МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Новосибирский государственный технический университет»

МЕТОДЫ КРИПТОГРАФИИ И ЗАЩИТЫ ДАННЫХ

Методические указания по курсу «Защита компьютерной информации»
для студентов дневного и заочного отделений АВТФ,
направления 09.03.01 «Информатика и вычислительная техника».

НОВОСИБИРСК
2019

Составители: к.т.н., доцент. *Качальский В.Г.*,
старший преподаватель. *Павенко Е.Н*

Рецензент: к.т.н., доцент каф. АСУ Секаев В.Г.

Работа подготовлена на кафедре
Автоматизированных систем управления

ОГЛАВЛЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1

Методы шифрования и основные понятия криптографии.

1. Исторический обзор.

Устройствами для обеспечения конфиденциальности сообщений человечество занималось очень давно. Уже в 5-ом веке до нашей эры появилось первое приспособление для кодирования текста- скиталь. Пояс почтальона (в дальнейшем лента) наматывался на деревянный цилиндр или конус (скиталь), вдоль оси вращения записывался текст (несколько строк, причем каждая буква на соседний виток). Далее пояс раскручивался и на нем была видна хаотическая последовательность букв. Получатель информации наматывал пояс на аналогичную скиталь и прочитывал текст. Такой метод шифрования можно назвать "перестановками".

Позже для шифрования текстовых сообщений и обеспечения конфиденциальности переписки стали использовать простейшие устройства: решетки, циферблаты и т.п. характерным примером является шифр Цезаря. Используется диск по периметру которого записан весь буквенный алфавит. В подлежащем засекречиванию тексте каждая буква заменяется на другую, отстоящую (от данной) на 3 знака по периметру диска. Это шифр "замены". Для усложнения сдвиг может производиться на переменное количество знаков (шифр Виженера). Изменение количества знаков производится в соответствии с ключевым словом, которое повторяется столько раз, сколько нужно для замены всех букв открытого текста.

Дальнейшее развитие шифра Виженера это использование текста какой либо книги или книжных шифров. Математически это можно представить так:

$$L_x = (M_x + K_x) \bmod 31$$

- сложение по модулю 31, где Lx- номер буквы шифротекста;

Mx- номер буквы открытого текста; Kx- номер буквы ключа.

Повторное применение шифра Виженера называют составным шифром Виженера:

$$L_x = (M_x + K_x^1 + K_x^2 + \ldots + K_x^N) \bmod 31$$

В конце позапрошлого века появились механические машины, в которых для преобразования текста, использовались несколько кодовых колес, цилиндров или других элементов, перемещающихся друг относительно друга в процессе обработки текста. Это так называемые ручные машины.

Упрощенную работу таких машин можно представить следующим образом. По периметру каждого колеса записаны все знаки используемого алфавита, причем на каждом колесе последовательность знаков разная. Все колеса размещены на одной оси и при повороте предыдущего колеса на один знак, (или на один оборот) последующие смещаются на один или несколько знаков, относительно друг друга. Колеса помещены в кожух, имеющий два окна. Через одно окно виден один знак первого колеса, через другое один знак последнего колеса. Поворотом первого колеса в первом окне устанавливается знак текста, подлежащий засекречиванию, в последнем окне считывается знак зашифрованного текста. Вращая в том же направлении первый диск устанавливают, в окне следующий знак текста и т.д.

Для надежной защиты телеграфных сообщений, после первой мировой войны появились электрические и электромеханические машины. Вначале это были громоздкие релейные системы и машины, имеющие колеса с профилированными ребордами. Работа некоторых из них аналогична механическим дисковым шифромашинам. Однако вместо нанесенных знаков алфавита диски имеют с одной стороны входные электрические контакты (их число равно числу знаков используемого алфавита), а с другой стороны диска столько же выходных контактов. Входные и выходные контакты соединены между собой в хаотичном, но заранее заданном порядке. Контакты смежных дисков обеспечивают надежное электрическое соединение. Ввод текста осуществляется с клавиатуры, аналогичной клавиатуре пишущей машинки или телетайпа.

В 30-х годах в Швеции появилась весьма компактная и простая в работе шифромашина "Хагелин". Шифромашины этой фирмы и их модификации

изготовлены в огромном количестве и были на вооружении военных, правительственных и дипломатических органов многих стран мира. Так только для вооруженных сил США в период второй мировой войны было заказано около 140 тысяч экземпляров. После войны штаб-квартира фирмы переместилась в Швейцарию, где эта фирма успешно функционирует до сих пор в городе Цуг под названием Crypto AG.

Перед второй мировой войной появились электронные машины. Первые из них были реализованы на электронных лампах и были, по существу, электронными аналогами самых совершенных механических разработок фирмы "Хагелин".

