

Кафедра «Электроника, вычислительная техника
и информационная безопасность»

В.Т. Еременко, А.Е. Георгиевский, В.М. Донцов,
О.А. Воронина, Н.А. Сафронова

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Методические указания
по выполнению лабораторных работ

Дисциплина – «Информационная безопасность»
Направления – 090900.62 «Информационная безопасность»
230400.62 «Информационные системы
и технологии»

**Допущено ФГБОУ ВПО «Госунiversитет - УНПК»
для использования в учебном процессе в качестве
методических указаний для высшего
профессионального образования**

Орел 2012

Авторы: д-р техн. наук, проф. каф. ЭВТиИБ
канд. техн. наук, доц. каф. ЭВТиИБ
канд. техн. наук, доц. каф. ЭВТиИБ
канд. техн. наук, доц. каф. ЭВТиИБ
канд. техн. наук, ст. преп. каф. ЭВТиИБ

В.Т. Еременко
А.Е. Георгиевский
В.М. Донцов
О.А. Воронина
Н.А. Сафронова

Рецензент: д-р техн. наук, проф. каф. ЭВТиИБ

А.П. Фисун

Методические указания содержат описание лабораторных работ, посвященных изучению дисциплины «Основы информационной безопасности», «Информационная безопасность».

Предназначены студентам, обучающимся по направлениям 090900.62 «Информационная безопасность», 230400.62 «Информационные системы и технологии», изучающим дисциплину «Информационная безопасность».

Редактор С.Н. Иванова
Технический редактор О.И. Усачева

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Государственный университет - учебно-научно-
производственный комплекс»
Лицензия ИД № 00670 от 05.01.2000 г.

Подписано к печати 26.12.2011 г. Формат 60х90 1/16.

Усл. печ. л. 2,6. Тираж 11 экз.

Заказ № _____

Отпечатано с готового оригинал-макета
на полиграфической базе ФГБОУ ВПО «Госуниверситет - УНПК»,
302030, г. Орел, ул. Московская, 65.

© ФГБОУ ВПО «Госуниверситет - УНПК», 2012

Введение

Предметом изучения являются теоретические основы обеспечения безопасности в вычислительных сетях, методы и средства некриптографической защиты, программные средства, криптография и стеганография, идентификация и аутентификация, угрозы безопасности информации в базах данных, операционных системах.

Дисциплина участвует в формировании профессиональных компетенций по пониманию сущности и значения информации в развитии современного общества (ПК-2), способности определять виды и формы информации, подверженной угрозам, виды и возможные методы и пути реализации угроз (ПК-8).

В результате изучения дисциплины студент должен:

- знать общие характеристики процессов сбора, передачи, обработки, накопления и хранения информации, перечень и характеристики угроз информационным ресурсам; основы концепций современной криптографии, принципы защиты информации криптографическими методами; порядок применения указанных средств и программ;
- иметь представление об основах методах защиты информации;
- уметь обрабатывать информацию криптографическими методами; правильно организовывать работу по шифрованию информации; грамотно оформлять результаты своих исследований в виде отчётов, аналитических докладов и статей; логично отстаивать свою точку зрения.
- владеть навыками анализа инфокоммуникационных сетей и систем, их информационной безопасности, применения шифрсредств, основными методами защиты информации; практическими умениями работать с различными криптографическими системами.

Лабораторная работа №1

ПРОСТЕЙШИЕ КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ

1.1 Краткие теоретические сведения

Полибианский квадрат

Одной из первых криптографических систем явился предложенный во II веке до н.э. греческим писателем Полибием метод шифрования сообщений. В основе системы лежит так называемый полибианский квадрат размером 5x5, заполненный алфавитом в случайном порядке (рис 1.1). Для шифрования на квадрате находили букву текста и вставляли в шифровку нижнюю от нее в том же столбце. Если буква была в нижней строке, то брали верхнюю из того же столбца. Так, для слова BARRACUDA получается шифрованное сообщение WPAAPTRHP. Данная система требует наличия идентичных квадратов у отправителя и получателя.

q	r	w	t	y
i	a	m	e	j
o	p	s	l	f
x	n	z	g	d
k	u	b	c	h

Рисунок 1.1

Перестановка по ключу

В эпоху Возрождения распространение получила система, в основе которой используются таблицы, дающие простые шифрующие процедуры перестановки букв в сообщении. Ключом в них служит фраза, задающая перестановку.

Например, сообщение НЕЯСНОЕ СТАНОВИТСЯ ЕЩЕ БОЛЕЕ НЕПОНЯТНЫМ, записывается в таблицу по столбцам.

Используя в виде ключа слово ЛУНАТИК, получаем такую таблицу:

л	у	н	а	т	и	к
4	7	5	1	6	2	3
Н	О	Н	С	Б	Н	Я
Е	Е	О	Я	О	Е	Т
Я	С	В	Е	Л	П	Н
С	Т	И	Щ	Е	О	Ы
Н	А	Т	Е	Е	Н	М

До перестановки

а	и	к	л	н	т	у
1	2	3	4	5	6	7
С	Н	Я	Н	Н	Б	О
Я	Е	Т	Е	О	О	Е
Е	П	Н	Я	В	Л	С
Щ	О	Ы	С	И	Е	Т
Е	Н	М	Н	Т	Е	А

После перестановки

В верхней строке ее записан ключ, а номера под ключом определены по естественному порядку соответствующих букв ключа в алфавите. Если в ключе встретились одинаковые буквы, они бы нумеровались слева направо. После этого текст считывается по строкам. Если его записывать группами по 5 букв, то получается шифровка: СНЯНН БОЯЕТ ЕООЕЕ ПНЯВЛ СЦОЫС ИЕТЕН МНТЕА.

Метод биграмм

В 1508 году в своей работе «Полиграфия» Иоганн Трисемус предложил шифровать по две буквы за раз. Такие шифры были названы биграммными. Наш более известный шифр биграммами называется *Playfair*. Он применялся Великобританией в Первую Мировую войну. Для получения шифра используется ключевое слово или фраза и таблица, которая для русского алфавита может иметь размер 5x6. Ключевое слово вписывалось в таблицу по строкам, а повторяющиеся буквы отбрасывались. Таблица дозаполнялась не вошедшими в нее буквами алфавита по порядку (рис. 1.2).

Р	Е	С	П	У	Б
Л	И	К	А	В	Г
Д	Ж	З	М	Н	О
Т	Ф	Х	Ц	Ч	Ш
Щ	Ь	Ы	Э	Ю	Я

Рисунок 1.2

Текст шифровки строился по очень простым правилам:

1. Если обе буквы биграммы исходного текста принадлежали одной колонке таблицы, то буквами шифра считались буквы, которые лежали под ними. Так, биграмма УН давала текст шифровки ВЧ.

Если буква открытого текста находилась в нижнем ряду, то для шифра бралась соответствующая буква из верхнего ряда и биграмм ОЯ давала шифр ШБ. (Биграмма из одной буквы или пары одинаковых букв тоже подчинялись этому правилу и текст ЕЕ давал шифр ИИ).

2. Если обе буквы исходного текста принадлежали одной строке таблицы, то буквами шифра считались буквы, которые лежали справа от них. Так биграмма ИВ давала текст шифровки КГ. Если буква открытого текста находилась в правой колонке, то для шифра бралась соответствующая буква из левой колонки и биграмма ОМ давала шифр ДН.

3. Если обе буквы биграммы лежали в разных рядах и колонках, то вместо них брали такие две буквы, чтобы вся четверка их составляла прямоугольник. При этом последовательность букв в шифре была зеркальной исходной паре. Так СТ шифровалась как РХ, а ТБ как ШР. При шифровании фразы ПУСТЬ КОНСУЛЫ БУДУТ БДИТЕЛЬНЫ по биграммам получается такая шифровка:

4.

ПУ СТ ЪК ОН СУ ЛЫ БУ ДУ ТБ ДИ ТЕ ЛЬ НЫ

УБ РХ ЫИ ДО ПБ КЩ РБ НР ШР ЖЛ ФР ИЩ ЗЮ

Трафаретный способ

Позднее знаменитый Джероламо Кардано открыл новый класс шифров перестановок, названный решетками или трафаретами. Они представляют собой квадратные таблицы, где четверть ячеек прорезана так, что при четырех поворотах они покрывают весь квадрат. Вписывание в прорезанные ячейки текста и повороты решетки продолжают до тех пор, пока весь квадрат не будет заполнен. Например, процесс шифровки фразы ПРИЕЗЖАЮ ШЕСТОГО трафаретом 4x4 выглядит так:

#	#	П	#
#	#	#	Р
#	И	#	#
Е	#	#	#

0°

З	#	#	#
#	Ж	#	#
#	#	#	А
#	#	Ю	#

90°

#	#	#	
#	#	Ш	#
Е	#	#	#
#	С	#	#

180°

#	Т	#	#
О	#	#	#
#	#	Г	#
#	#	#	О

270°

З	Т	П	
О	Ж	Ш	Р
Е	И	Г	А
Е	С	Ю	О

В

результате получается шифровка ЗТП ОЖШРЕИГАЕСЮО.

