Титулка

ЗМІСТ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

СЛОВАРЬ ТЕРМИНОВ

Открытый (исходный) текст — данные (не обязательно текстовые), передаваемые без использования криптографии.

Шифротекст, шифрованный (закрытый) текст — данные, полученные после применения криптосистемы.

Шифр, криптосистема — совокупность заранее оговоренных способов преобразования исходного секретного сообщения с целью его защиты.

Символ — это любой знак, в том числе буква, цифра или знак препинания.

Алфавит — конечное множество используемых для кодирования информации символов. Стандартный алфавит может быть изменён или дополнен символами. **Ключ** — параметр шифра, определяющий выбор конкретного преобразования данного текста. В современных шифрах криптографическая стойкость шифра целиком определяется секретностью ключа (принцип Керкгоффса).

Шифрование — процесс нормального применения криптографического преобразования открытого текста на основе алгоритма и ключа, в результате которого возникает шифрованный текст.

Расшифровывание — процесс нормального применения криптографического преобразования шифрованного текста в открытый.

Асимметричный шифр, двухключевой шифр, шифр с открытым ключом — шифр, в котором используются два ключа, шифрующий и расшифровывающий. При этом, зная лишь ключ зашифровывания, нельзя расшифровать сообщение, и наоборот.

Открытый ключ — тот из двух ключей асимметричной системы, который свободно распространяется. Шифрующий для секретной переписки и расшифровывающий — для электронной подписи.

Секретный ключ, закрытый ключ — тот из двух ключей асимметричной системы, который хранится в секрете. Криптоанализ — наука, изучающая математические методы нарушения конфиденциальности и целостности информации.

Система шифрования (шифрсистема) — это любая система, которую можно использовать для обратимого изменения текста сообщения с целью сделать его непонятным для всех, кроме адресата.

Криптостойкостью — это характеристика шифра, определяющая его стойкость к дешифрованию без знания ключа (т.е. способность противостоять криптоанализу).

Криптоаналитик — учёный, создающий и применяющий методы криптоанализа. Криптография и криптоанализ составляют криптологию, как единую науку о создании и взломе шифров (такое деление привнесено с запада, до этого в СССР и России не применялось специального деления).

Криптографическая атака — попытка криптоаналитика вызвать отклонения в атакуемой защищённой системе обмена информацией. Успешную криптографическую атаку называют взлом или вскрытие.

Дешифрование (дешифровка) — процесс извлечения открытого текста без знания криптографического ключа на основе известного шифрованного. Термин дешифрование обычно применяют по отношению к процессу криптоанализа шифротекста (криптоанализ сам по себе, вообще говоря, может заключаться и в анализе криптосистемы, а не только зашифрованного ею открытого сообщения).

Криптографическая стойкость — способность криптографического алгоритма противостоять криптоанализу.

Имитозащита — защита от навязывания ложной информации. Другими словами, текст остаётся открытым, но появляется возможность проверить, что его не изменяли ни случайно, ни намеренно. Имитозащита достигается обычно за счет включения в пакет передаваемых данных имитовставки.

Имитовставка — блок информации, применяемый для имитозащиты, зависящий от ключа и данных.

Электронная цифровая подпись(электронная подпись) — асимметричная имитовставка (ключ защиты отличается от ключа проверки). Другими словами, такая имитовставка, которую проверяющий не может подделать.

Центр сертификации — сторона, чья честность неоспорима, а открытый ключ широко известен. Электронная подпись центра сертификации подтверждает подлинность открытого ключа.

Хеш-функция — функция, которая преобразует сообщение произвольной длины в число («свёртку») фиксированной длины. Для криптографической хеш- функции (в отличие от хеш-функции общего назначения) сложно вычислить обратную и даже найти два сообщения с общей хеш-функцией.

ВВЕДЕНИЕ

Цель данных лабораторных работ состоит в предоставлении студентам возможности освоить основные принципы, методы, алгоритмы информационной безопасности. Исследовать и опробовать на практике различные методы защиты информации, а именно: сохранение целостности, конфиденциальности и доступности информации. при условии передачи через различные каналы связи.

Первая работа является подготовительной, где студен должен войти в курс дела.

Вторая работа показывает обязательность безопасность в наше время.

Третья работа предоставляет возможность рассмотреть алгоритмы с различных сторон (от построения схематической модели, до математического смысла).

*В связи с небольшим курсом приходиться пропускать некоторые части. Специалисту в системной инженерии не требуется создавать алгоритмы самостоятельно, так что мы можем пропустить аспект написания алгоритма или изучить их отдельно.

Четвёртая работа информирует о различных схемах шифрования. В ней мы знакомимся с различными способами работы блочных алгоритмов. (Использовать алгоритм можно любой)

Пятая работа показывает студенту о проблемах алгоритмов. В ней рассматривается свойства выбранного алгоритма и оценка данных свойств.

После пяти работ у студента должна быть система, которая способна шифровать/дешифровать данные двумя способами: потоковым(3ЛР), блочным(4ЛР). Так же оценка системы, её слабые стороны.

Шестая работа знакомит студента с ассиметричными алгоритмами и предлагает реализовать способ передачи *ключа шифровки* через небезопасный канал.

Седьмая работа является сбором всех предыдущих работ. В ней реализуется система, в которой: первично соединяется клиент-клиент через ассиметричную шифровку, после чего обмениваются ключам. Далее шифровка проходит через симметричное шифрование и данные передаются от участника А к участнику Б.

Восьмая работа является отдельной, она знакомит пользователя с цифровыми ключами. Дать понимание как они устроены и принцип их работы. Рассмотреть встроенные в систему подписи и создать свою ЭЦП.

1 МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

Тема: Методы защиты информации. Классификация криптосистем.

Цель: Изучить простые методы криптографической защиты информации, использовать полученные знания для сокрытия путём шифрования.

Теоретические ведомости

Появление новых информационных технологий и развитие мощных компьютерных систем хранения и обработки информации повысили уровни защиты информации и вызвали необходимость того, чтобы эффективность защиты информации росла вместе со сложностью архитектуры хранения данных. Постепенно защита информации становится обязательной: разрабатываются всевозможные документы по защите информации; формируются рекомендации; даже проводится ФЗ о защите информации, который рассматривает проблемы и задачи защиты информации, а также решает некоторые уникальные вопросы защиты информации.

Таким образом, угроза защиты информации сделала средства обеспечения информационной безопасности одной из обязательных характеристик информационной системы.

Рисунок 1.1 – Методы криптографического преобразования

1 Симметричные криптосистемы

1.1 Шифры перестановки

В шифрах средних веков часто использовались таблицы, с помощью которых выполнялись простые процедуры шифрования, основанные на перестановке букв в сообщении. Ключом в данном случае является размеры таблицы. Например, сообщение «Неясное становится ещё более непонятным» записывается в таблицу из 5 строк и 7 столбцов по столбцам.

Таблица 1.1

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	Ё	Е	Н	М

Для получения шифрованного сообщения текст считывается по строкам и группируется по 5 букв:

НОНСБ-НЯЕЕО-ЯОЕТЯ-СВЕЛП-НСТИЩ-ЕОЫНА-ТЕЕНМ

Несколько большей стойкостью к раскрытию обладает метод *одиночной перестановки* по ключу. Он отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы. Используя в качестве ключа слово - **ЛУНАТИК**, получим следующую таблицу:

Таблица 1.2 – Метод перестановки по ключу

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 левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с естественным порядком соответствующих букв ключа в алфавите. Если в ключе встретились бы одинаковые буквы, они бы нумеровались слева направо. Получается шифровка:

СНЯНН-БОЯЕТ-ЕООЕЕ-ПНЯВЛ-СЩОЫС-ИЕТЕН-МНТЕА

Для обеспечения дополнительной скрытности можно повторно шифровать сообщение, которое уже было зашифровано. Для этого размер второй таблицы подбирают так, чтобы длины её строк и столбцов отличались от длин строк и столбцов первой таблицы. Лучше всего, если они будут взаимно простыми.

Кроме алгоритмов одиночных перестановок применяются алгоритмы двойных перестановок. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При расшифровке порядок перестановок был обратный. Пример данного метода шифрования показан в таблице

Таблица 1.3 – Метод перестановки по ключу

	2	**4**	**1**	**3**
4	П	Р	И	Е
1	З	Ж	А	Ю
2	-	Ш	Е	С
3	Т	О	Г	О

	1	**2**	**3**	**4**
4	И	П	Е	Р
1	А	З	Ю	Ж
2	Е	-	С	Ш
3	Г	Т	О	О

	1	2	3	4
1	А	З	Ю	Ж
2	Е	-	С	Ш
3	Г	Т	О	О
4	И	П	Е	Р

Ключом к шифру служат номера столбцов 2413 и номера строк 4123 исходной таблицы. В результате перестановки получена шифровка:

АЗЮЖЕ_СШГТООИПЕР

Число вариантов двойной перестановки достаточно быстро возрастает с увеличением размера таблицы: для таблицы 3x3 их 36, для 4x4 их 576, а для 5x5 их 14400.

В средние века для шифрования применялись и магические квадраты. Магическими квадратами называются квадратные таблицы с вписанными в их клетки последовательными натуральными числами, начиная с единицы, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число. Для шифрования необходимо вписать исходный текст по приведённой в квадрате нумерации и затем переписать содержимое таблицы по строкам.

Таблица 1.4 – Исходный текст с идентификаторами

П	Р	И	Е	З	Ж	А	Ю	_	Ш	Е	С	Т	О	Г	О
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Таблица 1.5 – Магический квадрат

16	3	2	13		О	И	Р	Т
5	10	11	8	→	З	Ш	Е	Ю
9	6	7	12	←	-	Ж	А	С
4	15	14	1		Е	Г	О	П

В результате получается шифротекст, сформированный благодаря перестановке букв исходного сообщения.

Число магических квадратов очень резко возрастает с увеличением размера его сторон:

– для таблицы $3 \times 3 \Rightarrow 1$ существует только один квадрат;

– для таблицы $4 \times 4 \Rightarrow 880$;

– для таблицы $5 \times 5 \Rightarrow 250000$.

2 Шифры простой замены

2.1 Система шифрования Цезаря.

Основан на замене каждой буквы сообщения на другую букву того же алфавита, путем смещения от исходной буквы на *K* букв. Известная фраза

Юлия Цезаря `VENI VINI VICI` – пришел, увидел, победил, зашифрованная с помощью данного метода, преобразуется в `SBKF SFAF SFZF` (при смещении на 4 символа).