После войны были построены транзисторные шифромашины, затем появились машины, построенные на основе микроэлектронных интегральных схем. Микроминиатюризация позволила реализовать в относительно компактных шифромашинах этого поколения исключительно сложные алгоритмы, требующие для своей реализации десятки тысяч электронных элементов, объединенных в сотни регистров и схем. Применение малогабаритной цифровой памяти с большими сроками хранения и объемами хранимой информации позволило снабжать машину впрок большим количеством качественных ключей.

Устройства для обеспечения конфиденциальности речевых сообщений появились значительно позже, чем для текстовых. Однако уже в 1875 году, спустя всего лишь 5 лет после изобретения телефона, в США была подана заявка на изобретение, относящееся к закрытию телефонной связи.

В настоящее время для зашифрования телефонных переговоров применяют два принципиально различных метода: преобразование аналоговых параметров речи и цифровое зашифрование. Оба метода предусматривают использование шифрообразующих устройств, аналогичных тем, которые используются в шифромашинах для обработки текстовых сообщений.

2. Теоретическое обоснование криптографии.

Наиболее фундаментальные работы по защите информации криптографическими методами появились после Второй мировой войны. Наиболее известны работы Шеннона, в том числе опубликованный в 1949 г. доклад "Теория

связи в секретных системах". В основе этих работ лежат следующие предположения:

- Криптограф пытается найти методы обеспечения секретности и (или) аутентичности (подлинности) сообщений.
- Криптоаналитик пытается выполнить обратную задачу: раскрыть шифртекст или подделать его так, чтобы он был принят как подлинный.
- При этом допускается, что криптоаналитик противника имеет полный шифртекст и ему известен алгоритм шифрования, за исключением секретного ключа.
- При разработке методов наиболее надежной защиты информации, криптограф допускает также, что криптоаналитик противника может иметь несколько отрывков открытого текста и соответствующего ему шифртекст-а. На основе этого криптоаналитик может навязать фиктивный текст.
- Возможно также, что криптоаналитик противника, может попытаться навязать ранее полученный шифртекст вместо фактически передаваемого.

Модель криптографической системы, предложенной Шенноном, показана на рис.1.

Данный метод называется ***Гаммирование*** – это наложение/снятие на открытые/зашифрованные данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных/открытых данных.

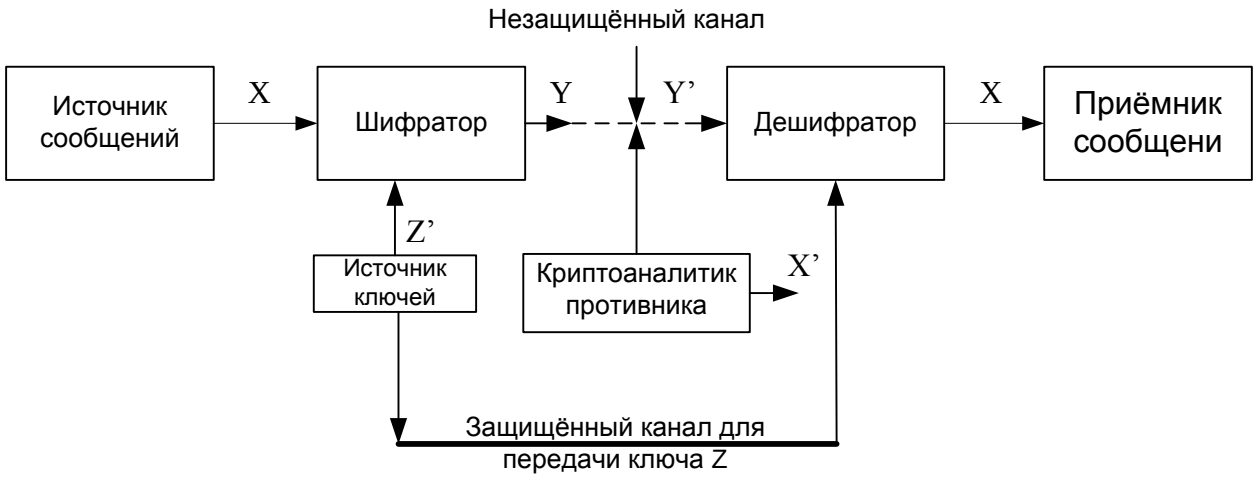

Рис.1. Модель криптографической системы

Источник сообщений порождает открытый текст

$X=\{x_1,x_2,\dots x_i\}$, i – число символов открытого текста. (1)

Источник ключей генерирует k знаков ключа - символов некоторого конечного алфавита А. Шифратор преобразует открытый текст X в шифртекст:

$Y=\{y_1,y_2,\dots y_j\}$, i – число символов закрытого текста. (2)

Последнее преобразование записывается в виде:

Y=E(X;Z), E – прямое преобразование(шифрование). (3)

$Z=\{z_1,z_2,\dots z_k\}$ - секретный ключ, k – размерность ключа. (4)

В нашем случае суммирование по модулю А общего числа допустимых символов в шифруемом алфавите.