Ч		В	Ы	П
О	К	:	Д	У
Г	Ш	З	Э	Ф
Л	Ъ	Х	А	,
Ю	Р	Ж	Щ	Н
Ц	Б	И	Т	Ь
.	С	Я	М	Е

Е	Л	Ц	:	П
.	Х	Ъ	А	Н
Ш	Д	Э	К	С
Ы		Б	Ф	У
Я	Т	И	Ч	Г
М	О	,	Ж	Ь
В	Ш	З	Ю	Р

Рисунок 1.3

Метод двух таблиц

В 1854 году Чарльз Уитсон предложил новую шифровку биграммами, которую называют двойной квадрат (по аналогии с полибианским квадратом). В отличие от полибианского, двойной квадрат использует сразу две таблицы, расположенные по горизонтали, а шифрование идет биграммами, как в шифре Playfair. Пусть имеются две таблицы со случайно расположенными алфавитами (рис.3). Для шифрования сообщение разбивается на биграммы. Первая буква биграммы находится в левой таблице, а вторая в правой. Затем мысленно в таблице строится прямоугольник так, чтобы буквы биграммы лежали в его противоположных вершинах. Другие две вершины этого прямоугольника дают буквы шифровки. Так для биграммы ОЖ получается шифровка АЦ. Если обе буквы биграммы лежат в одной строке, то и буквы шифровки берут из этой же строки. Первая буква биграммы шифровки берется из левой таблицы в столбце, соответствующем второй букве биграммы сообщения. Вторая же буква биграммы шифровки берется из правой таблицы в столбце, соответствующем первой букве биграммы сообщения. Так ТО превращается в ЖБ.

1.2 Задания для выполнения лабораторной работы

Напишите программу, реализующую метод шифровки сообщения:

1. – основанный на полибианском квадрате;
2. – путем перестановки по ключу;
3. – методом биграмм;
4. – трафаретным способом;
5. – методом двух таблиц.

Лабораторная работа №2

ПСЕВДОСЛУЧАЙНЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ

2.1 Краткие теоретические сведения

Основная проблема классической криптографии долгое время заключалась в трудности генерирования непредсказуемых последовательностей большой длины с применением короткого случайного ключа. Для ее решения широко используются генераторы псевдослучайных последовательностей. Получаемые программно из ключа, случайные или псевдослучайные ряды чисел называются гаммой.

Можно сформулировать три основных требования к криптографически стойкому генератору псевдослучайной последовательности (гаммы):

1. Период гаммы должен быть достаточно большим для шифрования сообщений различной длины.

2. Гамма должна быть трудно предсказуемой. Это значит, что если известны тип генератора и кусок гаммы, то невозможно предсказать следующий за этим куском бит гаммы с вероятностью выше определенной. Если криптоаналитику станет известна какая-то часть гаммы, он все же не сможет определить биты, предшествующие ей или следующие за ней.

3. Генерирование гаммы не должно быть связано с большими техническими трудностями.

Генераторы случайных чисел, использующиеся в современных языках программирования, являются несовершенными и непригодными для криптографии. Последовательности, получаемые с помощью данных генераторов, обладают маленьким периодом и статистической предсказуемостью. Генератор FRAN у БК-0010 имеет длину периода всего лишь 2^{15} , а у ДВК генератор имеет период лишь 8192!

Вследствие этого рекомендуется использовать одновременно два или три генератора, смешивая их значения. Если генераторы независимы, то сумма их последовательностей обладает дисперсией, равной сумме дисперсий отдельных последовательностей.

Одним из первых было предложение получать псевдослучайные последовательности как значения дробной части многочлена первой степени:

$$y_n = f(a \cdot n + b)$$

Если n пробегает значения натурального ряда чисел, то поведение y_n выглядит весьма хаотичным. Еще К. Якоби доказал, что при иррациональном коэффициенте a множество $\{y_n\}$ бесконечно.

Из простейших процедур, имитирующих случайные числа, наиболее употребляем так называемый *конгруэнтный способ*, приписываемый Д.Х. Леммеру:

$$g_{n+1} = K \cdot g_n + C \bmod M$$

В нем каждое последующее псевдослучайное число g_{n+1} получается из предыдущего g_n умножением его на K , сложением с C и взятием остатка от деления на M . Причем выбор различных K , C и M дает различные периоды псевдослучайных последовательностей. Математики доказали, что разумнее использовать целые числа. При $C=0$, а $M=2^N$ наибольший период достигается при $K=3+8 \cdot i$ или $K=5+8 \cdot i$ и нечетном начальном числе. При достаточно больших K ряд производит впечатление случайного.

Интересный класс генераторов случайных чисел был предложен Дж. Марсалиа и А. Зейманом. Генераторы этого типа основаны на использовании последовательностей Фибоначчи. Классический пример такой последовательности $\{0,1,1,2,3,5,8,13,21,34,\dots\}$. За исключением первых двух её членов, каждый последующий член равен сумме двух предшествующих. Если брать только последнюю цифру каждого числа в последовательности, то получится последовательность чисел $\{0,1,1,2,3,5,8,3,1,4,2,5,9,4,\dots\}$. Если эта последовательность применяется для начального заполнения массива большой длины, то используя этот массив, можно создать генератор случайных чисел Фибоначчи с запаздыванием, где складываются не соседние, а удаленные числа. Марсалиа и Зейман предложили ввести в схему Фибоначчи «бит переноса», который может иметь начальное значение 0 или 1. Генераторы этого класса можно рассматривать как усилители случайности. Берется случайное заполнение длиной в несколько тысяч бит и генерируются длинные последовательности случайных чисел.

2.2 Задания для выполнения лабораторной работы

1. В 1946 году Джон фон Нейман предложил генератор псевдослучайных чисел, основанный на том, что каждое

последующее число образуется возведением предыдущего в квадрат и отбрасыванием цифр с обоих концов. Постройте диаграмму распределения сгенерированной последовательности.

2. Написать программу, которая получает последовательности псевдослучайных чисел используя дробную часть многочлена

$$y_n = f(a \cdot n + b)$$

Получите последовательности, используя рациональные и иррациональные значения коэффициента a . Постройте диаграммы распределения полученной последовательности.

3. Создайте программу, реализующую конгруэнтный способ Лемера. Используйте целые, рациональные и иррациональные значения коэффициентов K , C и M . Постройте диаграммы распределения полученной последовательности.

4. Пользуясь конгруэнтным способом Лемера при $C=0$, $M=2^N$, $K=3+8 \cdot I$ напишите программу генерирующую последовательность случайных чисел. Постройте диаграмму распределения полученных чисел.

5. Реализуйте алгоритм Марсалиа-Зеймана. Получите последовательность из 1000 первых чисел Фибоначчи, отбрасывая все разряды кроме единиц. Складывая все числа, удаленные на n элементов (используйте «бит переноса»), получите ряд случайных чисел Фибоначчи. Постройте диаграмму распределения полученных чисел.

Лабораторная работа №3 ОСНОВЫ КЛАССИЧЕСКОЙ КРИПТОГРАФИИ

3.1 Краткие теоретические сведения

Шифр замены.

Самый простой и эффективный способ сделать текст нечитаемым – спрятать его, смешав с последовательностью случайных чисел, заданной ключом, с помощью операции XOR. При этом информация прячется в шум. Если числа в используемой для шифрования последовательности статистически независимы друг от друга, то и в шифровке они становятся такими же. Текст превращается во что угодно, то есть в шум. Из-за специфики операции XOR процедура расшифровывания совпадает с процедурой шифрования. Например, обозначив через t вектор бит сообщения, γ вектор случайной последовательности и s шифровку, получаем $t = s \text{ XOR } \gamma$ и $s = t \text{ XOR } \gamma$.

Шифр перестановки.

