шифрования сообщения M сначала выбирается случайное число k, взаимно-простое с $(p-1)$. Затем вычисляются $a = g^k \bmod p$, $b = y^k M \bmod p$.

Пара (a, *b)* является шифротекстом.

Для дешифрования (a, *b)* вычисляется

$$M = \frac{b}{a^x} \bmod p.$$

Преобразование обратимо, так как $a^x \equiv g^{kx} \bmod p$.

По сути описанное преобразование – это то же самое, что и экспоненциальный ключевой обмен по Диффи-Хеллману, за исключением того, что y – это часть ключа, а при шифровании сообщение умножается на y^k.

Лабораторная работа № 8

Разработка программного средства для реализации алгоритма 128-битового шифрования со статическим 128-битовым ключом

Метод экспоненциального ключевого обмена Диффи-Хеллмана – первая криптосистема с открытым ключом – был изобретен Диффи (W. Diffie) и Хеллманом (M. Hellman) в 1976 году. Криптостойкость метода определяется трудоемкостью вычисления дискретного логарифма. Метод может быть использован для распределения ключей – два абонента могут воспользоваться им для генерации общего секретного ключа, но его нельзя использовать для шифрования и дешифрования сообщений.

Сначала абоненты А и Б вместе выбирают большие простые числа n и g, так, чтобы g было примитивным элементом в конечном поле $GF(n)$. Эти два целых числа хранить в секрете необязательно, абоненты могут договориться об их использовании по несекретному каналу. Эти числа могут даже совместно использоваться группой пользователей. Затем реализуется следующий протокол:

- абонент А выбирает случайное большое целое число x и посылает абоненту Б

$$X = g^x \bmod n;$$

- абонент Б выбирает случайное большое целое число y и посылает абоненту А

$$Y = g^y \bmod n;$$

- абонент А вычисляет

$$k = Y^x \bmod n;$$

- абонент Б вычисляет

$$k' = X^y \bmod n.$$

И k, и k' равны $g^{xy} \bmod n$. Никто из прослушивающих этот канал злоумышленников не сможет вычислить это значение, им известны только n, g, X и Y, и не известны x и y. Для получения x и y необходимо вычислить дискретный логарифм. Таким образом, k – это секретный ключ, который абоненты вычисляют независимо. Выбор g и n может заметно влиять на криптостойкость. n должно быть обязательно большим числом: криптостойкость зависит также от трудоемкости разложения на множители чисел того же размера, что и n. Можно выбирать любое g, являющееся примитивным элементом; нет причин, по которым нельзя было бы выбрать наименьшее возможное g. На самом деле число g может даже и не быть примитивным элементом, оно лишь должно порождать достаточно большую подгруппу мультипликативной группы в $GF(n)$.

Без дополнительных мер безопасности (введения сертификатов открытых ключей) описанный метод ключевого обмена уязвим с точки

зрения атаки, известной под названием «человек посередине» (man-in-the-middle attack).

Предположим, злоумышленник может не только подслушивать сообщения абонентов, но и изменять и удалять сообщения, а также создавать совершенно новые ложные сообщения. Тогда злоумышленник может выдавать себя за абонента Б, сообщающего что-то абоненту А, или за А, сообщающего что-то Б. Атака состоит из следующих действий:

1. абонент А посылает абоненту Б свой открытый ключ. Злоумышленник перехватывает его и посылает Б свой собственный открытый ключ;
2. Б посылает А свой открытый ключ. Злоумышленник перехватывает его и посылает А свой собственный открытый ключ;
3. когда абонент А посылает сообщение абоненту Б, зашифрованное на его открытом ключе, злоумышленник его перехватывает. Так как сообщение в действительности зашифровано на открытом ключе злоумышленника, он расшифровывает его, снова зашифровывает на открытом ключе абонента Б и посылает его Б;
4. когда Б посылает сообщение абоненту А, зашифрованное на ее открытом ключе, злоумышленник его перехватывает. Так как сообщение в действительности зашифровано на открытом ключе злоумышленника, он расшифровывает его и затем снова зашифровывает на открытом ключе А и посылает абоненту А.