$Y = (X + Z) \bmod A$

Дешифратор, получив шифртекст, выполняет обратное преобразование:

X=D(Y;Z), где (5)

$D=E^{-1}$, - обратное преобразование(дешифрование). (6)

Пример.

Алгоритм шифрования:

Если в алфавите «А» - 30 допустимых букв:

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Буква	А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П

№	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Буква	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Э	Ю	Я	Ь	Ъ

(отсутствуют буквы й, ё, ы) то ключевому слову "ваза" соответствует последовательность сдвигов на 3, 1, 8, 1 знаков и слово **КРИПТОГРАФИЯ** после преобразования превратится в НССРХПМСГХСЬ.

Алгоритм дешифрования:

При дешифровании проводится обратное преобразование текста НССРХПМСГХСЬ в исходное слово КРИПТОГРАФИЯ.

Важной частью модели криптографической системы является "защищенный" канал, по которому передается секретный ключ:

$Z=\{z_1,z_2,\dots z_k\}$**.**

Таким каналом может быть канал электросвязи с шифрованием другими устройствами, нежели показанные на рис 1. однако чаще ключи развозятся специальными сотрудниками. В этом случае ключ представляет собой таблицу цифр, перфоленту, магнитную карточку или другой тип носителя с записанной информацией.

Следует особо отметить, что ***X, Y*** и ***Z***- независимые случайные величины. Статистические свойства величины X определяются источником сообщения, Y задается разработчиком криптографической системы, а Z создается и тиражируется специальным устройством заготовки ключей (источником ключа).

К. Шеннон рассматривал вопрос о стойкости криптографических систем с теоретической и практической точек зрения.

Первый вопрос он сформулировал так:

"Насколько надежна некоторая система, если криптоаналитик противника не ограничен во времени и обладает всеми необходимыми средствами для анализа криптограмм?"

Второй вопрос (о практической стойкости) в постановке Шеннона, можно сформулировать следующим образом:

"Надежна ли некоторая система, если криптоаналитик противника располагает ограниченным временем и вычислительными возможностями для анализа криптограмм?"

Решение вопроса о теоретической стойкости привело к следующему выводу: объем секретного ключа для построения теоретически стойкого шифра недопустимо велик для большинства практических применений.

Шеннон доказал, что при двух допущениях совершенно секретные системы существуют. Эти допущения следующие:

- секретный ключ используется только один раз и
- криптоаналитику доступен лишь шифртекст.

На основе этих допущений совершенная секретность означает, что открытый текст ***X*** и шифртекст ***Y*** статистически независимы, т.е.

$$P(X=x/Y=y)=P(X=x),$$

для всех возможных открытых текстов ***X*** и шифртекстов ***Y***.

Другими словами криптоаналитик не может улучшить апостериорное распределение вероятностей открытого текста, используя знание шифртекста независимо от того, каким временем и вычислительными ресурсами он располагает

для анализа. Было доказано также, что ключ не должен быть короче открытого текста т.е. $Z \geq X$.

Таким образом, возникает проблема секретного ключа. Она заключается в том, что на один знак открытого текста требуется, по крайней мере, один знак секретного ключа. При обработке огромных массивов информации, например в крупных вычислительных системах, обеспечить это достаточно сложно или такое решение неприемлемо по экономическим причинам. Поэтому в ряде случаев используют несовершенные шифры, не обеспечивающие совершенную секретность.

3. Задание на лабораторную работу.

3.1. На основе Примера 1. построить блок схему шифрующего и дешифрующего устройства.

3.2. Записать математические уравнения прямого и обратного преобразования исходного текста в криптограмму и обратно, используя выражения и уравнения (1) - (6).

3.3. Включить в алфавит все буквы кириллицы и вспомогательные знаки. Провести 3-5 примеров шифрования-дешифрования текста.

3.4. Составить алгоритм шифрования-дешифрования.

3.5. Написать программу шифрования-дешифрования на любом доступном Вам языке программирования с формированием загрузочного .exe файла

- с клавиатуры произвольного текста;
- с чтением и записью в текстовый файл.

Доказать работоспособность на примерах по п.п.3.3

4. Содержание отчета

4.1. Привести в отчете все преобразования, примеры, алгоритмы и программу данного простейшего алгоритма криптографии.

ЛАБОРАТОРНАЯ РАБОТА №2

Шифрование методом замены (подстановки)

Наиболее простой метод шифрования. Символы шифруемого текста заменяются другими символами, взятыми из одного алфавита (одноалфавитная замена) или нескольких алфавитов (многоалфавитная подстановка).

1**.** Одноалфавитная подстановка

Простейшая подстановка - прямая замена символов шифруемого сообщения другими буквами того же самого или другого алфавита.