По причине обратимости шифр замены в чистом его виде никогда не используется, а всегда употребляется вместе с перестановкой, например, бит внутри байта. Если после замены символы превращались во что угодно, но сохраняли свое исходное положение, то после перестановки они еще и расположены там где угодно. Перестановку можно рассматривать как умножение вектора сообщения на матрицу перестановки бит P с элементами 0 и 1 и размером в длину сообщения в битах. Как правило, перестановка делается после наложения на сообщение случайной последовательности, то есть, $s = P(t + \gamma)$. Перестановка может выполняться отдельными битами или группами бит как байты. Вычислительных способов перестановок существует множество, на любой вкус. Например, в программах широко применяется перестановка по номерам N от 0 до $L-1$ на основе рекуррентного выражения:

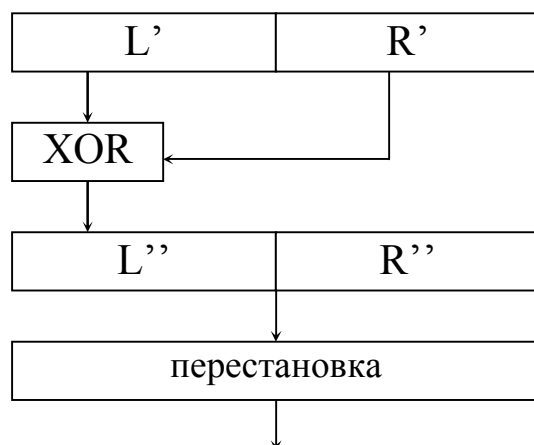
$$N_{i+1} = (K * N_i + M) \text{ MOD } L$$

при выполнении следующих 4 условий:

- 1) K и M берутся из интервала $[1, L-1]$,
- 2) M взаимно просто с L ,
- 3) $K-1$ делится на любой простой делитель L ,
- 4) $K-1$ делится на 4, если L делится на 4.

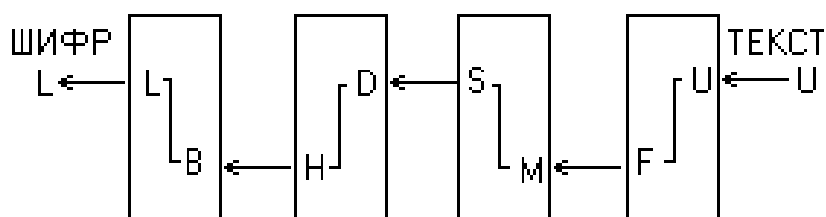
Система шифрования DES.

Стандарт DES был разработан IBM под именем Lucifer и предложен со своими корректировками Национальным Бюро Стандартов США в 1976 году как стандарт шифрования. Суть одного шага криптографического преобразования DES можно описать подобной схемой. Входной блок данных делится пополам на левую L' и правую R' части. После этого формируется выходной массив так, что его левая часть L'' представлена частью R' входного, а правая R'' формируется как сумма L' и R' операцией XOR. Далее выходной массив шифруется перестановкой с заменой. Можно убедиться, что все проведенные операции могут быть обращены и расшифровывание осуществляется за число операций, линейно зависящее от размера блока. В то же самое время, после нескольких таких взбиваний можно считать, что каждый бит выходного блока шифровки может зависеть от каждого бита сообщения.



Шифр Энигмы.

Предшественником современных криптографических машин была роторная машина, изобретенная Эдвардом Хеберном в 1917 году и названная Энигмой. Машина представляла собой ряд вращающихся на одной оси барабанов с электрическими контактами, обеспечивающих множество вариантов простой замены, определяемой текущим положением барабанов. В ранних моделях было 4 барабана, которые в начале работы устанавливались по кодовому слову, а в ходе шифрования они поворачивались при кодировании очередного символа как в счетчике электроэнергии. Таким образом, получался ключ заведомо более длинный, чем текст сообщения.



Например, в первом правом барабане провод от контакта, соответствующего букве U, присоединен к контакту буквы F на другой его стороне. Если же барабан поворачивался на один шаг, то этот же провод соответствовал замене следующей за U буквы V на следующую за F букву G. Так как барабаны соприкасались контактами, то электрический импульс от нажатой клавиши одной с буквой исходного текста, прежде чем достигал выхода, претерпевал 4 замены: по одной в каждом барабане.

3.2 Задания для выполнения лабораторной работы

1. Написать программу, осуществляющую кодирование текста путем шифра замены. Для реализации алгоритма используйте собственный генератор псевдослучайных чисел.
2. Написать программу, шифрующую сообщение методом перестановки. Для реализации алгоритма используйте собственный генератор псевдослучайных чисел.
3. Создать алгоритм «взбивания» сообщения, используя систему шифрования DES. Для реализации алгоритма используйте собственный генератор псевдослучайных чисел.
4. Создайте компьютерную модель криптографической машины Энигма. Для реализации алгоритма используйте собственный генератор псевдослучайных чисел.

Лабораторная работа 4

МОДЕЛИРОВАНИЕ ПРЕОБРАЗОВАНИЙ, ИСПОЛЬЗУЕМЫХ В АЛГОРИТМАХ ЗАЩИТЫ ИНФОРМАЦИИ

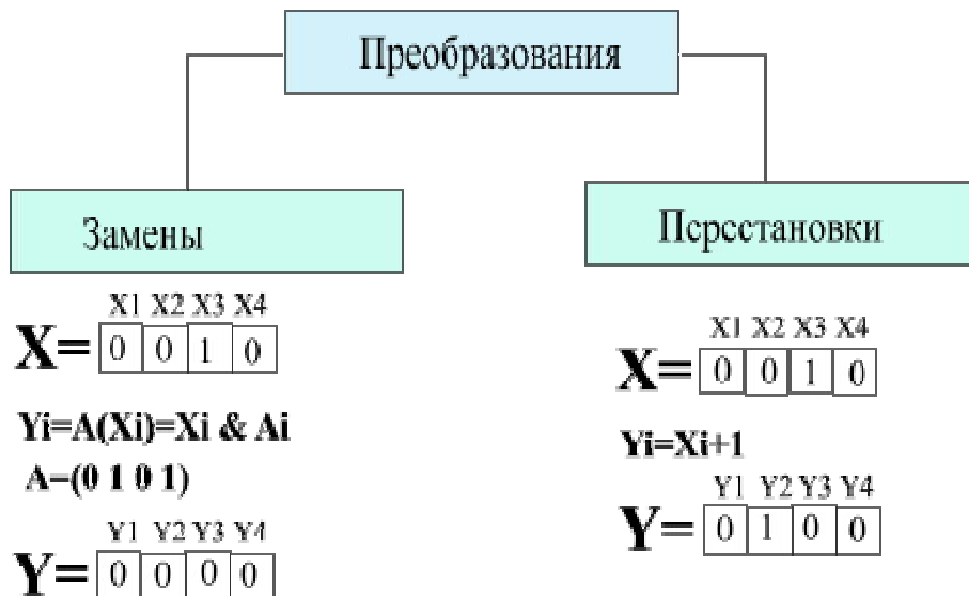
4.1 Краткие теоретические сведения

При построении шифросистем применяют алгоритмы, использующие преобразования из двух больших классов:

- 1) преобразования замены;
- 2) преобразования перестановки;

Под преобразованием будем понимать отображение вектора X в вектор Y , изменяя его по некоторой функции.

Рассмотрим примеры.



В процессе построения алгоритма шифрования применяют **сочетания** различных преобразований из этих классов. Различные виды преобразований осуществляют различные логические элементы, примерами которых могут быть:

- линейные регистры сдвига
- коммутаторы
- сумматоры
- булевы функции усложнения

Преобразования, реализуемые данными элементами, имеют следующий вид.

Преобразования линейного регистра сдвига

Линейный регистр сдвига реализует преобразование A , которое условно можно выразить формулой:

$$A(X) = Y,$$

где X, Y – двоичные векторы, принадлежащие n – мерному пространству и имеющие вид:

$$X = (x_1, x_2, \dots, x_n),$$

$$Y = (y_1, y_2, \dots, y_n).$$

Схема преобразования такова:

$$y_1 = x_2,$$

$$y_2 = x_3,$$

.....

$$y_{n-1} = x_n,$$

$$y_n = a_1 x_1 \oplus a_2 x_2 \oplus \dots \oplus a_n x_n, \text{ где } a_1, \dots, a_n \text{ могут принимать 2 значения: } 0, 1.$$

То есть преобразования линейного сдвига являются комбинацией преобразований замены и преобразований перестановки.