Греческим писателем Полибием за 100 лет до н.э. был изобретён так называемый полибианский квадрат размером 5 х 5, заполненный алфавитом в случайном порядке. Греческий алфавит имеет 24 буквы, а 25-м символом является пробел. Для шифрования на квадрате находили букву текста и записывали в шифротекст букву, расположенную ниже её в том же столбце. Если буква оказывалась в нижней строке таблицы, то брали верхнюю букву из того же столбца.

3 Шифры сложной замены

3.1 Шифр Гронсфельда

Это модификация шифра Цезаря с использованием числовым ключом. Для этого под буквами сообщения записывают цифры числового ключа. Если ключ короче сообщения, то его запись циклически повторяют. Шифротекст получают примерно так же, как в шифре Цезаря, но отсчитывают не третью букву по алфавиту (как в шифре Цезаря), а ту, которая смещена по алфавиту на соответствующую цифру ключа.

1) Пусть в качестве ключа используется группа из трех цифр – 314.
2) Тогда Сообщение СОВЕРШЕННО СЕКРЕТНО.
3) Ключ 3143143143143143143.
4) Шифровка ФПЖИСЬИОССАХИЛФИУСС.

В шифрах *многоалфавитной замены* для шифрования каждого символа исходного сообщения применяется свой шифр простой замены (свой алфавит).
В компьютере операция шифрования соответствует сложению кодов ASCII символов сообщения и ключа по модулю 256.

Задание

В соответствии с вашим вариантом из табл.1.6 зашифровать текст используя методы криптографической защиты представленные ниже. Регистр должен быть учтён.

1) Шифры перестановки:

a) метод перестановки по ключу;
b) алгоритм двойной перестановки;
c) магические квадраты.

2) Шифры замены:

a) шифр Цезаря;
b) Аффиный шифр;
c) шифр Виженера;
d) шифра Плейфера.

3) Выполнить шифрование методом гаммирования.

Записать результаты шифрования в отчёт, сравнить методы, выбрать оптимальный для заданной фразы и аргументировать свой выбор.

Вопросы для самоконтроля

1) Криптография и её роль в обществе.
2) Объяснить цель и задачи криптографии.
3) Пояснить какие бывают криптографические методы.
4) Что такое шифрование?
5) Как происходит процесс шифровки/дешифровки сообщения?
6) Как наложение гаммы влияет на исходный текст?

Таблиця 1.6 – Список фраз для шифрования

1	6 x 6	небольшое сообщение для тестирования
2	3 x 13	В атмосфере происходит около 1800 гроз.
3	7 x 4	федеральное законодательство
4	4 x 6	Международные стандарты;
5	3 x 13	самая дорогая пицца в мире стоит $1000.
6	4 x 9	применение информационных технологий
7	3 x 12	административный уровень секретности
8	3 x 11	83% младших братьев выше старших
9	3 x 11	обеспечение доступа к информации
10	10 x 4	Индонезия расположена на 17508 островах.
11	4 x 7	у рыбы сарган зеленые кости.
12	8 x 4	язык хамелеона длиннее его тела
13	6 x 4	защищенность информации.
14	5 x 6	в озеро Байкал впадает 336 рек
15	3 x 10	гоночный болид едет по трассе.
16	8 x 3	опасность ’’открывается”
17	4 x 8	процесс обеспечения целостности
18	4 x 9	рекомендация использования терминов
19	5 x 6	обеспечивающее ее формирование
20	5 x 5	общегосударственный орган
21	8 x 4	технических средств ее передачи
22	3 x 11	ворон и ворона — два разных вида.
23	5 x 6	наибольший ущерб субъектам ИБ
24	9 x 3	информационная безопасность
25	8 x 4	ущерб при сервисном обслуживании
26	5 x 7	свойство аутентичности пользователя
27	8 x 4	данные были действия выполнены?!
28	6 x 6	законодательный уровень безопасности
29	8 x 4	из множества потенциально угроз
30	4 x 9	Защита процессов, процедур, программ

2 ОСНОВЫ КРИПТОАНАЛИЗА

Тема: Основы криптоанализа. Частотный анализ.

Цель: Изучить основы классического криптоанализа, провести частотный анализ и расшифровать текст.

Теоретические ведомости

Моноалфавитный подстановочный шифр – шифр, в котором каждой букве исходного алфавита поставлена в соответствие одна буква шифра.

Например, возьмем слово **КУКУРУЗА**. Пусть букве **К** текста соответствует буква А шифра, букве **У** текста соответствует буква Б шифра, букве **Р** текста соответствует буква В шифра, букве **З** текста соответствует буква Г шифра, букве **А** текста соответствует буква Д шифра. После подстановки букв шифра вместо букв исходного теста слово **КУКУРУЗА** в зашифрованном виде будет выглядеть как АБАБВВГД. Недостатком подобного шифрования является то, что, если какая-то буква встречается в исходном тексте чаще всего, то и соответствующая ей буква шифра в зашифрованном тексте также встречается чаще всего. На рисунке 2.1 приведены частоты встречаемости букв в английском тексте.

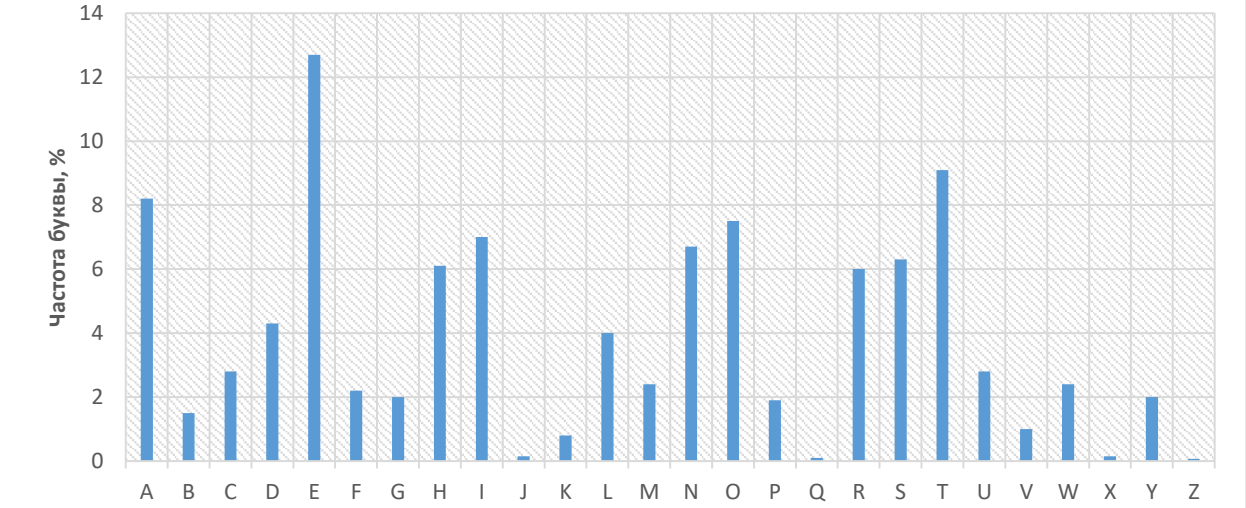

Рисунок 2.1 –Частота использования букв в Английском языке

Для примера мы взяли текст из ресурса Wikipedia[14]. Данный текст, предложенный для рассмотрения принципа частотной атаки, с подробным руководством можете ознакомиться на соответствующем ресурсе.

```
LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFEMVEWHKVSTYLXZI
XLIKIIXPIJVSZEYPERRGERIMWQLMGLMXQERIWGPSRIHMXQEREKIETXM
JTPRGEVEKEITREWHEXXLEXXMZITWAWSQWXSWEXTVEPMRXRSJGSTVRIE
YVIEXCVMUIMWERGMIWXMJMGCSMWXSJOMIQXLIVIQIVIXQSVSTWHKPEG
ARCSXRWIEVSWIIBXVIZMXFSJXLIKEGAEWHEPSWYSWIWIEVXLISXLIVX
LIRGEPIRQIVIIBGIIHMWYPFLEVHEWHYPSRRFQMXLEPPXLIECCIEVEWG
ISJKTVWMRLIHYSPHXLIQIMYLXSJXLIMWRIGXQEROIVFVIZEVAEKPIEW
HXEAMWYEPPXLMWYRMWXSGSWRMHIVEXMSWMGSTPHLEVHPFKPEZINTCMI
VJSVLMRSCMWMSWVIRCIGXMWYMX
```

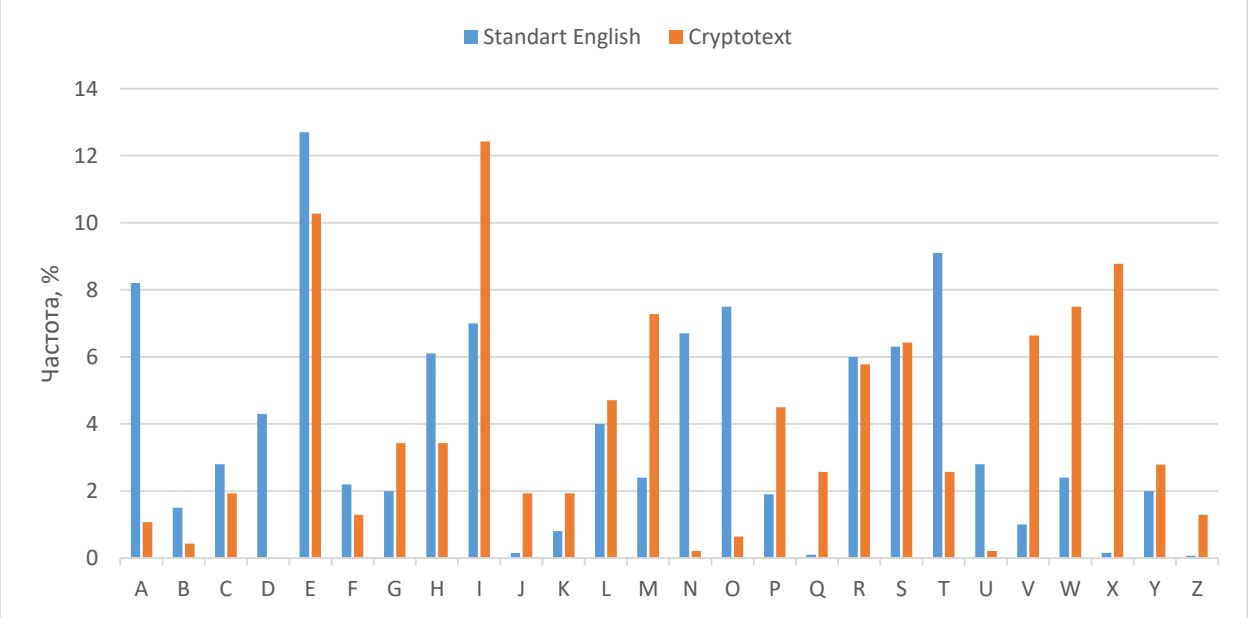

Рисунок 2.2 – Частотный анализ шифрованного текста

Как видно из графика на рис. 2.2 частоты использования символов неясны. Прямой корреляции между символами не наблюдается. Всегда нужно помнить, что частотный анализ не даёт нам полной картины. В каждом тексте соотношение символов будет различное. Так же данный вид взлома будет работать с разными показателями качества для разных языков.