Примеры таблиц замены:

1.

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

2.

А Б В Г Д Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ы Ъ Э Ю Я
Q W E R T Y U I O P [ ] A S D F G H J K L Z X C V B N M < > @ %

Первая строка – исходный алфавит, вторая строка – посимвольная замена.

Стойкость метода простой замены низкая. Зашифрованный текст имеет те же самые статистические характеристики, что и исходный, поэтому зная стандартные частоты появления символов в том языке, на котором написано сообщение, и подбирая по частотам появления символы в зашифрованном сообщении, можно восстановить таблицу замены. Для этого требуется лишь достаточно длинный зашифрованный текст, для того, чтобы получить достоверные оценки частот появления символов. Поэтому простую замену используют лишь в том случае, когда шифруемое сообщение достаточно коротко!

Стойкость метода равна 20 - 30, трудоемкость определяется поиском символа в таблице замены. Для снижения трудоемкости при шифровании таблица замены сортируется по шифруемым символам, а для расшифровки формируется таблица дешифрования, которая получается из таблицы замены сортировкой по заменяющим символам.

Многоалфавитная замена повышает стойкость шифра.

2. Многоалфавитная одноконтурная обыкновенная подстановка

Для замены символов используются несколько алфавитов, причем смена алфавитов проводится последовательно и циклически: первый символ заменяется на соответствующий символ первого алфавита, второй - из второго алфавита, и т.д. пока не будут исчерпаны все алфавиты. После этого использование алфавитов повторяется.

Рассмотрим шифрование с помощью ***таблицы Вижинера*** - квадратной матрицы с n^2 элементами, где n - число символов используемого алфавита. В первой строке матрицы содержится исходный алфавит, каждая следующая строка получается из предыдущей циклическим сдвигом влево на один символ.

Таблица Вижинера для русского алфавита:

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

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

Для шифрования необходимо задать ключ - слово с неповторяющимися символами. Таблицу замены получают следующим образом: строку "Символы шифруемого текста" формируют из первой строки матрицы Вижинера, а строки из раздела "Заменяющие символы" образуются из строк матрицы Вижинера, первые символы которых совпадают с символами ключевого слова.

При шифровании и дешифровании нет необходимости держать в памяти всю матрицу Вижинера, поскольку используя свойства циклического сдвига, можно легко вычислить любую строку матрицы по ее номеру и первой строке.

При шифровании символы из первой строки заменяются символами остальных строк по правилу

$$a(1,i) \rightarrow a(k,i), \tag{7}$$

где k - номер используемой для шифрования строки.

Используя свойства циклического сдвига влево элементы k-ой строки можно выразить через элементы первой строки

$$a(k,i) = \begin{cases} a(1,i+k-1), & \text{если } i \le n-k+1 \\ a(1,i-n+k-1), & \text{если } i > n-k+1 \end{cases} \tag{8}$$

При дешифровании производится обратная замена

$$a(k,i) \rightarrow a(1,i). \tag{9}$$

Необходимо решить следующую задачу: пусть очередной дешифруемый символ в тексте - a(1,j) и для дешифрования используется k-я строка матрицы Вижинера. Необходимо найти в k-ой строке номер элемента, равного a(1,j). Очевидно,

$$a(1,j) = \begin{cases} a(k,j-k+1), & \text{если } j \ge k \\ a(k,n-k+j+1), & \text{если } j < k \end{cases}$$

Таким образом при дешифровании по k-ой строке матрицы Вижинера символа из зашифрованного текста, значение которого равно a(1,j), проводится обратная подстановка

$$a(1,j) \rightarrow \begin{cases} a(1,j-k+1), & \text{если } j \ge k \\ a(1,n-k+j+1), & \text{если } j < k \end{cases}$$

Стойкость метода равна стойкости метода подстановки, умноженной на количество используемых при шифровании алфавитов, т.е. на длину ключевого слова и равна 20*L, где L - длина ключевого слова.

3. Задание на лабораторную работу.

3.1. На основе примера таблицы замены и таблицы Вижинера построить блок схему шифрующего и дешифрующего устройства.

3.2. Записать математические уравнения прямого и обратного преобразования исходного текста в криптограмму и обратно, используя выражения и уравнения (7) - (9).

3.3. Включить в алфавит все буквы кириллицы и вспомогательные знаки. Провести 3-5 примеров шифрования-дешифрования текста.

3.4. Составить алгоритм шифрования-дешифрования.

3.5. Написать программу шифрования-дешифрования на любом доступном Вам языке программирования и доказать работоспособность на примерах с чтением и записью в файл и с клавиатуры произвольного текста.

4. Содержание отчета

4.1. Привести в отчете все преобразования, примеры, алгоритмы и программу данного простейшего алгоритма криптографии.

ЛАБОРАТОРНАЯ РАБОТА №3

Вычисление циклического контрольного кода

1. Теоретический материал.