Преобразования коммутатора

Коммутатор реализует преобразование перестановки координат векторов X , которое условно можно выразить формулой:

$$K_n(X) = Y,$$

где X, Y – двоичные векторы, принадлежащие n – мерному пространству и имеющие вид:

$$X = (x_1, x_2, \dots, x_n),$$

$$Y = (y_1, y_2, \dots, y_n).$$

Схема преобразования такова:

Если $K_n = (i_1 \ i_2 \ i_3 \ \dots \ i_n)$, где i_j – номер координаты (индекс),

то $y_1 = x_{i_1}$;

$$y_2 = x_{i_2};$$

.....

$$y_n = x_{i_n}.$$

Рассмотрим пример.

$$K=(3 \ 2 \ 4 \ 1)$$

$$Y_i=X_{k_i}$$

$$X = \begin{array}{c} x_1 \ x_2 \ x_3 \ x_4 \\ \begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 0 \\ \hline \end{array} \end{array}$$

$$Y = \begin{array}{c} y_1 \ y_2 \ y_3 \ y_4 \\ \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \end{array}$$

Преобразования сумматора

Сумматор реализует линейную булеву функцию от n переменных вида:

$$S_n(X)=Y$$

где X, Y – двоичные векторы, принадлежащие n – мерному пространству и имеющие вид:

$$X = (x_1, x_2, \dots, x_n),$$

$$Y = (y_1, y_2, \dots, y_n).$$

Схема преобразования такова:

$$S_n(x_1 \ x_2 \ \dots \ x_n) = x_1 \oplus x_2 \oplus \dots \oplus x_n$$

Построение ГПЧ1

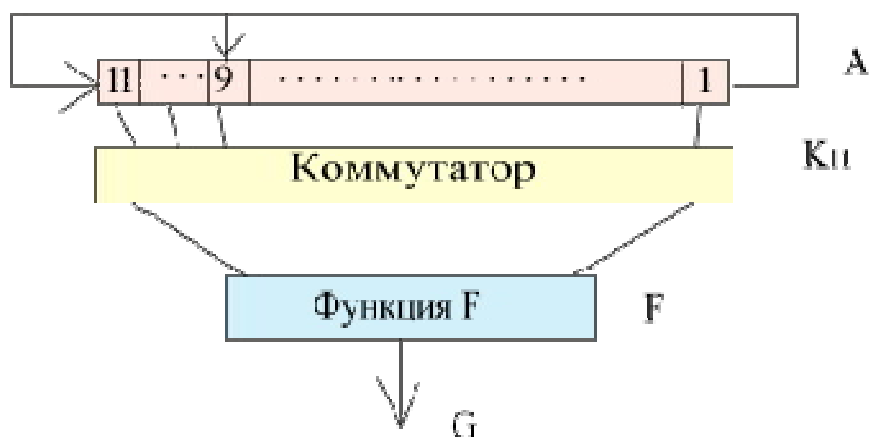
Генератор псевдослучайных чисел первого вида является комбинацией следующих, последовательно применяемых элементов:

A – линейный регистр сдвига, причем в нашем случае управляющим битом является 9-й;

K_{11} – коммутатор;

F_1 – булева функция от 11 переменных.

ГПЧ1 схематично можно представить следующим образом.



$A, K, F1$ – заданы конкретными соотношениями.

Обозначим начальное состояние ячеек $1, 2, 3, \dots, 11$ через b_1, b_2, \dots, b_{11} . Тогда выходная последовательность $G=\{g_i\}$, полученная с помощью ГПЧ1 имеет следующий вид:

$$g_i = F1(K_{11}(A(b_i))).$$

Построение ГПЧ2

Генератор псевдослучайных чисел второго вида является комбинацией следующих, последовательно применяемых элементов:

A – линейного регистра сдвига;

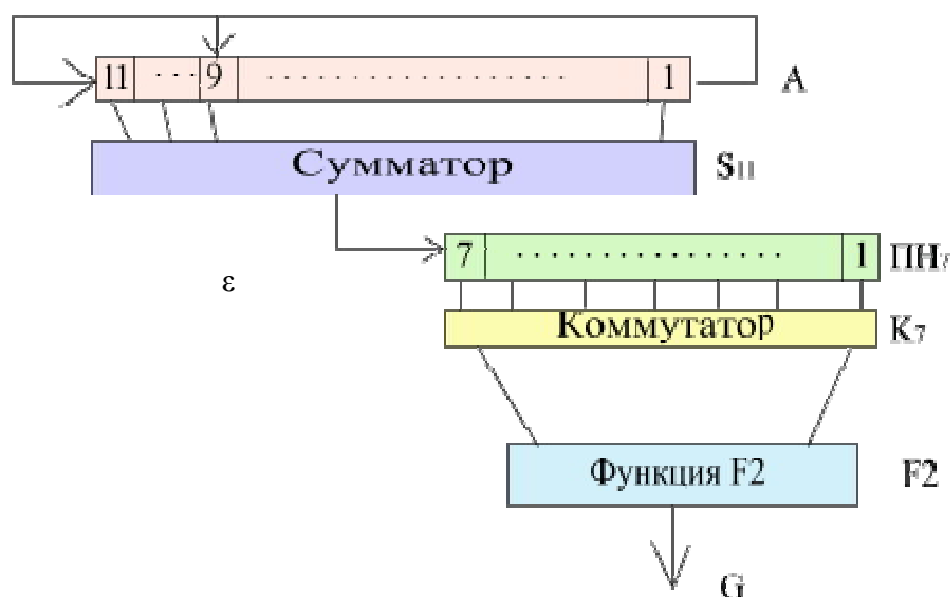
S_{11} – сумматора;

$ПН_{11}$ – продвигающего накопителя, преобразование которого описано ниже;

K_7 – коммутатора;

$F2$ – булевой функции от 7 переменных.

Схема генератора:



Продвигающий накопитель(ПН)

Реализует преобразование следующего вида:

Обозначим состояние ячеек 1, 2,, n элемента ПН через (c_1, c_2, \dots, c_n) . На накопитель поступает входной знак, равный ε , в результате чего данное состояние преобразуется в состояние $(c_2, c_3, \dots, c_n, \varepsilon)$.

Обозначим начальное состояние ячеек линейного регистра сдвига 1, 2, 3,, 11 через b_1, b_2, \dots, b_{11} . А состояние ячеек продвигающего накопителя (ПН₇) в t-й такт через $c^{(t)} = (c^{(t)}_1, c^{(t)}_2, \dots, c^{(t)}_7)$. $C^{(0)}$ – начальное состояние ПН. В t-й такт вырабатывается t-й знак выходной последовательности g_t . Тогда выходная последовательность $G=\{g_i\}$, полученная с помощью ГПЧ2 имеет следующий вид:

$$g_t = F(K_7(c^{(t)})),$$
$$\text{где } c^{(t)} = (c^{(t)}_1, c^{(t)}_2, \dots, c^{(t)}_7) = (c^{(t-1)}_2, c^{(t-1)}_3, \dots, \varepsilon^{(t-1)}),$$
$$\text{где } \varepsilon^{(t-1)} = S_{11}(A^{(t-1)}(b)), t=1, 2, 3, \dots$$

Варианты схем ГПЧ1 и ГПЧ2

В работе предлагается 4 варианта схем генераторов ГПЧ1 и ГПЧ2. Необходимо запрограммировать 2 варианта из предложенных 4-х.

Вариант 1

В преобразовании А последний элемент y_{11} формируется по следующей функции $y_{11} = x_1 \oplus x_9$. Данное преобразование А, будет использоваться и в остальных вариантах.

$$K^{(1)}_{11} = (2\ 4\ 8\ 11\ 10\ 1\ 3\ 5\ 7\ 9);$$
$$F1^{(1)}(x_1, \dots, x_{11}) = x_1 \oplus x_2 * x_3 \oplus x_4 * x_6 * x_9 \oplus x_{10} \text{ or } x_{11};$$
$$K^{(1)}_7 = (3\ 4\ 5\ 6\ 7\ 1\ 2);$$
$$F2^{(1)}(x_1, \dots, x_{11}) = x_1 * x_7 \oplus x_3 * x_4$$

Вариант 2

$$K^{(2)}_{11} = (1\ 3\ 5\ 7\ 9\ 11\ 10\ 8\ 4\ 6\ 2);$$
$$F1^{(2)}(x_1, \dots, x_{11}) = x_1 \text{ or } x_{10} \oplus x_2 \text{ or } x_5 * x_6 \oplus x_{11};$$
$$K^{(2)}_7 = (3\ 5\ 7\ 1\ 4\ 2\ 6);$$
$$F2^{(2)}(x_1, \dots, x_{11}) = x_1 * x_6 \oplus x_4 * x_5$$

Вариант 3

$$K_{11}^{(3)} = (2\ 4\ 6\ 10\ 7\ 8\ 3\ 5\ 1\ 9\ 11);$$

$$F1^{(3)}(x_1, \dots, x_{11}) = x_2 \text{ or } x_{10} \oplus x_7 \text{ or } x_5 * x_6 \oplus x_{11};$$

$$K_7^{(3)} = (3\ 5\ 7\ 2\ 4\ 1\ 6);$$

$$F2^{(3)}(x_1, \dots, x_{11}) = x_2 * x_6 \oplus x_4 * x_7$$

Вариант 4

$$K_{11}^{(4)} = (2\ 7\ 4\ 8\ 1\ 3\ 5\ 9\ 10\ 11\ 6);$$

$$F1^{(4)}(x_1, \dots, x_{11}) = x_3 \text{ or } x_9 \oplus x_8 \oplus x_7 * x_{11};$$

$$K_7^{(4)} = (3\ 2\ 1\ 4\ 5\ 7\ 6);$$

$$F2^{(4)}(x_1, \dots, x_{11}) = x_3 * x_4 \oplus x_6 * x_7$$

4.2 Задания для выполнения лабораторной работы

1) Запрограммировать 2 из 4 предложенных вариантов ГПЧ1 и ГПЧ2.