На данном примере мы видим, что по частоте использования в стандарте английского языка преобладает буква **E**. В нашем зашифрованном тексте это может быть буква **I**. Так же можем предположить, что **X** является **T**. Таким образом мы можем продолжить сравнение и получить график на рис.2.3.

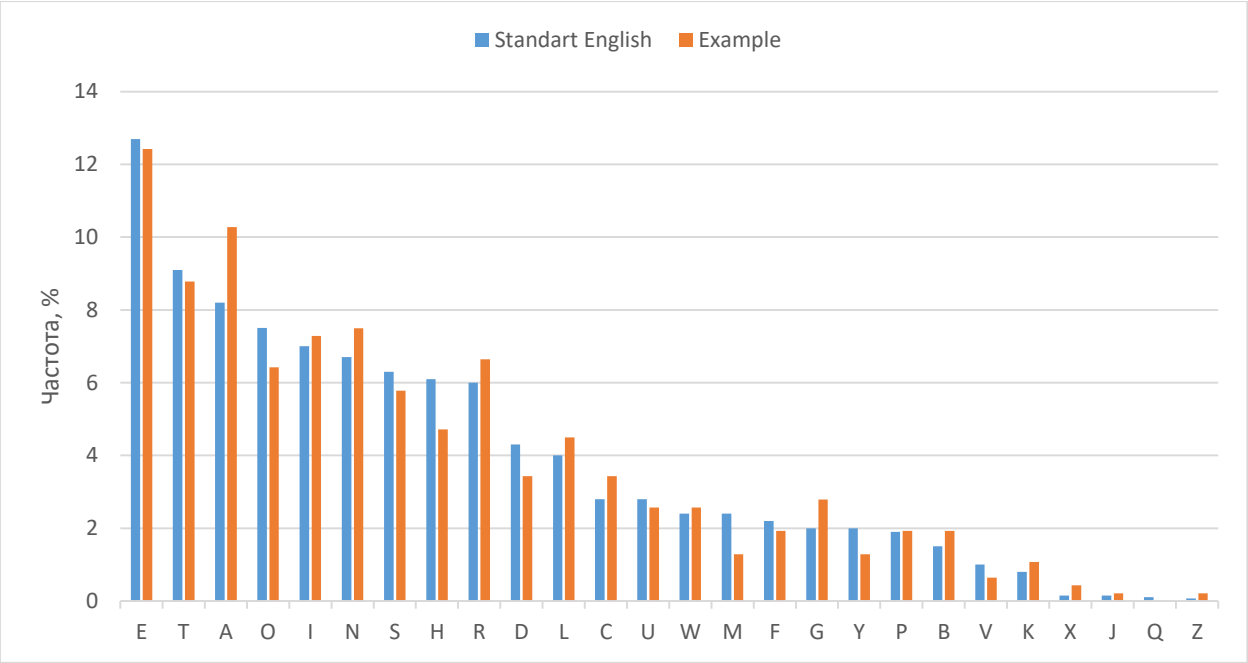

Рисунок 2.3 – Результат сравнения частот стандарта с расшифрованном текстом

```
Hereupon Legrand arose, with a grave and stately air, and
brought me the beetle from a glass case in which it was
enclosed. It was a beautiful scarabaeus, and, at that
time, unknown to naturalists—of course a great prize in
a scientific point of view. There were two round black
spots near one extremity of the back, and a long one near
the other. The scales were exceedingly hard and glossy,
with all the appearance of burnished gold. The weight of
the insect was very remarkable, and, taking all things
into consideration, I could hardly blame Jupiter for his
opinion respecting it.
```

Частотность существенно зависит, однако, не только от длины текста, но и от его характера. Например, в техническом тексте обычно редкая буква **F** может появляться гораздо чаще. Поэтому для надёжного определения средней частоты букв желательно иметь набор различных текстов[3].

Для маскировки частот появления тех или иных букв в тексте используется ***полиалфавитный шифр***. В котором шифрование очередного символа открытого текста согласно некоторому правилу.

Задание

Проработав материал изучить основные виды взлома. Ознакомившись с теоретическим материалом расшифровать предоставленный текст используя частотный анализ для определения исходного текста. Варианты для задания соответствуют первому номеру в папке «Материалы/ЛР2».

Способы выполнения лабораторной работы:

a) используя предоставленное ПО;

b) использовать альтернативное ПО (без функции автоматической расшифровки текста);

c) ручным способом.

В результате выполнения лабораторной работы написать отчёт, в котором должны быть предоставлены:

– отрывок шифрованного(исходного) текста;

– данные частотного анализа, предоставлены графики;

– описаны основные этапы выполнение работы;

– предоставлена таблица переменных для замены символов;

– отрывок расшифрованного текста;

– вывод к выполненной работе;

Ход работы

1) Откройте приложение. Для лучшей обработки файла включена «Подсветка изменений» и «Смена регистров». Их можно выключить во вкладке «Параметры» (рис. 2.4).

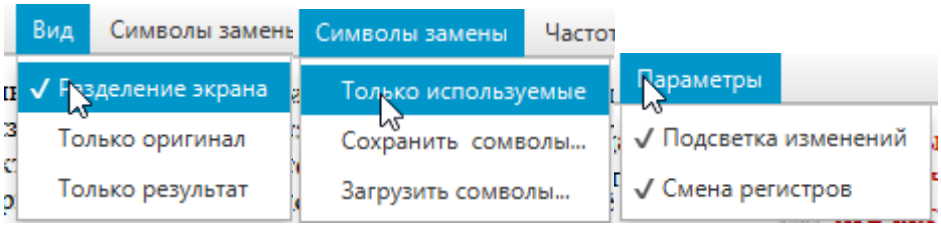

Рисунок 2.4 – Параметры для настройки

2) Выберите меню «Файл», далее пункт «Открыть шифр…» и выберите файл с текстом варианта. После открытия файла появятся вся нужная информация.

Приложение разделено на две области:

a) Текстовая (в ней предоставлен исходный/обработанный текст);

b) Символьная (в ней предоставлены символы замены).

Работа с программой предполагает замену определённых символов на их шифрованные аналоги. В результате данной обработки мы получим исходный текст.

Для корректной замены нужно выполнить следующие шаги (рис.2.5):

1. Выбрать символ для замены.
2. Предоставить заменяющий символ в поле и подтвердить изменение нажав клавишу «Enter».
3. Включить данный символ в активный режим.
4. Нажать кнопку обновить текст.

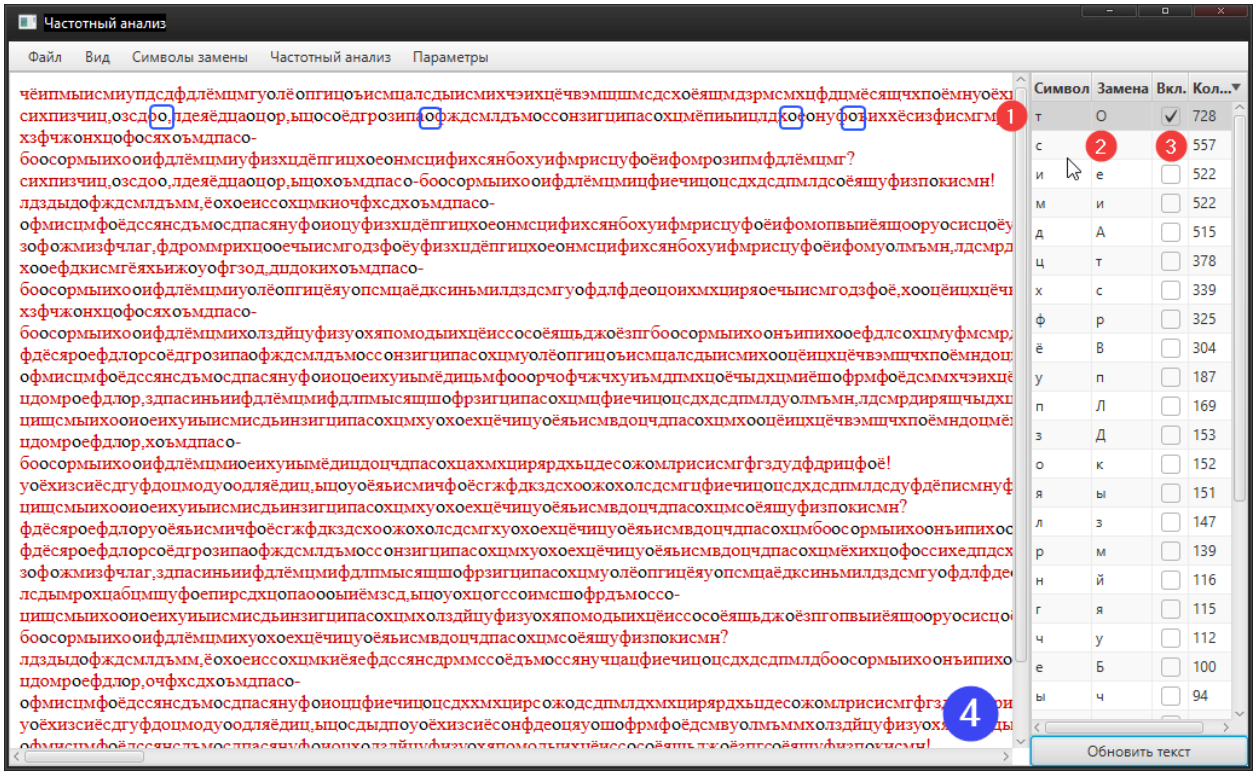

Рисунок 2.5 – Пример замены символа в тексте

3) Проведите частотный анализ открыв соответствующую вкладку в меню (рис.2.6). Запишите в отчёт свои предположения на счёт частотных совпадений.