Целостность информации контролируется с помощью техники, которая называется полиномиальными кодами или циклическим избыточным кодом (Cyclic Redundancy Code) или CRC кодом.

CRC обеспечивает обнаружение ошибок при передаче или хранении информации с вероятностью до 99%.

В случае, если какой-либо файл пользователя был испорчен при переносе с одного носителя на другой, попытка чтения испорченного файла вызывает сообщение об ошибке – неверная CRC.

CRC используется практически всеми архиваторами. Архиватор для каждого упаковываемого файла вычисляет значение CRC. При распаковке архива прежде всего проверяется целостность файлов, для чего архиватор снова вычисляет CRC и сравнивает с тем, которое было вычислено при упаковке. В случае, если целостность архива нарушена, при распаковке выводится сообщение о том, что у файла неправильное значение CRC, и распаковка не производится.

2. Способы вычисление циклического контрольного кода

В основе теории вычисления циклического контрольного кода лежит понятие полинома или, как его еще называют, многочлена.

Полином - это формально заданный степенной ряд, т.е. сумма множества степенных выражений независимых переменных. Например,

$$3x^4 + 5x^2 + 7x + 6.$$

Степенью полинома называют число r равное максимальному показателю степени полинома. Для приведенного примера r = 4.

Формальная запись примера должна выглядеть $3x^4 + 0x^3 + 5x^2 + 7x^1 + 6x^0$.

В общем случае, любой блок информации в памяти компьютера можно считать полиномом, если рассматривать в качестве переменной x один бит.

CRC коды построены на рассмотрении битовой строки как строки коэффициентов полинома.

k - битовая строка - коэффициенты полинома степени k -1.

Самый левый бит строки - коэффициент при старшей степени.

Например, строка 110001 представляет полином $x^5 + x^4 + x^0$.

Назовем этот блок информационным полиномом и обозначим его A(x).

Для вычисления контрольного кода понадобится еще один полином, называемый порождающим полиномом.

Порождающий полином – это предварительно специальным образом подобранный полином, на который впоследствии будет делиться информационный полином для вычисления контрольного кода.

Этот полином является ключом циклического кода, обозначим его G(x).

Для получения контрольного кода информационный полином A(x) умножается на x^r в (что соответствует сдвигу на r разрядов влево), а затем делится на порождающий полином G(x). Частное – полином Q(x) – отбрасывается (не участвует в дальнейших вычислениях). Остаток от деления R(x) является контрольным полиномом.

Умножение информационного полинома на x^r реализуется тривиально - как дописывание после младшего разряда полинома r нулевых разрядов.

Пример 1.

Предположим, что контролируются данные вида 11010011. В качестве порождающего полинома выбрано число 10011 (r = 4), следовательно, контрольный код будет получен делением 110100110000 на 10011.

Для организации деления можно реализовать специальный алгоритм :

r + 1 старших разрядов информационного полинома складываются по модулю 2 с порождающим полиномом и в частное заносится 1. В дальнейшем, полином, полученный после сложения информационного полинома с порождающим полиномом, будем называть остатком информационного полинома. После сложения самый старший разряд остатка информационного полинома равен 0, поэтому сдвигаем на один разряд вправо. Если следующий разряд остатка

полинома равен нулю, то в частное заносится 0 и сдвигаем дальше, в противном случае снова производится сложение по модулю 2 группы разрядов длиной r + 1 остатка информационного полинома с порождающим полиномом. Последнее действие повторяется до тех пор, пока степень остатка не станет меньше степени порождающего полинома. Полученный остаток и будет являться контрольным кодом.

Замечание 1. ***Поскольку нас интересует только контрольный код - остаток от деления, частное вычислять и запоминать не надо.***

Замечание 2. ***Полиномиальная арифметика выполняется по модулю 2. Сложение и вычитание происходит без переноса разрядов.***

Так что обе эти операции эквивалентны операции XOR (eXclusive OR). Деление выполняется как обычно в двоичной системе счисления с той лишь разницей, что вычитание выполняется по модулю 2. Часто вычисления в соответствии с этими правилами называют CRC-арифметикой.

Для иллюстрации рассмотрим деление двоичных чисел:

```
110100110000    | 10011
10011                 11000111
 10010110000
 10011
    1110000
    10011
     111100
     10011
      11010
      10011
       1001   - остаток от деления, т.е.  R(x).
```

Очевидно, что число разрядов полинома R(x) определяется разрядностью порождающего полинома G(x).

Из теории следует, чем больше величина R(x), тем больше способность контрольного кода обнаруживать ошибки при хранении или передаче данных.
Выбор порождающего полинома – нетривиальная задача. Однако, большинство алгоритмов вычисления CRC использует заранее известные (и исследованные специалистами) полиномы.

Существуют международные стандарты на вид G(x), например:

CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$ (системы телекоммуникации)
Используется для 6-разрядных кодов.