2) Реализовать запрограммированные варианты при начальных состояниях:

– ячеек линейного регистра сдвига $b=(b_1, b_2, \dots, b_{11})=(11100001010);$

– ячеек ПН $c=(1010101).$

3) Выдать выходные последовательности G

Рекомендуемый интерфейс программы

Задание начальных состояний ячеек линейного регистра сдвига $b=(b_1, b_2, \dots, b_{11})$ и ячеек ПН предпочтительнее организовать в виде двумерных массивов – Stringgrid. Также желательно отображение длины получаемой последовательности.

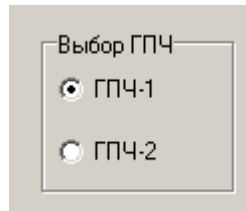
Альфа

Бета

Длина последовательности

16

Необходимо на заданных начальных условиях иметь возможность запуска любого из генераторов. Выбор генераторов можно организовать RadioGroup.



Выбор ГПЧ

☒ ГПЧ-1

☐ ГПЧ-2

Задание варианта параметров ГПЧ.



К11

1

f

1

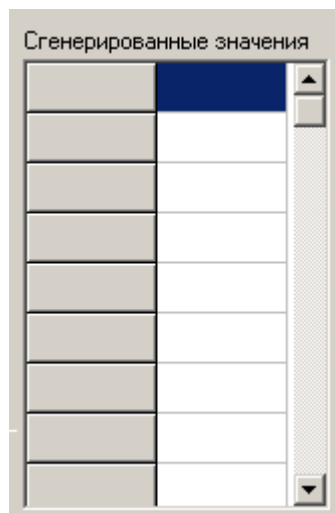
К7

1

Fi

1

Полученные результаты занести в таблицу.



Сгенерированные значения

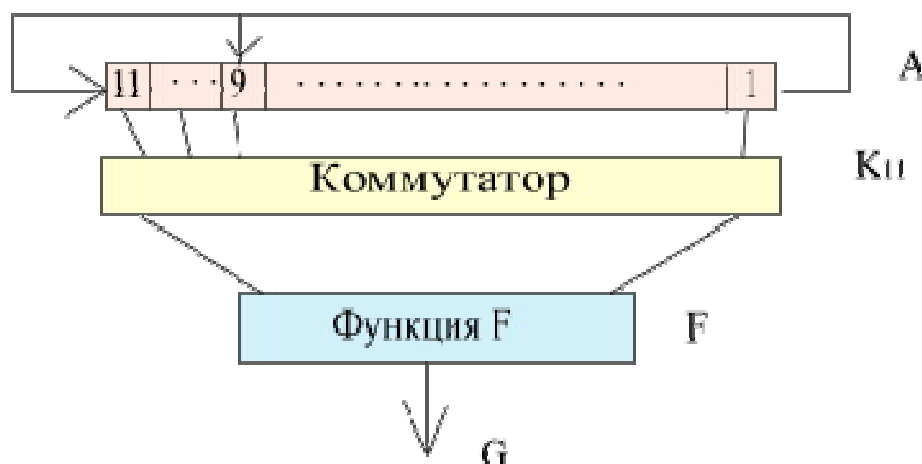
Лабораторная работа 5

ИССЛЕДОВАНИЕ ОСНОВНЫХ ВЕРОЯТНОСТНЫХ ХАРАКТЕРИСТИК ПОСЛЕДОВАТЕЛЬНОСТЕЙ, ВЫРАБАТЫВАЕМЫХ ГЕНЕРАТОРАМИ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

5.1 Краткие теоретические сведения

Для выполнения данной работы необходимо выполнить работу по построению алгоритма и реализации ГПЧ1 и ГПЧ2.

Генераторы псевдослучайных чисел обычно реализуются в виде некоторой схемы. Пример схемы генератора:



При запуске генератора начальные состояния элементов памяти (в приведённом примере – начальные состояния ячеек линейного регистра сдвига) выбираются случайно, равновероятно из некоторого множества X . И при этом начальном состоянии вырабатывается некоторая выходная последовательность.

Основным требованием к последовательностям, вырабатываемым генераторами псевдослучайных чисел, является равная вероятность появления всех знаков в последовательности при случайном равновероятном выборе начального состояния из множества X , а также равная вероятность появления в последовательности всех цепочек знаков до некоторой длины.

Вероятностные характеристики последовательности

Для двоичной последовательности важными характеристиками её вероятностных свойств являются:

- 1) $P(1)$ – Вероятность появления 1 в последовательности, называемая полностью последовательности;
- 2) $P_{\text{пер}}$ – Вероятность знакопеременности в последовательности, то есть вероятность того, что соседние знаки в последовательности являются различными.

Для двоичной последовательности $G=\{g_i\}$, $i=1,2,3,\dots,N$, вырабатываемой некоторым генератором ГПЧ, значение этих характеристик определяется следующими величинами:

$$1) \quad P(1) = (1/N) * \sum g_i,$$

где $i=1, 2, 3, \dots, N$.

Для качественного ГПЧ данная величина должна стремиться к 0,5 при бесконечном увеличении длины последовательности.

$$\text{При } N \rightarrow \infty \quad \lim (P(1)) = 0,5$$

$$2) \quad P_{\text{пер}} = (1/(N-1)) * \sum e_i,$$

где $i=1, 2, 3, \dots, N-1$,

$$e_i = 1, \text{ при } g_i \neq g_{i+1}$$

$$e_i = 0, \text{ при } g_i = g_{i+1}$$

Для качественного ГПЧ данная величина также должна стремиться к 0,5 при бесконечном увеличении длины последовательности.

$$\text{При } N \rightarrow \infty \quad \lim (P_{\text{пер}}) = 0,5$$

В данной работе исследуются выходные последовательности, вырабатываемые генераторами, описанными и реализованными в предыдущей работе – ГПЧ1 и ГПЧ2.

5.2 Задания для выполнения лабораторной работы

1 Генерация последовательностей.

Используя программу предыдущей лабораторной работы, реализующую работу двух моделей генераторов, и выбранные начальные значения ячеек памяти регистра – b и накопителя – c , выработать и записать в память следующие последовательности. Длина последовательности при всех генерациях равна 1000 ($N=1000$).

- 1) С помощью ГПЧ1, используя первый вариант схемы генератора, при начальном заполнении регистра сдвига $b=(01010101011)$ выработать последовательность $\Gamma^{(1)} = \{g_i^{(1)}\}$

- 2) С помощью ГПЧ1, используя второй вариант схемы генератора, при том же начальном заполнении регистра выработать последовательность

$$\Gamma^{(2)} = \{g_i^{(2)}\}$$

- 3) С помощью ГПЧ2, используя первый вариант схемы генератора, при том же начальном заполнении регистра b начальном заполнении продвигающего накопителя $c=(1100111)$ выработать последовательность

$$\Gamma^{(3)} = \{g_i^{(3)}\}$$

- 4) С помощью ГПЧ2, используя второй вариант схемы генератора, при том же начальном заполнении регистра b начальном заполнении продвигающего накопителя c выработать последовательность

$$\Gamma^{(4)} = \{g_i^{(4)}\}$$

2 Расчет величин

С помощью полученных последовательностей вычислить и записать в память следующие последовательности:

$$\Gamma^{(1)} \oplus \Gamma^{(3)} = \{g_i^{(1)} \oplus g_i^{(3)}\}$$

$$\Gamma^{(2)} \oplus \Gamma^{(4)} = \{g_i^{(2)} \oplus g_i^{(4)}\}$$

Вычислить величины $P(1)$ и $P_{\text{пер}}$ для полученных 6 последовательностей.

Лабораторная работа 6

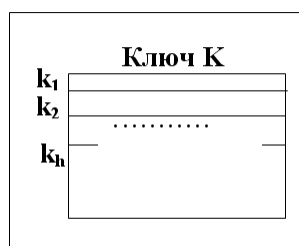
ОПРЕДЕЛЕНИЕ КЛЮЧА ШИФРОСИСТЕМЫ МЕТОДОМ «ВСТРЕЧИ В СЕРЕДИНЕ АТАКИ»

6.1 Краткие теоретические сведения

Для применения шифрования исходный текст разбивается на блоки. Каждый из блоков кодируется определённым ключом K . Для расшифровки необходимо применить тот же ключ K к зашифрованным блокам текста.

Начальные условия

Даны открытый и зашифрованный тексты. Обозначим их через $A=(R_0, R_1)$ и $B=(R_5, R_4)$, где R_i – блок текста, состоящий из 6 разрядов. В общем виде алгоритм шифрования состоит из N циклов шифрования. На каждом цикле применяется цикловый ключ k_i . Цикловые ключи независимо друг от друга изменяют свои значения. Ключ K системы представляет собой сочетание из N цикловых ключей $k_1 k_2 k_3 \dots k_h$.