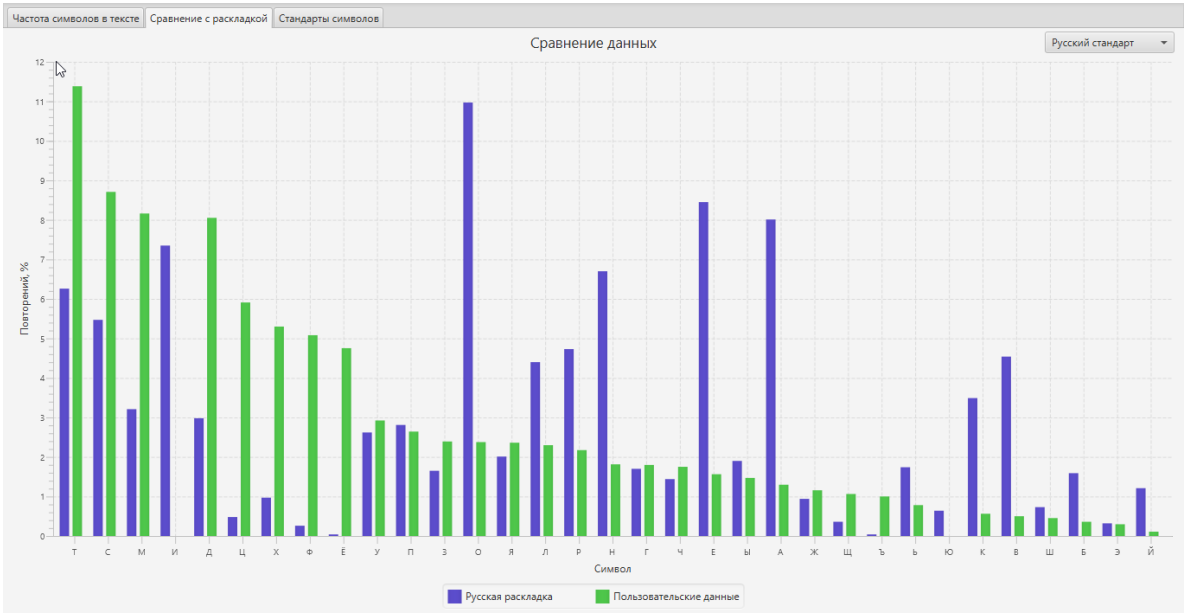

Рисунок 2.6 – Визуальное сравнение частот символов текста и стандарта

4) В результате расшифровки текста (рис.2.7) сохранить его через пункт меню «Файл»-«Сохранить результат…». Символы замены возможно сохранить через «Символы замены»-«Сохранить символы…».

5) Записать данные в отчёт, включая таблицу символов и расшифрованный текст.

Рисунок 2.7 – Результат расшифровки текста

Вопросы для самоконтроля

1. **Вот тут я хрен знает что писать. Напишу в самом конце для всех работ**

3 ПОТОКОВЫЕ ШИФРЫ. ГАММИРОВАНИЕ

Тема: Изучение потоковых шифров. Моделирование работы скремблера.

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

Теоретические ведомости

Криптография представляет собой совокупность методов преобразования данных, направленных на то, чтобы сделать эти данные бесполезными для злоумышленника. Такие преобразования позволяют решить два главных вопроса, касающихся безопасности информации:

– защиту конфиденциальности;

– защиту целостности.

Проблемы защиты конфиденциальности и целостности информации тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой.

1 Виды криптосистем

Для многих обывателей термин «криптография» означает что-то загадочное и таинственное. Однако в настоящее время различные виды шифрования можно встретить буквально везде – это и простые кодовые замки на дипломатах, и многоуровневые системы защиты секретных файлов. Люди сталкиваются с ней, когда вставляют в банкомат карточку, совершают денежные переводы, покупают через интернет товары, общаются по мессенджерам, отправляют письма на электронную почту. Любые дела, связанные с информацией, так или иначе имеют отношение к криптографии. Но, несмотря на всё

многообразие сфер применения, в настоящее время существует всего несколько способов шифрования. Все эти методы криптографии относятся к двум видам криптографических систем:

– симметричным (с секретным ключом)

– ассиметричным (с открытым ключом).

Симметричные системы позволяют шифровать и расшифровывать информацию с помощью одного и того же ключа. Расшифровать криптографическую систему секретного ключа невозможно, если дешифровщик не обладает секретным ключом.

Стойкость системы целиком зависит от внутренней структуры генератора ключевой последовательности. Если генератор выдаёт последовательность с небольшим периодом, то стойкость системы будет невелика. Напротив, если генератор будет выдавать бесконечную последовательность истинно случайных бит, то мы получим *«ленту однократного использования»* с идеальной стойкостью.

Реальная стойкость потоковых шифров лежит где-то посредине между стойкостью простой моноалфавитной подстановки и «ленты однократного использования». Генератор ключевой последовательности выдаёт поток битов, который выглядит случайным, но в действительности является детерминированным и может быть в точности воспроизведен на приемной стороне. Чем больше генерируемый поток похож на случайный, тем больше усилий потребуется от криптоаналитика для взлома шифра.

2 Гаммирование

Простейшей и в то же время наиболее надёжной из всех схем шифрования является так называемая *лента однократного использования* (рис.**Ошибка! Источник ссылки не найден.**), изобретение, которое чаще всего связывают с именем Г.С. Вернама.

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

С точки зрения теории криптоанализа, метод шифрования случайной однократной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым *«однократное гаммирование»*).

Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение – информация о вскрытом участке гаммы не даёт информации об остальных её частях[7].

Допустим, в тайной деловой переписке используется метод однократного наложения гаммы на открытый текст.

Наложение гаммы – это выполнение операции сложения по модулю 2 её элементов с элементами открытого текста. Эта операция в языке программирования C++ обозначается знаком ^, а в математике – знаком $\oplus$.

Стандартные операции над битами:

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,1 \oplus 0 = 1,\ 1 \oplus 1 = 0$$

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

Режим шифрования однократного гаммирования реализуется следующим образом:

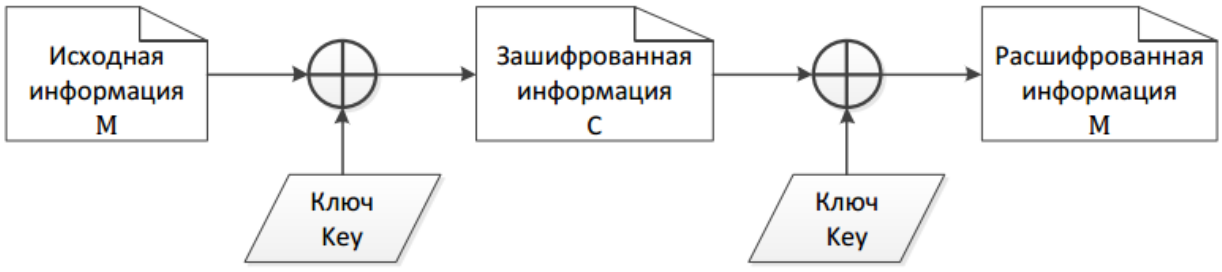

Рисунок 3.1 – Схема однократного гаммирования Вернама

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

$$C_i = P_i \oplus Key_i, \tag{3.1}$$

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

Задача нахождения ключа по известному шифротексту и открытому тексту может быть решена, исходя из (3.1). Для этого нужно обе части равенства сложить с по модулю 2:

$$\blacksquare_i \oplus P_i = P_i \oplus Key_i \oplus P_i = Key_i, \tag{3.2}$$

$$Key_i = C_i \oplus P_i. \tag{3.3}$$

Таким образом получаются формулы для решения обеих поставленных задач. Т.к. открытый текст представлен в символьном виде, а ключ – в своём шестнадцатеричном представлении, то в соответствие с таблицей ASCII-кодов можно представить ключ в символьном виде. Тогда уже будут возможны операции (3.1), (3.3), необходимые для решения поставленных задач.

К. Шенноном было доказано, что если ключ является фрагментом истинно случайной двоичной последовательности с равномерным законом распределением, причём его длина равна длине исходного сообщения, и используется этот ключ только один раз, после чего уничтожается, то такой шифр является *абсолютно стойким*, даже если криптоаналитик располагает неограниченным ресурсом времени и неограниченным набором вычислительных ре-

сурсов. Действительно, противнику известно только зашифрованное сообщение C , при этом все различные ключевые последовательности Key возможны и равновероятны, а значит, возможны и любые сообщения P , т.е. криптоалгоритм не даёт никакой информации об открытом тексте [5].

Необходимые и достаточные условия абсолютной стойкости шифра:

– полная случайность ключа;

– равенство длин ключа и открытого текста;

– однократное использование ключа.

2.1 Пример шифра абсолютной стойкости

В заданном примере изображено как абоненты переписки Алиса и Боб передают зашифрованное сообщение. В их переписку вмешалась Ева, которая прослушивает все данные, но не имеет ключей для дешифровки.

1) Алиса пишет сообщение:

```
Текст:   Буду в 9:00
  HEX:   D0 91 D1 83 D0 B4 D1 83 20 D0 B2 20 39 3A 30 30
```

Мы можем изображать текст в шестнадцатеричном формате(HEX), тогда английская буква будет соответствовать паре, а русская четырём символам.

```
Ключ:    Ключ Message
 HEX:    D0 9A D0 BB D1 8E D1 87 20 4D 65 73 73 61 67 65
```

2) Шифрует его с помощью ленты однократного использования.

```
Ключ:    8:
```

```
SJ[WU
Шифр:    00 0B 01 38 01 3A 00 04 00 9D D7 53 4A 5B 57 55
```

3) Отправляет через канал связи шифротекст.