CRC-8 Dallas = $x^8+x^5+x^4+1$ Poly : 0x31
Используется для 8-разрядных кодов.

CRC8 - CCITT = x^8+x^2+x+1 - полином 07h
Используется для 8-разрядных кодов.

CRC16- CCITT = $x^{16}+x^{12}+x^5+x^0$ = G(x) - полином 1021h
Используется для 18-разрядных кодов.

CRC-16 и CRC-CCITT ловят одиночные, двойные ошибки, групповые ошибки длины не более 16 и нечетное число изолированных ошибок с вероятностью 99,997%.

CRC32 IEEE = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$ - полином 04c11db7h

Этот полином используется в технологии локальных вычислительных сетей Ethernet.

CRC32 = $x^{32}+x^{31}+x^{30}+x^{29}+x^{27}+x^{26}+x^{24}+x^{23}+x^{21}+x^{20}+x^{19}+x^{15}+x^9+x^8+x^5$ - полином 0edb8832 0h

(обратим внимание, что этот полином – зеркальное отражение первого CRC32)

Этот полином используется различными архиваторами.

3. Использование CRC

Полиномиальные коды используются для проверки целостности данных и при хранении или передаче данных между компьютерами.

Идея состоит в том, чтобы добавить (хранить) контрольную сумму к контролируемому блоку данных. Впоследствии осуществляется сравнение хранящегося эталона и вновь вычисленной по тому же алгоритму контрольной суммы. В случае совпадения принимается решение о том, что целостность не была нарушена. Так, при передаче данных по сети отправитель и получатель заранее договариваются о конкретном порождающем полиноме G(x), у него коэффициенты при младшем члене и при старшем члене должны быть равны 1.

Пусть степень G(x) равна r. Для вычисления контрольной суммы блока из m бит надо, чтобы обязательно m > r. Контрольная сумма добавляется к блоку, рассматриваемому как полином A(x) так, чтобы передаваемый блок с контрольной суммой был кратен G(x). Когда получатель получает блок с контрольной суммой, он делит его на G(x). Если остаток не равен «0», то были ошибки при передаче.

Пример 2.

Вычисление CRC-кода

Описанный ниже алгоритм используется для вычисления двух разных CRC-кодов.

Последовательность вычислений следующая:

1. Нахождение логического исключающего ИЛИ между младшим битом начального значения CRC (00 или FF) и младшим битом данных.
2. Если результат равен 0, то:
   a. Сдвиг вправо **CRC.**
3. Если результат равен 1:
   a. Поиск нового значения CRC путем вычисления логического исключающего ИЛИ между **CRC**(полученным остатком) и полиномом CRC.
   b. Сдвиг вправо **CRC.**

с. Установка старшего бита **CRC** в 1.

4. Сдвиг вправо данных.

5. Повтор данной последовательности 8 раз для CRC8.

Данный алгоритм может использоваться для вычисления, как CRC8, CRC16 так и CRC32. Отличие состоит только в разрядности сдвигового регистра (8 разрядов для CRC8, 16 разрядов для CRC16 и 32 для CRC32) и значении полинома.

Значение полинома равно 18h для CRC8 и 4002h для CRC16, 0edb8832 0h для CRC32.

4. Задания и вопросы на защите:

4.1. На основе Примера 1. Привести 2-3 своих примера получения остатка в двоичной и полиномиальной арифметике.

4.2. Выполнить компьютерную реализацию прямого алгоритма вычисления CRC на основе Примера 2. Язык реализации – C, C++

4.3. Изучить самостоятельно табличные алгоритмы вычисления CRC и алгоритмы вычисления CRC для любой длины информационного кода /обратный алгоритм/. (Найти в Интернете).

4.4. Реализовать алгоритмы CRC8 и CRC16 для табличного метода и привести примеры получения контрольной суммы произвольного файла, приписать CRC к концу файла и поверить его программно на контрольную сумму.

5. Содержание отчета

5.1. Привести в отчете все преобразования, примеры, алгоритмы и программу получения и проверки контрольных CRC .

5.2. Выполнить программы с интерфейсом для пользователя.

ЛАБОРАТОРНАЯ РАБОТА №4

Варианты и приёмы защиты программного обеспечения от копирования.

Необходимо, чтобы контролирующая часть защищаемой программы (КЧЗП) "запомнила" свой компьютер и потом при запуске сравнивала имеющиеся характеристики с характеристиками "родного" компьютера. В случае их расхождения можно считать, что программа незаконно скопирована, и прервать ее выполнение. Для этого надо найти какие-то параметры, которые бы индивидуально характеризовали каждую вычислительную систему. На самом деле это весьма нетривиальная задача, поскольку открытая архитектура построения компьютеров подразумевает их обезличенность.