При известных открытом и зашифрованном текстах необходимо найти ключ K .

Описание модели блочной шифросистемы, используемой в данной работе

Предлагаемая модель преобразует блок открытого текста в блок зашифрованного текста за 4 ($N=4$) цикла шифрования. Для лучшего понимания алгоритма примем следующие обозначения. Пусть преобразование алгоритма имеет вид

$E_k(A)=B$, где A – открытый текст, B – шифротекст.

Данное преобразование можно представить как композицию

$E_{k1}(E_{k2} \dots (E_{k4} (A)))=B$, где E_{ki} - цикловое преобразование с использованием k_i ключа. Каждый ключ представляет собой двоичный вектор длины N . А общий ключ системы – вектор длины $N*N$. В данной работе за длину вектора i -го ключа примем 6 бит.

Таким образом, ключом схемы является двоичная (бинарная) – матрица K размера $4*6$.

$$K = \begin{pmatrix} k_{11} & k_{12} & \dots & k_{16} \\ k_{21} & k_{22} & \dots & k_{26} \\ k_{31} & k_{32} & \dots & k_{36} \\ k_{41} & k_{42} & \dots & k_{46} \end{pmatrix}$$

Заполнение памяти

Будем перебирать все значения $K'=(k_1, k_2)$, то есть 2 цикловых ключа (половину всех цикловых ключей $N/2=2$). На каждом таком ключе K' зашифровываем открытый текст A .

$$E_{K'}(A) = E_{k1}(E_{k2}(A)) = S$$

Проходим 2 цикла вместо 4 (N). Получаем «полузашифрованное» значение S . Будем считать S неким адресом памяти и по этому адресу запишем значение K' . Необходимо перебрать все значения K' .

Определение ключа

Для расшифровки необходимо к блоку $B=(R_5, R_4)$ применить тот же алгоритм, используя на i -м цикле расшифрования цикловый ключ k_{5-i} , $i=1,2,3,4$, и в полученном блоке (R_1, R_0) переставить левую и правую половины.

Перебираем все возможные $K''=(k_3, k_4)$. На получаемых ключах расшифровываем шифротекст B .

$$E_{K''}(B) = E_{k4}E_{k3}(B) = S'$$

Получаем «полурасшифрованное» значение S' .

При шифровании блока А и расшифровке блока В частями истинного ключа, значения S и S' равны.

Проверяем, если по адресу S' не пусто, то достаём оттуда ключ K' и получаем кандидата в ключи $K=(K', K'')$.

Однако нужно заметить, что первый же полученный кандидат K не обязательно является истинным ключом. Для данного открытого текста A и шифрованного текста B выполняется $E_K(A)=B$, но на других значениях открытого текста A' и шифротекста B' , полученного из A' на истинном ключе, равенство может нарушаться. Но иногда бывает достаточно получить такой «псевдоэквивалентный» ключ. В противном же случае после завершения процедур будет получено некое множество ключей $\{K', K''\}$, среди которых находится истинный ключ. Всё зависит от конкретных характеристик криптосистемы.

Криптосистема, применяемая в данной работе

Опишем i -й цикл шифрования, на котором блок (R_{i-1}, R_i) преобразуется в блок (R_i, R_{i+1}) , $i = 1, 2, 3, 4$.

Пусть **xor** - покомпонентное сложение по модулю 2, а $*$ - сложение по модулю 2^6 . Младшие разряды слагаемых расположены справа. T – подстановка циклического сдвига векторов:

$$T(a_1, a_2, a_3, \dots, a_6) = (a_2, a_3, \dots, a_6, a_1).$$

Тогда

$$R_{i+1} = R_{i-1} \text{ xor } f(R_i, K_i),$$

где

$$f(R_i, K_i) = T^3(R_i * K_i), \quad i = 1, 2, 3, 4.$$

Блок (R_0, R_1) является блоком открытого текста, а блок (R_5, R_4) – блоком шифрованного текста. Для расшифрования необходимо к блоку зашифрованного текста (R_5, R_4) применить тот же алгоритм, используя на i -м цикле расшифрования ключевой вектор K_{5-i} , $i = 1, 2, 3, 4$ и в полученном блоке (R_1, R_0) переставить R_1 и R_0 .

6.2 Задания для выполнения лабораторной работы

Пусть имеются блоки открытого текста (R_0, R_1) , (P_0, P_1) , (Q_0, Q_1) и полученные соответственно из этих блоков на неизвестном ключе

K , блоки шифрованного текста (R_5, R_4), (P_5, P_4), (Q_5, Q_4). В результате работы программы вывести значение определённого ключа **K**.

Схема выполнения работы

1) Последовательно перебираем все возможные значения первых двух цикловых ключей k_1, k_2 , тем самым определяя всевозможные $K'=(k_1, k_2)$.

2) Для каждого найденного K' с помощью алгоритма зашифрования, применённого к (R_0, R_1) вычисляем значения пары (R_2, R_3)

3) Записываем значение пары $R''=(k_1, k_2)$ по адресу (R_2, R_3) в память.

4) Последовательно перебираем все возможные значения последних двух цикловых ключей k_3, k_4 , тем самым, определяя всевозможные $K''=(k_3, k_4)$.

5) Для каждого найденного K'' с помощью алгоритма расшифрования, применённого к (R_5, R_4) вычисляем значения пары (R_3, R_2).

6) Обращаемся в память по адресу (R_2, R_3). Если по этому адресу находятся пары R'' (пары значений шестибитовых векторов), то данное значение K'' последовательно объединяем с найденными R'' и получившиеся векторы (R'', K'') рассматриваем как кандидаты в истинное значение ключа.

7) Далее каждый кандидат подвергается дополнительной отбраковке с помощью зашифрования блоков и сравнения результатов зашифрования с блоками соответственно. Неотбракованные значения признаются ключом.

Лабораторная работа 7

ИССЛЕДОВАНИЕ ЭФФЕКТА ВЫРОЖДЕНИЯ СХЕМ УСЛОЖНЕНИЯ, ПОСТРОЕННЫХ НА ОСНОВЕ ЛИНЕЙНЫХ РЕГИСТРОВ СДВИГА.

7.1 Краткие теоретические сведения

Схемы усложнения могут быть построены на основе различных и (или) идентичных линейных преобразованиях.

Схемы, построенные на основе различных линейных преобразований

Линейное преобразование, осуществляемое автономным регистром сдвига с функцией обратной связи $e(x) = a_1 x_1 \oplus \dots \oplus a_n x_n$, обозначим через A_e . Если x, y – двоичные вектора размерности n , $y = A_e(x)$, $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$, то

$$y_1 = x_2,$$

$$y_2 = x_3,$$

$$\dots \dots \dots$$

$$y_{n-1} = x_n,$$

$$y_n = e(x)$$

Рассмотрим линейный **генератор гаммы**, построенный на основе преобразования A_e . Пусть δ – булева функция от n переменных, определяемая равенством

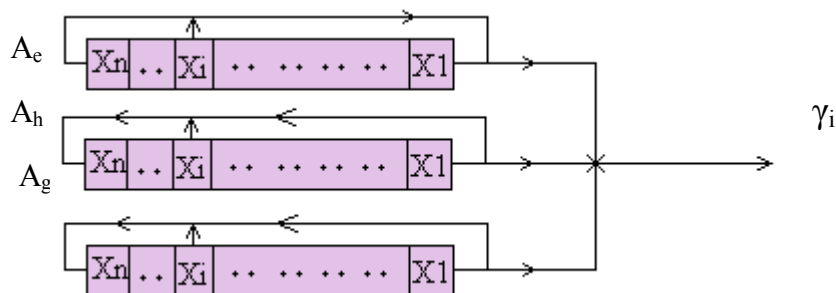
$$\delta(x_1, x_2, \dots, x_n) = x_1.$$

Выходная последовательность рассматриваемого генератора гаммы определяется так:

$$\{\delta(A_e^{i-1}(x))\}, \text{ где } i = 1, 2, \dots$$

Для усложнения выходных последовательностей часто используются комбинации линейных генераторов, но не всякая комбинация осуществляет усложнение последовательности.

В качестве примера рассмотрим узел из трех линейных генераторов, соответствующих линейным функциям e, h, g от n переменных.