4) Боб прослушивая канал связи считывает шифротекст:

```
Шифр:    00 0B 01 38 01 3A 00 04 00 9D D7 53 4A 5B 57 55
```

5) Подставляя верную гамму:

```
Гамма:   Ключ Message
  HEX:   D0 9A D0 BB D1 8E D1 87 20 4D 65 73 73 61 67 65
```

6) В результате Боб получает расшифрованное сообщение Алисы:

```
  HEX:   D0 91 D1 83 D0 B4 D1 83 20 D0 B2 20 39 3A 30 30
Текст:   Буду в 21:00
```

Предположим, что в связь вмешалась Ева. Она может только прослушивать канал, не изменяя данные. Как только Алиса отправил данные Бобу Ева получила зашифрованный текст:

```
Шифр:    00 0B 01 38 01 3A 00 04 00 9D D7 53 4A 5B 57 55
```

Но не зная верной гаммы она не сможет расшифровать текст. Предположим, что она нашла один из вариантов ключей:

```
Ключ:    Ключ Mes}age
 HEX:    D0 9A D0 BB D1 8E D1 87 20 4D 65 73 7d 61 67 65
```

В результате чего произвела дешифровку текста:

```
  HEX:   D0 91 D1 83 D0 B4 D1 83 20 D0 B2 20 37 3A 30 30
Текст:   Буду в 7:00
```

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

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

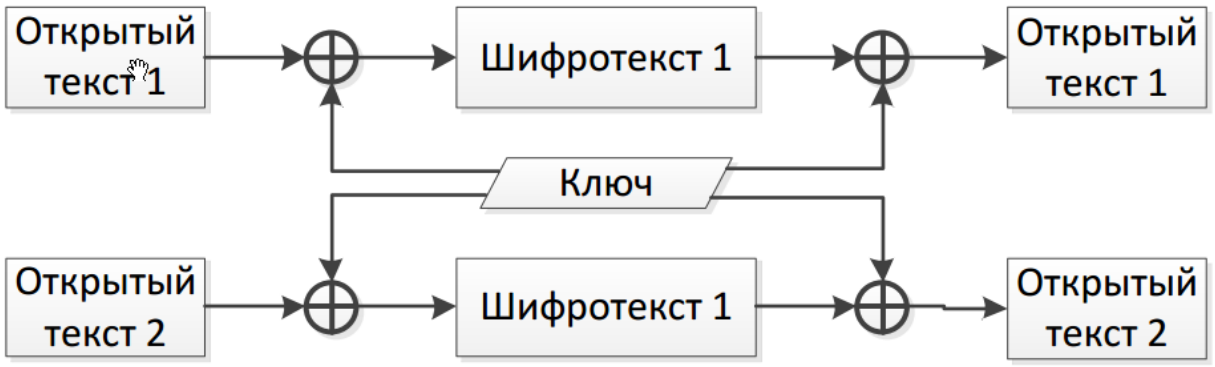

Рисунок 3.2 – Общая схема шифрования двух различных текстов одним ключом

С помощью формул режима однократного гаммирования получаются шифротексты обеих телеграмм:

$$Y_1 = P_1 \oplus K,\ Y_2 = P_2 \oplus K \tag{3.4}$$

Задача нахождения открытого текста по известному шифротексту двух телеграмм, зашифрованных одним ключом, может быть решена, используя (3.4). Для этого нужно сложить по модулю 2 оба равенства (3.4). Учитывая такие свойства операции XOR, как

$$1 \oplus 1 = 0, 1 \oplus 0 = 1, \tag{3.5}$$

можно получить

$$Y_1 \oplus Y_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2. \tag{3.6}$$

Предположим, что одна из телеграмм является "рыбой", т.е. имеет фиксированный формат, в который вписываются значения полей, и злоумышленнику доподлинно известен этот формат. Тогда он получает достаточно много пар $Y_1 \oplus Y_2$ (известен вид обоих шифротекстов) и, предположим, P_1 . Тогда с учётом(3.5) получается:

$$Y_1 \oplus Y_2 \oplus P_1 = P_1 \oplus P_2 \oplus P_1 = P_2. \tag{3.7}$$

Таким образом, злоумышленник получает возможность определить те символы сообщения P_2, которые находятся на позициях известной "рыбы" сообщения P_1. Догадываясь по логике сообщения P_2, злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения P_2. Затем он может использовать (3.7), вместо P_1 подставляя ранее определённые символы сообщения P_2. И так далее. Действуя подобным образом, злоумышленник если даже не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

3 Скремблер

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

Рассмотрим сдвиговый регистр с обратной связью (Linear Feedback Shift Register, сокращённо LFSR) – логическое устройство, схема которого показана на рисунке3.3.

Сдвиговый регистр представляет собой последовательность бит. Количество бит определяется длиной сдвигового регистра. Если длина равна п бит, то регистр называется n-битным сдвиговым регистром. Всякий раз, когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо на 1 позицию. Новый крайний левый бит является функцией всех остальных битов регистра. На выходе сдвигового регистра оказывается младший значащий бит [5].

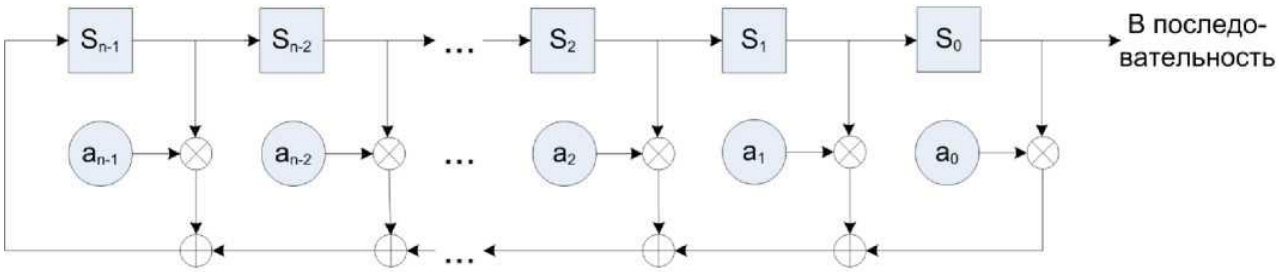

Рисунок 3.3 – Схема LFSR

LFSR состоит из *n* ячеек памяти, двоичные состояния которых в моменты времени $t = 0, 1, \ldots$ характеризуются значениями $S_0(t), S_1(t), \ldots, S_{n-1}(t) \in A = \{0, 1\}$.Выходы ячеек памяти связаны не только последовательно друг с другом, но и с сумматорами 0 в соответствии с коэффициентами передачи $a_0(t), a_1(t), \ldots, a_{n-1}(t) \in A$: если $a_i = 1$, то значение $S_i \ t$

i-й ячейки передаётся на один из входов *i*-го сумматора; если же $a_i = 0$, то такая передача отсутствует. Обычно коэффициенты передачи задаются с помощью полинома:

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

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

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

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

Текущие значения нулевой ячейки регистра используются в качестве элементов порождаемой LFSR двоичной псевдослучайной последовательности $s_y = S_0 \ t$ (см. Рисунок 3.3).

Данная последовательность извлеченных бит должна обладать тремя свойствами:

Сбалансированность:

для каждого интервала последовательности количество двоичных единиц должно отличаться от числа двоичных нулей не больше, чем на несколько процентов от их общего количества на интервале.

Цикличность:

непрерывную последовательность одинаковых двоичных чисел называют циклом. Появление иной двоичной цифры автоматически начинает новый цикл. Длина цикла равна количеству одинаковых цифр в нём. Необходимо, чтобы половина всех «полосок» (подряд идущих идентичных компонентов последовательности) имела длину 1, одна четвертая - длину 2, одна восьмая - длину 3, и т.д.

Корреляция:

если часть последовательности и её циклично сдвинутая копия поэлементно сравниваются, желательно, чтобы число совпадений отличалось от числа несовпадений не более, чем на несколько процентов от длины последовательности.

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

Чтобы построить *N*-разрядный скремблер, создающий последовательность наибольшей длины, пользуются примитивными многочленами. **Примитивный (базовый) многочлен степени** *n* по модулю 2 – это неприводимый многочлен, который является делителем $x^{2^n-1} + 1$, но не является делителем

$x^d + 1$ для всех *d*, на которые делится $2^n - 1$. **Неприводимый многочлен степени** (*n*) нельзя представить в виде умножения никаких других многочленов, кроме него самого и единичного.

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

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

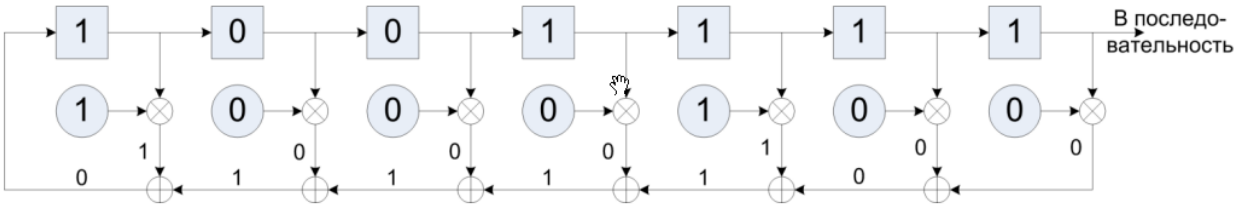

Рисунок 3.4 – Схема LFSR для многочлена $x^7 + x^6 + x^2$

Код для этого LFSR на языке C выглядит следующим образом:

```
1  int LFSR ()
2  {
3  /* Начальное состояние */
4  static unsigned long ShiftRegister = 79;
5  ShiftRegister =
6   ((((ShiftRegister >> 6) ^ (ShiftRegister >> 2)) & 0x01) <<
7  6) | (ShiftRegister >> 1);
8  return ShiftRegister & 0x01;
9  }
```

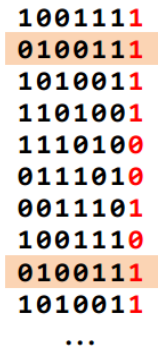

Рисунок 3.5 – Изменение состояния скремблера $x^7 + x^6 + x^2$

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

Задание

Часть первая.

Не предполагает использование дополнительного ПО, выполнение заданий математического расчёта должны быть выполнены вручную:

1) На основе вариантов из табл. 3.1 определить полином скремблера и его начальное значение.

2) Составить схему скремблера.

3) Составить математическую модель работы со скремблера для данных размером 16-бит.

4) Продемонстрировать работу скремблера из прошлого пункта на примере случайной последовательности.

Часть вторая.

Используя любой язык программирования, будь то C++, Java, C#, JS, Matlab или модель в Simulink. Реализовать приложение для шифровки/дешифровки сообщений:

1) Шифруемый текст(из табл. 1.6) должен храниться в файле.

2) Ключ шифрования должен задаваться случайным образом. Так же предусмотреть возможность ручного изменения ключа.

3) Зашифрованный текст должен сохраняться в один файл, а использовавшийся при шифровании ключ – отображаться на экране или сохраниться в отдельный файл.

4) Отобразить время шифрования скремблера.

5) Исследовать последовательность на свойства сбалансированности, цикличности, корреляции.

Отчёт должен включать в себя: Данные варианта, схема скремблера, математическая модель, демонстрация работы, реализация алгоритма шифрования, а также результаты анализа свойств и выводы к работе.