Рассмотрим, что все же можно предложить для КЧЗП в качестве характеристик, которые могли бы проверяться при работе защищаемой программы.

1. Физические дефекты винчестера

При работе жесткого диска (винчестера) возможно возникновение сбойных участков на магнитной поверхности. Такие дефектные места могут иметься даже на совершенно новом винчестере. Номера этих сбойных участков помещаются в FAT (их признак - код FF7). При инсталляции защищаемой программы на винчестер в ее контролирующую часть записываются их адреса. В процессе выполнения программы осуществляется сравнение адресов сбойных участков, записанных в КЧЗП и в FAT. В случае запуска незаконно скопированной программы будет обнаружено расхождение сравниваемых адресов (первые не будут составлять подмножество вторых) и произойдет выполнение аварийных действий, предусмотренных для такого случая.

Развитием этой идеи является метод, в котором некоторые исправные кластеры помечаются как сбойные, и в них помещается информация для КЧЗП. (При копировании такие кластеры не передаются.)

2. Дата создания BIOS

Можно попытаться сузить класс компьютеров, на которых возможно функционирование незаконно скопированной программы. Это достигается, например, путем введения проверки даты создания BIOS, которая записана в ПЗУ каждого компьютера. Эта дата заносится в КЧЗП, и в процессе выполнения защищаемой программы осуществляется сравнение дат создания BIOS, записанных в КЧЗП и ПЗУ компьютера. В случае если защищаемая программа будет незаконно скопирована и установлена на компьютер другой серии, то КЧЗП это обнаружит и будут выполнены аварийные действия.

Дата создания BIOS записана в ПЗУ по адресу FFFF:0005 и занимает 8 байтов.

3. Версия используемой OS

Так как версия операционной системы, с которой работает пользователь на данном компьютере, не меняется на протяжении достаточно длительного времени, то ее также можно использовать в качестве параметра для организации проверки в КЧЗП. Этот контроль желательно проводить в комплексе с другими методами защиты, например, дополнительно к проверке даты создания BIOS, что еще более суживает класс машин, на которых может работать защищаемая программа.

4. Серийный номер диска

При форматировании диска на него создается так называемый серийный номер, его несложно изменить с помощью программ типа DiskEdit.

5. Тип компьютера

Для обеспечения работы защищаемой программы только на компьютере одного клона надо, чтобы она могла определять его тип. Такая информация содержится в байте, расположенном по адресу FFFF:000E в ROM BIOS со следующей кодировкой:

PC - FF; XT - FE,FB; PCjr - FD; AT - FC; PS/2 - FC,FA,F8;

PC-совместимый - F9.

6. Конфигурация системы и типы составляющих ее устройств

В развитие идеи, изложенной выше, рассмотрим способ, при котором КЧЗП в качестве параметра использует всю конфигурацию системы. Конечно надо учитывать, что этот способ будет давать ошибки и отказывать в выполнении законной копии программы, если какие либо составляющие вычислительной системы будут изменены (например, отключен принтер), либо наоборот законная и незаконная копии программы будут работать на системах с одинаковой конфигурацией.

Компьютер может быть снабжен различными типами дисководов для гибких дисков и винчестеров. Рассмотрим с помощью каких средств можно получить информацию об их характеристиках кроме доступа к CMOS памяти.

Прерывание 21h DOS компьютера AT имеет функции 32h и 36h, связанные с определением характеристик установленных накопителей. Функция 36h сообщает текущие сведения о доступном пространстве на диске, номер которого загружается в регистр DL. Она возвращает:

- число секторов на один кластер в регистре AX;
- число незанятых кластеров - в BX;
- число байтов в одном секторе - в CX;
- число кластеров на диске - в DX.

Теперь легко узнать, каков объем диска, номер которого помещался в регистр DL. Для этого достаточно вычислить произведение значений регистров AX,CX,DX.

Функция 32h, позволяет получить таблицу с параметрами накопителя, номер которого загружен в регистр DL. Ее адрес будет содержаться в регистрах DX:BX. Подробное описание байтов этой таблицы можно найти в любом справочнике по прерываниям.

Состав установленного оборудования проверяется при загрузке, и результат проверки помещается в регистр статуса. Этот регистр занимает два байта, начиная с адреса 0040:0010, и в табл. 3 представлены значения его битов.

Для доступа к этому регистру кроме прямого обращения по адресу можно воспользоваться прерыванием 11h BIOS, которое возвращает два байта его значений в регистре AX.

В языке Turbo C имеется специальная функция BIOSEQUIP из библиотеки стандартных функций <bios.h>, которая возвращает целое число, описывающее оборудование, входящее в систему. Возвращаемое значение интерпретируется набором битовых полей, как представлено в табл. 4.