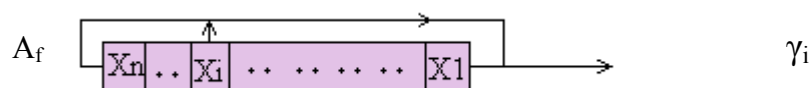
Пусть x – вектор размерности n , тогда

$$\gamma_i = \delta(A_e^{i-1}(x)) \oplus \delta(A_h^{i-1}(x)) \oplus \delta(A_g^{i-1}(x))$$

Требуется проверить, существует ли n -местная линейная функция f и начальное заполнение y – двоичный вектор размерности n , такие, что

$$\gamma_i = \delta(A_f^{i-1}(x))$$

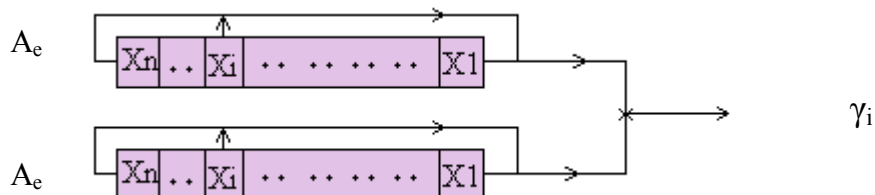
В таком случае предложенная схема была бы равносильна следующей схеме.



Одним из способов решения данной задачи является полный перебор линейных булевых полиномов и векторов их начальных заполнений и сравнении последовательностей, порожденных первой и второй схемами при всех x .

Схемы, построенные на основе идентичных линейных преобразований.

Рассмотрим линейный генератор гаммы, построенный на основе двух идентичных линейных регистров сдвига.



Начальные состояния регистров – два различных двоичных вектора длины n . Выходная последовательность генератора имеет вид:

$$\gamma_i = \delta (A_e^{i-1}(x)) \oplus \delta (A_e^{i-1}(y))$$

Требуется проверить, существует ли линейная функция от n переменных \mathbf{f} и двоичный вектор длины $n - z$, такие, что

$$\gamma_i = \delta (A_f^{i-1}(z))$$

Один из способов решения данной задачи состоит в переборе функций \mathbf{f} и начальных заполнений – z и сравнении результатов их применения с некоторой контрольной последовательностью $\{\gamma_i\}$ выбранной длины.

7.2 Задания для выполнения лабораторной работы

1) Решить задачу из пункта 1 при следующих значениях параметров:

$$n = 11,$$

$$x = (10010010111),$$

$$e(x) = x_1 \oplus x_3 \oplus x_7 \oplus x_9 \oplus x_{10},$$

$$h(x) = x_1 \oplus x_3 \oplus x_6 \oplus x_9,$$

$$g(x) = x_1 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{10}.$$

2) Решить задачу из пункта 2 при следующих значениях параметров:

$$n = 11,$$

$$x = (11000101001),$$

$$y = (00010000111).$$

$$e(x) = x_9 \oplus 1.$$

3) Написать отчет.

Выполнение задания

4.1. Запрограммировать алгоритмы решения задач из пунктов 1 и 2.

4.2. Входными данными для программ должны быть значения параметров:

- для первой задачи n , x , коэффициенты функций e , h , g ;
- для второй задачи n , x , y , коэффициенты функции e .

В отчете должны содержаться:

- 1) Листинги исходных текстов программ решения указанных в пунктах 1 и 2 задач.
- 2) Полученные значения искоемых функций обратной связи и начальных заполнения линейных регистров.

Рекомендуемый интерфейс программы.

Решения первой и второй задач можно реализовать как двумя отдельными программами, так и одной. При реализации одной программой рекомендуется сделать ее с помощью трех форм. Одна главная и по отдельной форме для каждой задачи. На формах решения расположить кнопки начала перебора и компонент для отображения найденных значений коэффициентов функции обратной связи и начального заполнения. Для этого удобно использовать компонент StringGrid. Ввод исходных данных для программы можно реализовать как константы внутри программы, либо любым удобным образом.

8.1 Краткие теоретические сведения

Системой линейных булевых уравнений относительно неизвестных x_1, x_2, \dots, x_m называется совокупность уравнений вида

или в матричной форме

$$A^*X=B,$$

$X = (x_1, x_2, \dots, x_m)$ – вектор неизвестных величин;

$B = (b_1, b_2, \dots, b_n)$ – вектор правых частей системы.

При этом a_{ij} , b_{ij} , x_{ij} — имеют булевы значения, то есть принадлежат множеству $\{0, 1\}$.

Основные определения

Обозначим **расширенную матрицу** системы через $A^{(0)}$, которая получается приписыванием справа к матрице коэффициентов A столбца B правых частей системы.

Решением системы называется упорядоченная совокупность значений неизвестных, удовлетворяющая каждому из уравнений системы.

Если система имеет хотя бы одно решение, то она называется **совместной**; в противном случае система называется **несовместной**.

Обозначим j -ю строку матрицы A через вектор коэффициентов a_j , и будем далее считать, что матрица A - квадратная $(n \times n)$ -матрица.

Матрица называется **невырожденной**, если ее определитель не равен нулю. Матрица является невырожденной тогда и только тогда, когда ее **строки (столбцы) линейно независимы**.

Ранг матрицы равен числу ее линейно независимых строк (столбцов). Для того, чтобы совместная система имела единственное решение, необходимо и достаточно, чтобы ранг матрицы системы был равен числу неизвестных.

СЛБУ с квадратной матрицей системы A ($n = m$) имеет единственное решение тогда и только тогда, когда матрица системы невырожденная.

Обозначим через **def A** **дефект матрицы A**:

$$\text{def } A = n - \text{rang } A,$$

тогда система (если она совместна) имеет $2^{\text{def } A}$ решений.

Самым известным и распространенным методом решения СЛБУ является алгоритм исключения Гаусса.

Алгоритм исключения Гаусса

Алгоритм Гаусса состоит из двух этапов, называемых прямым ходом и обратным ходом.

Первый этап алгоритма (прямой ход) заключается в следующем:

- Найдем в расширенной матрице $A^{(0)}$ столбец с j -м номером. В нем найдем элемент $a_{ij}^{(0)} \neq 0$. Если такой элемент отсутствует, то матрица A вырожденная и система решений не имеет.
- Поменяем местами j -ю и i -ю строки матрицы $A^{(0)}$.
- Сложим строку a_j с теми из $a_{j+1}, a_{j+2}, \dots, a_n$, j -й элемент которых равен 1.

Если матрица A – невырожденная, то выполняя преобразования, описанные в вышеуказанных пунктах, из исходной системы $A \cdot X = B$ получаем новую систему

$$R \cdot X = C,$$

где R – верхняя треугольная матрица с единицами на главной диагонали, X – вектор неизвестных, C – вектор правых частей. Если матрица вырожденная, то на главной диагонали матрицы R располагаются def A нулевых элементов. Если в треугольной матрице R имеется хотя бы одна строка, полностью состоящая из нулей (пусть, например, ее номер равен j), и при этом диагональный ее элемент c_j равен 1, то система линейных булевых уравнений с матрицей коэффициентов A несовместна.

Обратный ход алгоритма Гауса

Второй этап алгоритма рассматривается только для невырожденных матриц, так как вырожденные – решений не имеют. Итак, в невырожденной матрица вычисление неизвестных происходит по следующим формулам:

$$x_n = c_n,$$

$$x_{n-1} = c_{n-1} \oplus r_{n-1,n} * x_n,$$

и так далее до x_1 .

8.2 Задания для выполнения лабораторной работы

Запрограммировать алгоритм Гаусса для решения СЛБУ с квадратной $(n \times n)$ -матрицей коэффициентов A , $n \leq 30$. При этом входными данными программы являются число $n \leq 30$ и имя файла, в котором записана расширенная матрица СЛБУ, то есть матрица с коэффициентами и правыми частями. Файл должен иметь текстовый формат, матрица записывается по строкам слитно, без каких-либо разделительных символов между строками и между элементами строки.

Программа должна выполнять следующие *функции*:

- определять совместность СЛБУ;
- вычислять ранг СЛБУ;

для СЛБУ полного ранга (т.е. в случае, когда $\text{rang } A = n$) находить решение системы, в противном случае выдавать число решений системы.

Лабораторная работа 9

ИЗУЧЕНИЕ НЕКОТОРЫХ СВОЙСТВ ЛИНЕЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТНЫХ МАШИН

9.1 Краткие теоретические сведения

Линейной последовательностной машиной (ЛПМ) называется конечный автомат, преобразующий входную последовательность $u(1), u(2), \dots$ в выходную последовательность $y(1), y(2), \dots$ по правилу

$$s(t+1) = A \cdot s(t) \oplus B \cdot u(t),$$

$$y(t) = C \cdot s(t) \oplus D \cdot u(t),$$

где A, B, C, D - характеристические матрицы размерности $n \times n$, $n \times r$, $m \times n$ и $m \times r$ соответственно.

Введём некоторые обозначения для состояний автомата:

$s(0), s(1), \dots$ - последовательность внутренних состояний ЛПМ,

$s(0)$ - начальное состояние ЛПМ;

$s(t)$ является вектором-столбцом длины n ;

$u(t)$ является вектором-столбцом длины r ;

$y(t)$ является вектором-столбцом длины m ;

Знаком \oplus обозначим операцию покомпонентного сложения столбцов по модулю 2.