Варианты

Таблица 3.1 – Варианты скремблеров

Вариант	**Скремблеры**	
0	$x^8 + x^7 + x^6 + x^3 + x^2 + 1$	$x^8 + x^5 + x^3 + x^2 + 1$
1	$x^9 + x^3 + 1$	$x^9 + x^4 + 1$
2	$x^{10} + x^5 + x^4 + x^2 + 1$	$x^{10} + x^7 + 1$
3	$x^5 + x^4 + x^2 + 1$	$x^5 + x^2 + 1$
4	$x^{11} + x^5 + x^2 + 1$	$x^{11} + x^2 + 1$
5	$x^7 + x^5 + x^2 + 1$	$x^7 + x + 1$
6	$x^{12} + x^7 + x^3 + x + 1$	$x^{12} + x^6 + x^4 + x + 1$
7	$x^8 + x^6 + x^2 + 1$	$x^8 + x^4 + x^3 + x^2 + 1$
8	$x^{11} + x^3 + x^2 + 1$	$x^{11} + x^{10} + x^9 + x^2 + 1$
9	$x^6 + x^5 + x + 1$	$x^6 + x + 1$

Вопросы для самоконтроля

1) Преимущества и недостатки однократного гаммирования.

2) Почему размерность открытого текста должна совпадать с ключом?

3) Как по открытому тексту и шифротексту получить ключ?

4) Необходимые и достаточные условия абсолютной стойкости шифра?

5) Как, зная текст одного из сообщений (P_1 или P_2), определить другое, не пытаясь определить ключ?

6) Преимущества и недостатки использования скремблера.

7) Свойства, которыми должна обладать псевдослучайная последовательность, генерируемая скремблером.

4 СИММЕТРИЧНОЕ ШИФРОВАНИЕ

Цель: Изучить стандарты алгоритмов DES и ГОСТ симметричного шифрования. Получить практические навыки в реализации шифрования симметричной криптосистемы.

Теоретические ведомости

Это функция шифрования, которая применяется к блокам текста фиксированной длины. Текущее поколение блочных шифров работает с блоками текста длиной 256 бит (32 байт). Такой шифр принимает на вход 256-битовый открытый текст и выдаёт 256-битовый шифрованный текст.

Блочный шифр является обратимым: существует функция дешифрования, которая принимает на вход 256-битовый шифрованный текст и выдаёт исходный 256-битовый открытый текст. Открытый и шифрованный текст всегда имеет один и тот же размер, который называется размером блока.

1 Алгоритм шифрования DES

Алгоритм шифрования данных DES (Data Encryption Standard) относится к группе методов симметричного блочного шифрования. Он действовал в США в качестве стандарта в течение 25 лет (с 1977 по 2002 гг.)[5, 7, 11].

Общая схема процесса шифрования DES показана на рисунке 1. Как и в случае любой схемы шифрования, здесь на вход функции шифрования подаётся два типа данных – открытый текст, который требуется зашифровать, и ключ. В данном случае длина открытого текста предполагается равной 64 битам, а длина ключа – 56 битам.

Из левой части рисунка 4.1 видно, что процесс преобразования открытого текста состоит из трёх этапов. Сначала 64-битный блок открытого текста поступает для обработки на вход начальной перестановки (*IP*), в результате чего получаются переставленные входные данные.

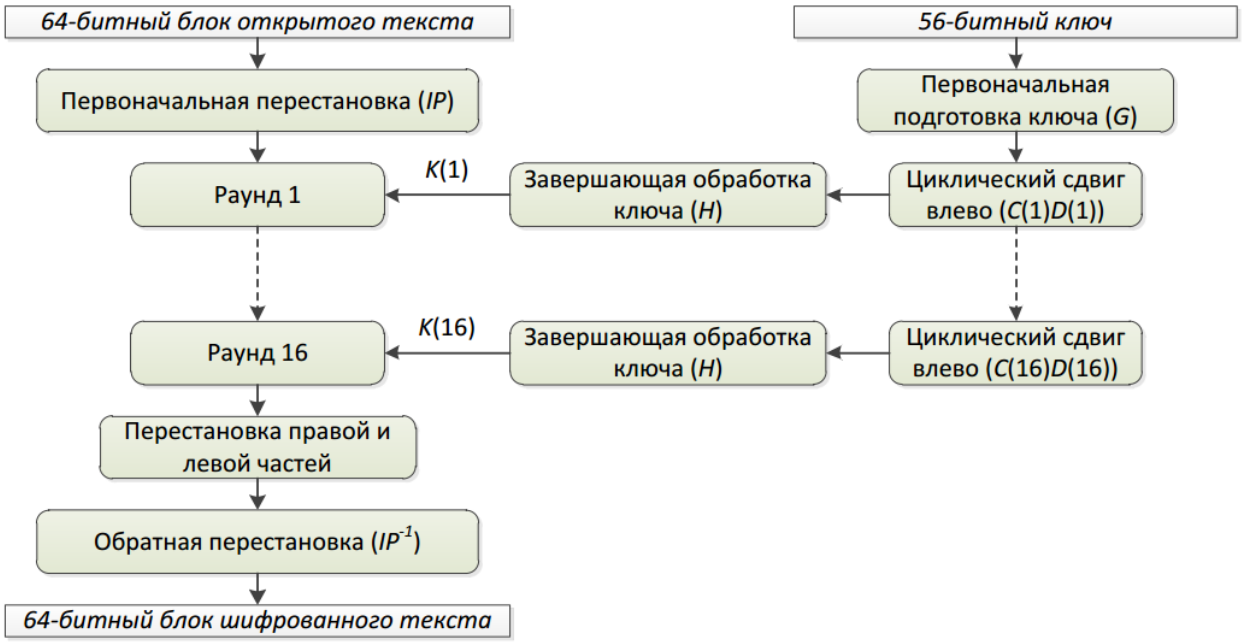

Рисунок 4.1 – Общая схема алгоритма шифрования DES

Затем следует этап, состоящий из 16 раундов применения одной и той же функции, в которой используются операции перестановки и подстановки. На выходе последнего (16-го) раунда получается 64-битная последовательность, являющаяся некоторой функцией открытого текста и ключа. Левая и правая половины полученной последовательности данных меняются местами, образуя предрезультат. Наконец, этот предрезультат проходит через перестановку IP^{-1} обратную начальной, в результате чего получается 64-битный блок шифрованного текста. Согласно рекомендациям Шеннона, в каждом раунде выполняется один шаг перемешивания (с использованием соответствующего раундового ключа и S-блоков), после которого следует шаг рассеивания, не зависящий от ключа.

В правой части рисунка 1 показано, каким образом используется 56-битный ключ. Сначала к ключу тоже применяется функция перестановки. Затем с помощью циклического сдвига влево и некоторой перестановки из полученного результата для каждого из 16 раундов генерируется подключ *K(i)*. Функция перестановки одна и та же для всех раундов, но генерируемые подключи оказываются разными.

Необходимо отметить, что **все** таблицы, приведённые в данной лабораторной работе, **стандартные**, а, следовательно, должны включаться в реализацию алгоритма в неизменном виде. Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс дешифрования путём подбора ключа. Структура алгоритма DES показана на рисунке 4.2.

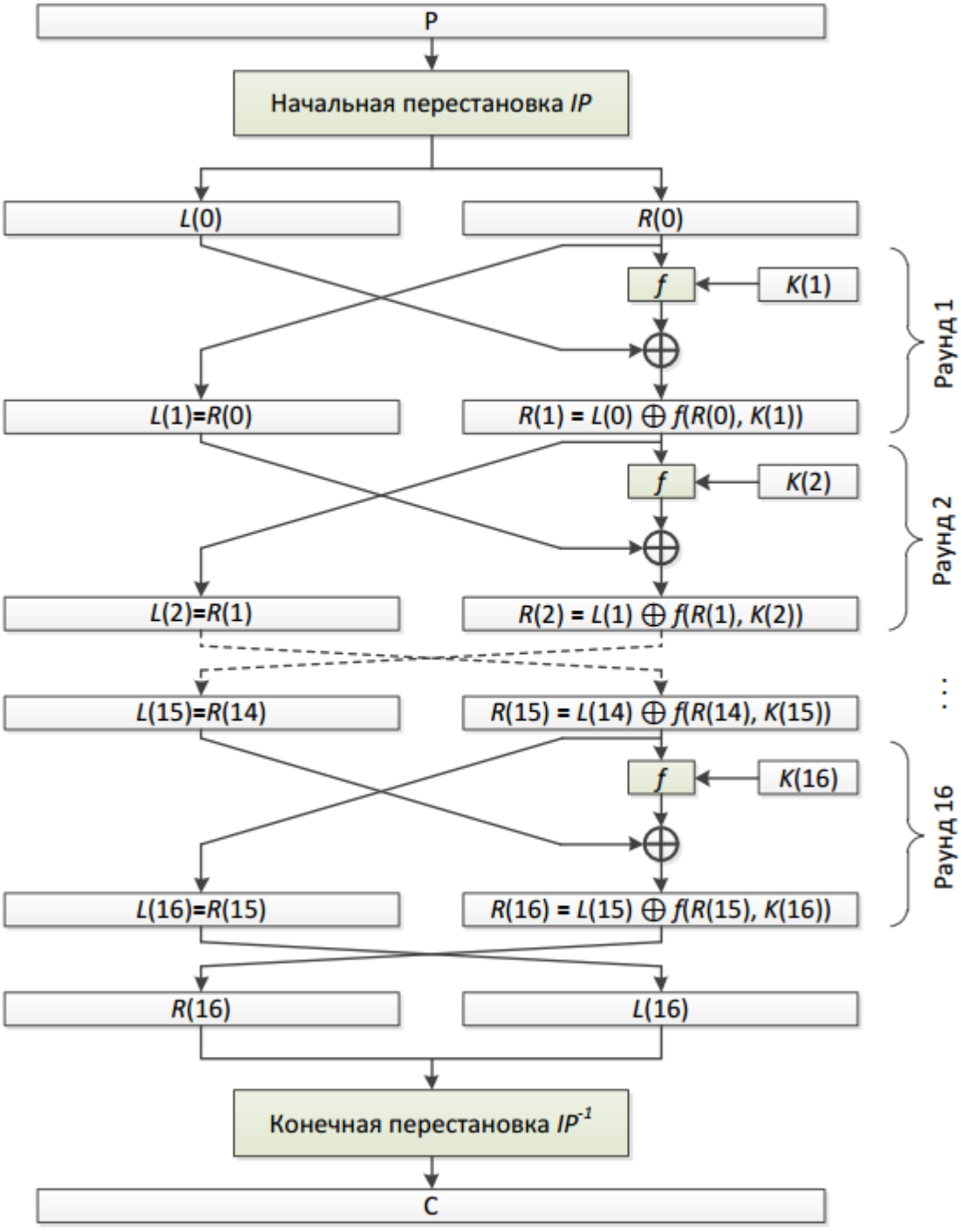

Рисунок 4.2 – Структура алгоритма шифрования DES

Пусть из файла считан очередной 8-байтный блок P, который преобразуется с помощью матрицы начальной перестановки IP (см. таблицу 4.1) следующим образом: бит 58 блока *P* становится битом 1, бит 50 – битом 2 и т.д., что даст в результате: $P\ \ 0\quad = IP(P)$. Полученная последовательность бит $P(0)$ разделяется на две последовательности по 32 бита каждая: $L(0)$ – левые или старшие биты, $R(0)$– правые или младшие биты.