Таблица 3

Регистр состава установленного оборудования

БИТ	СОДЕРЖИМОЕ	ЗНАЧЕНИЕ
0	0/1	Нет/есть НГМД
1	0/1	Нет/есть математический сопроцессор 80x87
2-3	11	Оперативная память 64 Кбайта (в AT не используется и всегда равна 11)
4-5	11/01/10	Начальный видеорежим (монохромный/цветной 40*25/ цветной 80*25)
6-7	00/01/10/11	Число НГМД, если бит 0 =1 (соответственно 1,2,3,4)
8		XT/AT не используется (наличие микросхемы DMA)
9-11		Число адаптеров коммуникации RS232
12	1	Есть игровой адаптер (в AT не используется)
13		XT/AT не используется
14-15		Число присоединенных принтеров

Еще одно прерывание BIOS - 15h через функцию C0h позволяет получить адрес в ПЗУ, определяемый регистрами ES:BX, по которому находится табл. 4.

Таблица 4

БАЙТЫ	ЗНАЧЕНИЕ
0-1	Число байтов в таблице
2	Код модели компьютера (см. выше)
3	Различие между AT и XT/286 (подмодель)
4	Номер ревизии BIOS
5	80h - 3-й канал DMA, используется BIOS 40h - второй контроллер прерываний i8259 установлен 20h - таймер реального времени установлен 10h - int15h/AH=4Fh вызывается перед int 9h 8h - допустимо ожидание внешнего события 4h - расширение BIOS размещено в 640 Кбайтах 2h - шиной является Micro Channel вместо шины ISA 1h - резерв Замечание: 1/10/86 XT BIOS возвращает некорректное значение 5-го байта

Для определения типа дисплея надо проверить бит номер 1 байта, находящегося по адресу 0040:0087. Когда этот бит равен 1 - подсоединяется монохромный дисплей, а когда он равен 0 - цветной.

7. Получение инженерной информации жесткого диска

Контроллер IDE, SCSI имеет специальную команду выдачи информации о подключенном устройстве. Программа IdeInfo выдает блок 512 байт информации о жестком диске, если в системе есть контроллер IDE и жесткий диск. Информация содержит параметры диска и его серийный номер. Некоторые OS не дают доступ к этой информации.

Выполнить один пункт из Задания 1 и один пункт из Задания 2:

Задание 1. Защита от копирования.

Варианты заданий:

"Заразить" xx..x.exe файл лабораторной работы №1-2 так, чтобы он работал только на «*своей»* машине. Необходимо "повредить" таблицу настройки адресов и части программы для невозможности "выкусить" приписанную часть. Программа должна устойчиво работать на "*своей*" машине и не работать на соседней.

В качестве идентифицирующей информации использовать:

1. BIOS (осторожно с Shadow)
2. Информацию IDE диска(не работает в Win NT)
3. N сетевой карты(MAC адрес)
4. Быстродействие процессора
5. N видео карты
6. N процессора(через сброс)
7. Приписку за концом файла
8. Другие варианты на Вашу изобретательность …….

Задание 2. Защита программного продукта от несанкционированного использования.

Задача состоит в том, чтобы ввести в исполняемую программу дополнительный модуль, который ограничивает использование Вашего продукта и требует дополнительной информации, позволяющей «допустить к использованию» программы или пакета программ.

Варианты заданий:

1. Ограничение времени использования продукта ('trial/evaluation');
2. Ограничение функциональности продукта ('demo/crippled');
3. Регулярные напоминания о необходимости регистрации ('nag screens');
4. Запрос регистрационного кода ('regcode');
5. Генерация системных ошибок ('crash/GPF');

6. Сбор и передача персональных данных ('spyware');
7. Вредоносные действия ('malware');
8. Внесение ошибок в обработку данных и др.

ЛИТЕРАТУРА

1. Алферов А.П. и др. Основы криптографии: Учебное пособие. - М.: Гелиос АРВ, 2001.
2. Анин Б.Ю. Защита компьютерной информации. - СПб.: БХВ-Петербург, 2000.
3. Блэк У. Интернет: протоколы безопасности. Учебный курс / Пер. с англ. - СПб.: Питер, 2001.
4. Конеев И.Р., Беляев А.В. Информационная безопасность предприятия. - СПб.: БХВ-Петербург, 2003.
5. Лукацкий А.В. Обнаружение атак. - СПб.: БХВ-Петербург, 2001.
6. Мамаев М., Петренко С. Технологии защиты информации в Интернете: Специальный справочник. - СПб.: Питер, 2002.
7. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. Под ред. В.Ф. Шаньгина. - 2-е изд., перераб. и доп. - М.: Радио и связь, 2001.
8. Чмора А.Л. Современная прикладная криптография. - М.: Гелиос АРВ, 2001.
9. ГОСТ Р 34.10-2001. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электрронной подписи на базе ассиметричного криптографического алгоритма.
10. ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электрронной подписи на базе ассиметричного криптографического алгоритма.
11. ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования.
12. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.