При заданном начальном состоянии $s(0)$ линейная последовательностная машина однозначно реагирует на входную последовательность, то есть зная входную последовательность можно заранее предсказать результат на выходе. Это может быть выражено формулой полной реакции ЛПМ следующим образом:

Формула полной реакции:

$$s(t) = A^t \cdot s(0) \oplus \sum_{i=1}^{t-1} A^{t-i} \cdot B \cdot u(i),$$

$$y(t) = C \cdot A^t \cdot s(0) \oplus \sum_{i=0}^t H(t-i) \cdot u(i),$$

где $H(t) = C \cdot A^{t-1} \cdot B$ при $t > 0$ и $H(t) = D$ при $t = 0$.

Классы эквивалентности

Для ЛПМ важным является понятие **эквивалентных состояний**. Два состояния s_1 и s_2 называются **эквивалентными**, если под воздействием одной и той же входной последовательности ЛПМ, находящаяся в начальных состояниях $s(0)=s_1$ и $s(0)=s_2$, генерирует одинаковые выходные последовательности, начиная со второго знака. Таким образом множество всех возможных состояний ЛПМ разбивается на классы эквивалентности.

Классы эквивалентности могут быть описаны с помощью диагностической матрицы K , которая определяется следующим образом:

$$K = \begin{pmatrix} C * A \\ \dots \\ C * A^{n-1} \end{pmatrix}$$

Два состояния являются эквивалентными тогда и только тогда, когда выполняется равенство

$$K * s_1 = K * s_2$$

9.2 Задания для выполнения лабораторной работы

1) Составьте программу генерации выходных последовательностей ЛПМ.

С помощью датчика случайных чисел наработайте входную последовательность $u(1), u(2), \dots, u(t)$ и сохраните ее для дальнейшего использования. Для простоты машинной реализации ограничим размерности векторов $u(t)$, $y(t)$ и $s(t)$ восемью, и будем представлять вектора в байтовом виде.

Постройте таблицы результатов умножения характеристических матриц $A=[a_1, \dots, a_8]$, $B=[b_1, \dots, b_8]$, $C=[c_1, \dots, c_8]$, $D=[d_1, \dots, d_8]$ на всевозможные вектора (байты), где a_i (b_i , c_i , d_i) есть байт, представляющий 1-ю строку матрицы A (B , C , D), $i=1, 2, \dots, 8$.

Используя заданные матрицы a, b, c, d и входную последовательность из п.2.2, выработайте выходные

последовательности для каждого из 255 возможных начальных состояний ЛПМ.

Отсортируйте полученные последовательности и определите число классов эквивалентных состояний ЛПМ.

Повторите задание при другой входной последовательности и сравните полученные результаты.

2) Выработайте выходные последовательности при каждом начальном состоянии ЛПМ и одинаковой входной последовательности.

3) Определите число классов эквивалентных состояний.

4) Повторите пункты 1.2 и 1.3 с другой входной последовательностью, проанализируйте полученные результаты.

Вариант № 1

A=[1, 235, 32, 2, 15, 3, 32, 100] B=[12, 45, 56, 13, 127, 214, 1, 11]
C=[134, 230, 200, 6, 9, 3, 101, 201] D=[154, 254, 16, 18, 20, 154, 3, 7]

Вариант № 2

A=[2, 255, 32, 2, 15, 3, 32, 100] B=[2, 45, 56, 13, 127, 214, 1, 11]
C=[34, 230, 200, 6, 9, 3, 101, 201] D=[14, 254, 16, 18, 20, 154, 3, 7]

Вариант № 3

A=[10, 255, 32, 2, 15, 3, 32, 100] B=[1, 45, 56, 13, 127, 214, 1, 11]
C=[13, 230, 200, 6, 9, 3, 101, 201] D=[54, 254, 16, 18, 20, 154, 3, 7]

Вариант № 4

A=[1, 255, 32, 2, 15, 3, 32, 100] B=[12, 45, 56, 13, 127, 214, 1, 11]
C=[14, 230, 200, 6, 9, 3, 101, 201] D=[14, 254, 16, 18, 20, 154, 3, 7]

Лабораторная работа 10

ПОСТРОЕНИЕ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В ЗНАКОВО-ЦИФРОВОЙ СИСТЕМЕ С МИНИМАЛЬНЫМ ВЕСОМ ДЛЯ УСКОРЕНИЯ АЛГОРИТМА ЭКСПОНЕНЦИРОВАНИЯ

10.1 Краткие теоретические сведения

Экспоненцирование больших чисел (по модулю большого целого числа) лежит в основе ряда хорошо известных криптографических алгоритмов, таких, как алгоритмы открытого распределения ключей и цифровой подписи. Подобные операции имеют существенную вычислительную сложность, что может привести к нежелательным временным зависимостям алгоритмов подписи и шифрования.

Одним из методов выполнения модульного экспоненцирования является метод «квадратов и умножений слева направо». Он состоит в следующем:

Необходимо выполнить вычисление

$$X = M^e \bmod N. \quad (10.1)$$

Экспонента e имеет двоичное представление. То есть:

$$e = (e(s-1), e(s-2), \dots, e(0)). \quad (10.2)$$

Таким образом в десятичном виде e определяется как

$$e = e(s-1) \cdot 2^{s-1} + e(s-2) \cdot 2^{s-2} + \dots + e(0).$$

Алгоритм вычисления (1) имеет вид

```
X := 1;
For I := s-1 to 0 do
  Begin
    X := X2 mod N;
    X := X * me(i) mod N;
  End;
```

В конце вычислений результат находится в переменной X .

Из алгоритма видно, что трудоемкость вычисления (10.1) зависит от числа единиц в двоичном представлении экспоненты e .

Знаково-цифровая система представления чисел – это представление вида $e = (e(s-1), e(s-2), \dots, e(0))$, в котором наряду с 0

и 1 используется значение -1 . При этом сохраняется алгоритм вычисления экспоненты. Однако знаково-цифровое представление чисел неоднозначно, и так как трудоемкость алгоритма вычисления опять зависит от количества ненулевых элементов, то для минимизации трудоемкости следует подобрать знаково-числовое представление с минимальным весом.

Алгоритм минимизации веса

Существует алгоритм для минимизации веса. Исходными данными для него является двоичное представление числа e . В алгоритме вводится дополнительная переменная $e(s)$ (отсутствующая в представлении (2)), ее начальное значение равно 0.

Структура алгоритма по шагам:

1) Если i – максимальный индекс, для которого $e(i) \neq 0$ и $e(i+1) \neq 0$ – не существует, то конец.

2) Если $e(i) \neq e(i+1)$, то

$e(i) := e(i+1)$;

$e(i+1) := 0$;

Иначе

3) Пусть j – максимальный индекс, для которого выполняются условия:

$$\begin{cases} e(j) \neq e(i) \\ e(j-1) = e(j-2) = \dots = e(i). \end{cases}$$

Если $e(j) = 0$, то $e(j) := e(i)$, иначе $e(j) := 0$.

Положить $e(i) = -e(i)$ и $e(k) = 0$ для всех $k \in (i, j)$.

Перейти к шагу 1.

Данный алгоритм вычисляет знаково-цифровое представление числа e с минимальным весом.

10.2 Задания для выполнения лабораторной работы

1) Составьте программу минимизации веса числа в знаково-цифровом представлении.

2) Проведите тесты со случайно сгенерированными числами длиной 128, 256, 512 и 1024 бита. Определите для каждой длины среднее значение веса минимизированного знаково-цифрового представления.

3) Рассчитайте выигрыш в трудоемкости вычисления операции модульного экспоненцирования с переходом от двоичного знаково-цифровому представлению экспоненты.

4) Результаты отразите в отчете.

Литература

1. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии. Учебное пособие. – М.: Гелиос-АРВ, 2001.
2. Столлингс В. Криптография и защита сетей. Принципы и практика. 2-е изд. – М: Вильямс, 2001.
3. Введение в криптографию / Под общей ред. Яценко В.В. – М: МЦИМО, «ЧеРо», 1998.
4. Белкин П.Ю., Михальский О.О., Першаков А.С. и др. Программно-аппаратные средства обеспечения информационной безопасности. Защита программ и данных: Учебное пособие для вузов – М.: Радио и связь, 2000
5. Мельников В. П. Информационная безопасность и защита информации : учеб. пособие для вузов. - М. : Академия , 2008.
6. Шаньгин В. Ф. Защита компьютерной информации. Эффективные методы и средства : учеб. пособие для вузов. - М. : ДМК Пресс , 2008.
7. Герасименко В.А. Защита информации в автоматизированных системах обработки данных. В 2-томах – М.: Энергоатомиздат, 1999