Таблица 4.1 – Матрица начальной перестановки IP

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

Затем выполняется шифрование, состоящее из 16 итераций. Результат *i*- й итерации описывается следующими формулами:

$$L(i) = R(i-1),$$
$$R(i) = L(i-1) \oplus f(R(i-1), K(i)).$$

Функция f называется **функцией шифрования**. Её аргументы – это 32- битная последовательность $R(i-1)$, полученная на $(i-1)$-й итерации, и 48-битный ключ $K(i)$, который является результатом преобразования 64- битного ключа K. Подробно функция шифрования и алгоритм получения ключей $K(i)$ описаны ниже. На 16-й итерации получают последовательности $R(16)$ и $L(16)$, которые дополнительно меняются местами и конкатенируют в результирующую 64-битную последовательность $R(16)L(16)$.

Затем позиции бит этой последовательности переставляют в соответствии с матрицей IP^{-1} (см. таблицу 4.2).

Таблица 4.2 – Матрица обратной перестановки IP^{-1}

40	8	48	16	56	24	64	32

39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Матрицы IP^{-1} и IP соотносятся следующим образом: значение 1-го элемента матрицы IP^{-1} равно 40, а значение 40-го элемента матрицы IP равно 1. Значение 2-го элемента матрицы IP^{-1} равно 8, а значение 8-го элемента матрицы IP равно 2, и т.д.

Процесс расшифрования данных является инверсным по отношению к процессу шифрования. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP, а затем над последовательностью бит $R(16)L(16)$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке. Итеративный процесс расшифрования может быть описан следующими формулами:

$$R(i-1) = L(i),\ \ i = \overline{1,16},$$
$$L(i-1) = R(i) \oplus f(L(i), K(i)),\ \ i = \overline{1,16}.$$

На 16-й итерации получают последовательности $L(0)$ и $R(0)$, которые меняют местами иконкатенируют в 64-битовую последовательность $L(0)R(0)$. Затем позиции битов этой последовательности переставляют в соответствии с матрицей IP^{-1}. Результат такой перестановки – исходная 64- битовая последовательность.

1.1 Функция шифрования *f* для алгоритма DES

Теперь рассмотрим функцию шифрования $f(R(i-1), K(i))$. Схематически она показана на рисунке 4.3:

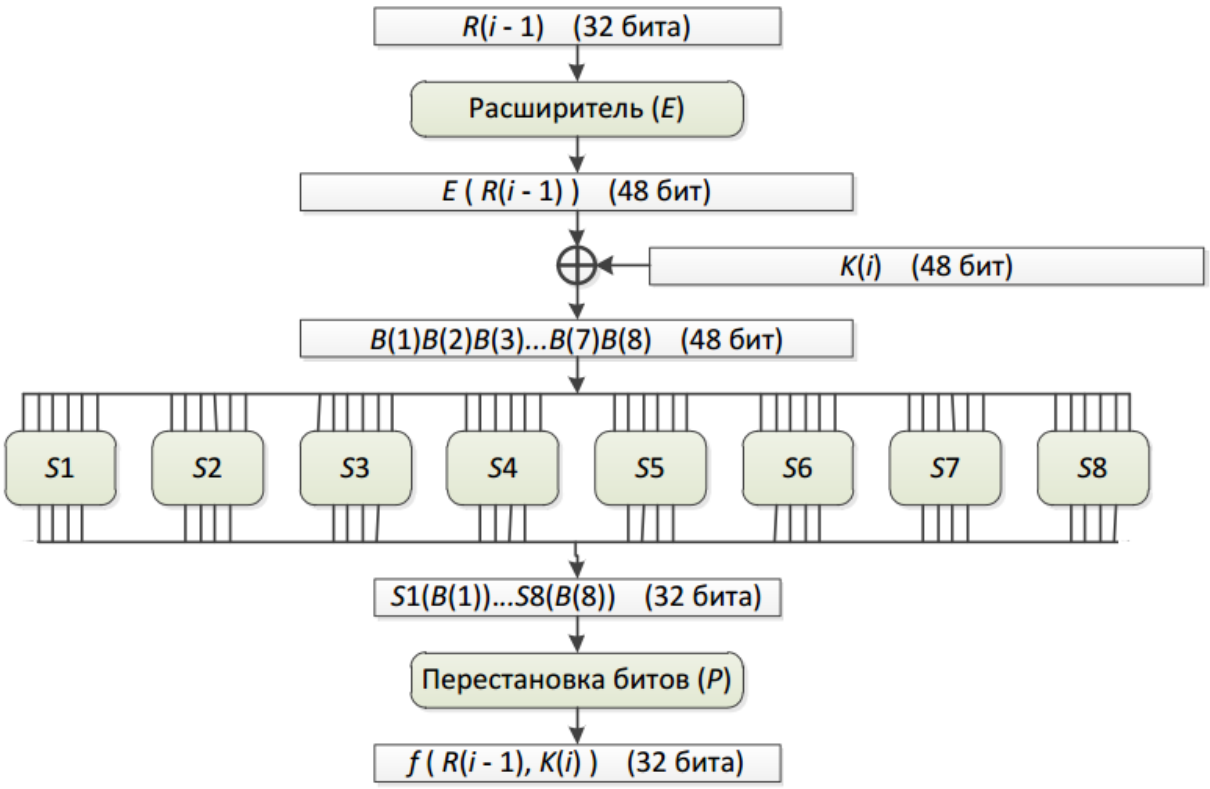

Рисунок 4.3 – Вычисление функции $f(R(i-1), K(i))$

Для вычисления значения функции используются следующие функции-матрицы:

E – расширение 32-битной последовательности до 48-битной;

$S1, S2, \ldots, S8$ – преобразование 6-битного блока в 4-битный;

P – перестановка бит в 32-битной последовательности.

Функция расширения E определяется таблицей 4.3. В соответствии с этой таблицей первые 3 бита $E(R(i-1))$ – это биты 32, 1 и 2, а последние – 31, 32 и 1.

Таблица 4.3 – Функция расширения E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Результат функции $E(R(i-1))$ есть 48-битная последовательность, которая складывается по модулю 2 (операция $\oplus$) с 48-битным ключом $K(i)$. Получается 48-битная последовательность, которая разбивается на восемь 6-битных блоков $B(1)B(2)B(3)B(4)B(5)B(6)B(7)B(8)$, т.е. $E(R(i-1)) \oplus K(i) = = B(1)B(2) \ldots B(8)$. Функции $S1, S2, \ldots, S8$ определяются таблицей 4.4.

Таблиця 4.4 – Функции преобразования $S1, S2, \ldots, S8$

		Номер столбца																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Номера строк	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	**S1**
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	**S2**
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	**S3**
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	**S4**
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	**S5**
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	**S6**
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	**S7**
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	**S8**
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

К таблице 4.4 требуются дополнительные пояснения. Пусть на вход функции-матрицы Sj поступает 6-битный блок $B(j) = b_1b_2b_3b_4b_5b_6$,тогда двухбитное число b_1b_6 указывает номер строки матрицы, а $b_2b_3b_4b_5$ – номер столбца. Результатом $S_j(B(j))$ будет 4-битный элемент, расположенный на пересечении указанных строки и столбца.

Например, $B(1) = 011011$. Тогда $S_1(B(1))$ расположен на пересечении строки 1 и столбца 13. В столбце 13 строки 1 задано значение 5. Значит, $S1(011011) = 0101$.

Применив операцию выбора к каждому из 6-битных блоков $B(1), B(2), \ldots, B(8)$, получим 32-битную последовательность $S1(B(1))$ $S2(B(2))$ $S3(B(3))\ldots S8(B(8))$.

Наконец, для получения результата функции шифрования надо переставить биты этой последовательности. Для этого применяется функция перестановки P (см. таблицу 4.5). Во входной последовательности биты переставляются так, чтобы бит 16 стал битом 1, а бит 7 – битом 2, и т.д.

Таблица 4.5 – Функция перестановки *P*

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Таким образом, $f(R(i-1), K(i) = P(S1(B(1)), \ldots, S8(B(8)))$.

1.2 Формирование подключей для алгоритма DES

Чтобы завершить описание алгоритма шифрования данных, осталось привести алгоритм получения 48-битных ключей $K(i), i = \overline{1,16}$. На каждой итерации используется новое значение ключа $K(i)$, которое вычисляется из

начального ключа K. K представляет собой 64-битный блок с восемью битами контроля чётности, расположенными в позициях $8,\ 16,\ 24,\ 32,\ 40,\ 48,\ 56,\ 64$.

Для удаления контрольных битов и перестановки остальных используется функция G первоначальной подготовки ключа (см. таблицу 4.6).

Таблица 4.6 – Матрица *G* первоначальной подготовки ключа

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Результат преобразования $G(K)$ разбивается на два 28-битных блока $C(0)$ и $D(0)$, причём $C(0)$ будет состоять из бит $57,\ 49,\ \ldots\ ,\ 44,\ 36$ преобразованного ключа $G(K)$, а $D(0)$ будет состоять из бит $63,\ 55,\ \ldots\ ,\ 12,\ 4$ преобразованного ключа $G(K)$. После определения $C(0)$ и $D(0)$ рекурсивно определяются $C(i)$ и $D(i), \overline{i = 1,16}$, . Для этого применяют циклический сдвиг влево на один или два бита в зависимости от номера итерации, как показано в таблице 4.7.

Таблица 4.7 – Таблица сдвигов для вычисления ключа

Номер итерации	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг (бит)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Полученное значение вновь "перемешивается" в соответствии с матрицей H (см. таблицу 4.8).

Таблица 4.8 – Матрица *H* завершающей обработки ключа

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40

51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Ключ $K(i)$ будет состоять из бит $14,\ 17,\ \ldots\ ,\ 29,\ 32$ последовательности $C(i)D(i)$. Таким образом, $K(i) = H(C(i)D(i))$.