Атака возможна, даже если открытые ключи А и Б хранятся в базе данных. Злоумышленник может перехватить запрос А к базе данных и подменить открытый ключ Б своим собственным. То же самое он может сделать и с открытым ключом абонента А. Злоумышленник может атаковать базу данных и подменить открытые ключи Б и А своими соб-

ственными. Затем, дождавшись, когда абоненты начнут обмениваться сообщениями, он выполняет перехват и подмену. Подобная атака весьма эффективна, так как у А и Б нет возможности проверить, действительно ли они общаются именно друг с другом. Если вмешательство злоумышленника не приводит к заметным задержкам при передаче сообщений, абоненты не смогут обнаружить, что кто-то, расположенный между ними, читает все их секретные сообщения.

Каждый абонент криптосети может опубликовать свой открытый ключ $X = g^x \bmod n$ в общей базе данных. Если абонент А захочет установить связь с абонентом Б, ему понадобится только получить открытый ключ Б и затем сгенерировать общий секретный ключ. Он может зашифровать сообщение этим ключом и послать его Б. Абонент Б извлечет открытый ключ А и вычислит общий секретный ключ. Каждая пара абонентов может использовать уникальный секретный ключ; не требуется никаких предварительных обменов данными между ними. Открытые ключи должны пройти сертификацию, чтобы предотвратить атаки, связанные с подменой ключей (в первую очередь для противодействия атаке «человек посередине»), и должны регулярно меняться.

Протокол ключевого обмена для нескольких участников

Описанный выше протокол ключевого обмена легко можно расширить для случая трех и более участников. В приводимом ниже примере А, Б и В вместе генерируют общий секретный ключ.

А выбирает случайное большое целое число x и вычисляет

$$X = g^x \bmod n.$$

Б выбирает случайное большое целое число y и посылает абоненту В

$$Y = g^y \bmod n.$$

К выбирает случайное большое целое число z и посылает абоненту А

$$Z = g^z \bmod n.$$

А посылает Б

$$Z' = Z^x \bmod n.$$

Б посылает В

$$X' = X^y \bmod n.$$

В посылает А

$$Y' = Y^z \bmod n.$$

А вычисляет

$$k = Y'^x \bmod n.$$

Б вычисляет

$$k = Z'^y \bmod n.$$

В вычисляет

$$k = X'^z \bmod n.$$

Секретный ключ k равен $g^{xyz} \bmod n$, и никто из подслушивающих каналы злоумышленников не сможет вычислить это значение. Протокол можно легко расширить для четырех и более участников, просто добавляются участники и этапы вычислений.

Односторонняя генерация ключа

Этот вариант метода Диффи-Хеллмана позволяет А сгенерировать ключ и послать его Б.

А выбирает случайное большое целое число x и генерирует

$$k = g^x \bmod n.$$

Б выбирает случайное большое целое число y и посылает А

$$Y = g^y \bmod n.$$

А посылает Б

$$X = Y^x \bmod n.$$

Б вычисляет

$$z = y^{-1}$$

$$k' = X^{z} \bmod n.$$

Если все выполнено правильно, то $k = k'$. Преимуществом этого метода состоит в том, что k можно вычислить заранее, до взаимодействия, и абонент А может шифровать сообщения с помощью k задолго до установления соединения с Б. Он может послать сообщение сразу множеству абонентов, а передать ключ позднее – каждому в отдельности.

Задание на лабораторную работу

1. Разработать программу для шифрования/дешифрования по схеме ЭльГамаля.

№ варианта	Модуль криптографической схемы
1	1009
2	1013
3	1019
4	1021
5	1031
6	1033
7	1039
8	1049
9	1051
10	1061
11	1063
12	1069
13	1087
14	1091
15	1093

2. Разработать две программы, которые, устанавливая связь с применением сокетов, реализуют одностороннюю генерацию ключа.

Контрольные вопросы

1. Назначение цифровой подписи.

2. В чем отличие криптосхемы ЭльГамаля от RSA?

3. На чем базируется криптостойкость системы ЭльГамаля?

Литература

1. Таненбаум Э. Компьютерные сети. – СПб.: Питер, 2005. – 992 с.
2. Кульгин М.В. Компьютерные сети. Практика построения. Для профессионалов. – СПб.: Питер, 2005. – 462 с.
3. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб.: Питер, 2005 – 672 с.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

по дисциплине

«Информационная безопасность»

Методические указания к выполнению лабораторных работ для студентов направления

15.03.04 Автоматизация технологических процессов и производств

Составитель канд. техн. наук А.А. Евдокимов

Редактор

Подписано в печать . . г. Формат 60×84/16

Уч. – изд. л. Тираж 30 экз.

ГОУ ВПО «Северо-Кавказский государственный технический университет»

Невинномысский технологический институт (филиал)

Невинномысск, ул. Гагарина, 1