Восстановление исходного текста осуществляется по этому же алгоритму, но вначале используется ключ $K(15)$, затем – $K(14)$, и т.д.

2 Алгоритм шифрования ГОСТ 28147-89

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

ГОСТ предусматривает три режима шифрования (простая замена, гаммирование, гаммирование с обратной связью) и один режим выработки имитовставки. Первый из режимов шифрования предназначен для шифрования ключевой информации и не может использоваться для шифрования других данных, для этого предусмотрены два других режима. Режим выработки имитовставки (криптографической контрольной комбинации) предназначен для имитозащиты шифруемых данных, т.е. для их защиты от случайных или преднамеренных несанкционированных изменений.

Алгоритм построен по тому же принципу, что и DES: это классический блочный шифр с секретным ключом. Однако он отличается от алгоритма DES

большей длиной ключа, большим количеством раундов и более простой схемой построения самих раундов. В таблице 4.9 приведены его основные параметры, для удобства – в сравнении с параметрами алгоритма DES:

Таблица 4.9 – Сравнение параметров алгоритмов DES и ГОСТ

Параметры	ГОСТ	DES
Размер блока шифрования	64 бита	64 бита
Длина ключа	256 бит	56 бит
Число раундов	32	16
Узлы замен (S-блоки)	не фиксированы	фиксированы
Длина ключа для одного раунда	32 бита	48 бит
Схема выработки раундового ключа	простая	сложная
Начальная и конечная перестановки бит	нет	есть

В силу намного большей длины ключа алгоритм ГОСТ гораздо устойчивей алгоритма DES к вскрытию посредством полного перебора по множеству возможных значений ключа.

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

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

Также, как и в алгоритме DES, после 32 раунда выполняется перестановка левой и правой частей полученного шифротекста:

$$L_{32}R_{32} \rightarrow R_{32}L_{32}$$

Один такт алгоритма ГОСТ показан на рисунке 4.4. Один такт алгоритма осуществляет одно преобразование Фейстеля[6].

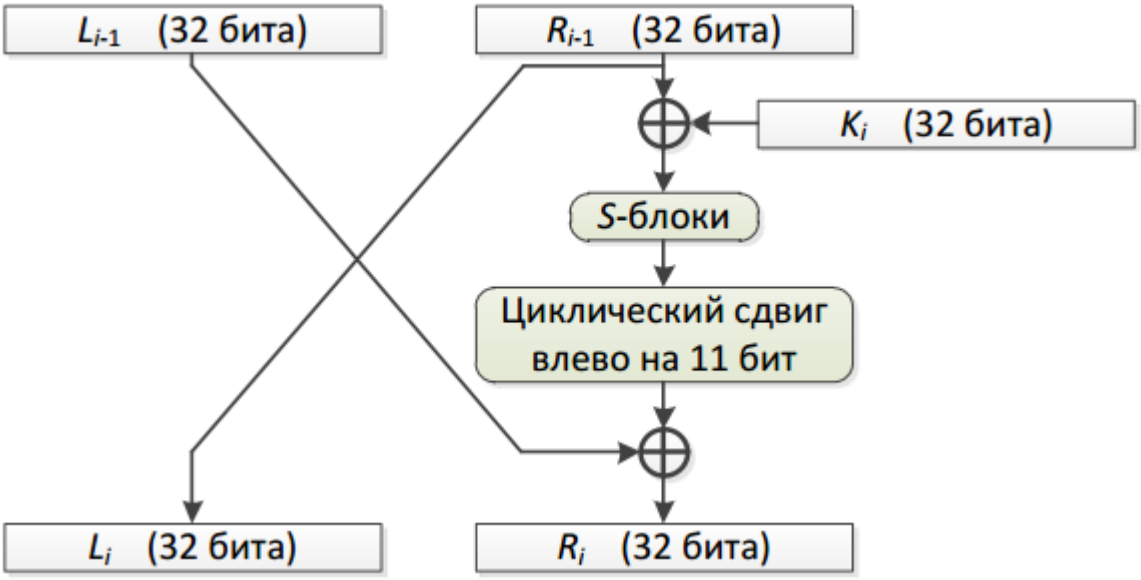

Рисунок 4.4 – Один такт алгоритма ГОСТ 28147-89

2.1 Описание функции *f* для алгоритма ГОСТ 28147-89

Сначала правый блок R_{i-1} складывается по модулю 2^{32} с подключом K_i. Полученное 32-битное сообщение делится на восемь 4-битных чисел. Каждое из этих 4-битных чисел преобразуется соответствующим S-блоком в другое 4-битное число. Поэтому любой S-блок определяется некоторой 16-битной перестановкой на множестве из 16 элементов $0,\ 1,\ \ldots\ ,\ 15$. В ГОСТ использовались следующие S-блоки:

$$
\begin{aligned}
S1 &= 4\ 10\ 9\ 2\ 13\ 8\ 0\ 14\ 6\ 11\ 1\ 12\ 7\ 15\ 5\ 3\ ,\\
S2 &= 14\ 11\ 4\ 12\ 6\ 13\ 15\ 10\ 2\ 3\ 8\ 1\ 0\ 7\ 5\ 9\ ,\\
S3 &= 5\ 8\ 1\ 13\ 10\ 3\ 4\ 2\ 14\ 15\ 12\ 7\ 6\ 0\ 9\ 11\ ,\\
S4 &= 7\ 13\ 10\ 1\ 0\ 8\ 9\ 15\ 14\ 4\ 6\ 12\ 11\ 2\ 5\ 3\ ,\\
S5 &= (6\ 12\ 7\ 1\ 5\ 15\ 13\ 8\ 4\ 10\ 9\ 14\ 0\ 3\ 11\ 2\ ),\\
S6 &= (4\ 11\ 10\ 0\ 7\ 2\ 1\ 13\ 3\ 6\ 8\ 5\ 9\ 12\ 15\ 14\ ),\\
S7 &= (13\ 11\ 4\ 1\ 3\ 15\ 5\ 9\ 0\ 10\ 14\ 7\ 6\ 8\ 2\ 12\ ),\\
S8 &= (l\ 15\ 13\ 0\ 5\ 7\ 10\ 4\ 9\ 2\ 3\ 14\ 6\ 11\ 8\ 12\ ).
\end{aligned}
$$

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

2.2 Формирование подключей ГОСТ 28147-89

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

$$1\,2\,3\,4\,5\,6\,7\,8\,1\,2\,3\,4\,5\,6\,7\,8\,1\,2\,3\,4\,5\,6\,7\,8\,8\,7\,6\,5\,4\,3\,2\,1.$$

При дешифровании порядок использования подключей меняется на противоположный.

Задание

Выполнение

Данные соответствующие варианту из таблицы4.10.

1) Описание алгоритма, заданного вариантом.
2) Блок-схема алгоритма с описанием блоков.
3) Математическая модель алгоритма. Шифрование/дешифровка сообщения по варианту из таблицы1.6.
4) Результаты шифрования сообщения заданным ключом.

Для получения максимального балла требуется выполнить дополнительные задания:

7) Написать алгоритм шифровки-дешифровки заданным способом. Алгоритм должен быть представлен в виде подробной блок-схемы.

– AES (американский стандарт шифрования);
– ГОСТ 28147-89 (российский стандарт шифрования);
– DES (стандарт шифрования данных в США);
– 3DES (тройной DES);
– RC2 (Шифр Ривеста);
– RC5;
– Другие блочные алгоритмы…

8) Следующим шагом выберите алгоритм для шифрования. В нашем случае это будет
9) Составить шаблон для шифрования.
10) Выбор методов шифрования. Зашифровать для начала ручным способом, после чего написать алгоритм автоматической шифровки сообщения на любом языке. Для примера можно использовать один из следующих алгоритмов:
11) Инструкция

12) Выбрать данные из таблицы. Разложить данные

13) Расшифровать криптотекст требуемым методом.

14) Что записать в отчёт.

Таблиця 4.10 – Варианты для шифрования

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Вопросы для самоконтроля

1) Общая схема алгоритма шифрования DES.

2) Почему длина ключа для алгоритма DES равна 56 бит?

3) Как соотносятся между собой матрицы P^{-1} и P в алгоритме DES?

4) Как происходит шифрование функции $f\ R\ i-1\ ,\mathrm{K}\ i$ в алгоритме DES?

5) Алгоритм получения раундовых ключей $\mathrm{K}\ i$ в алгоритме DES.

6) В чём заключается процесс расшифрования данных в DES?

7) Что представляет собой функция $f()$ в алгоритме ГОСТ?

8) Как используется заданный ключ при шифровании в алгоритме ГОСТ?

9) Общие черты и отличия алгоритмов ГОСТ и DES.

5 РЕЖИМЫ РАБОТЫ БЛОЧНЫХ ШИФРОВ. СХЕМЫ КРАТНОГО ШИФРОВАНИЯ

Цель: Изучить и реализовать режимы работы блочных шифров и схемы кратного шифрования для симметричных алгоритмов шифрования.

Теоретические ведомости

1 Режимы работы блочных шифров

Режимами шифрования называют различные алгоритмы обработки данных, построенные на основе базового режима ECB. Криптографическая стойкость этих алгоритмов определяется в основном стойкостью базового режима. Однако особенности различных режимов шифрования позволяют использовать блочный шифр для решения различных криптографических задач.

1.1 Режим электронной шифровальной книги [ECB]

Простейшим режимом является ***режим электронной шифровальной книги*** (*Electronic Codebook*), когда открытый текст обрабатывается блоками по 64 бита и каждый блок шифруется с одним и тем же ключом[5]. Термин ***шифровальная книга*** объясняется тем, что при заданном ключе каждый 64-битный блок открытого текста представляется уникальным блоком шифрованного текста. Если длина сообщения превышает 64 бита, то оно разделяется на 64-битные блоки с добавлением при необходимости заполнителей к последнему блоку.

Уравнение шифрования режима ECB: $C_i = E_k(P_i)$, где $i = \overline{1,N}$ P_i – входные блоки открытого текста, C_i – соответствующий шифрованный текст, E_k – алгоритм шифрования с использованием ключа k .

Уравнение дешифрования режима ECB: $P_i = D_k(C_i)$, где $i = 1, N, Dk = E_k^{-1}$ – алгоритм дешифрования с использованием ключа k.

Режим ECB имеет следующие особенности:

1) Замены и перестановки отдельных блоков в шифротексте не нарушают корректности расшифрования остальных блоков текста.

2) Шифрование на одном ключе одинаковых блоков открытого текста даёт одинаковые блоки шифротекста.

3) В силу хорошего перемешивания информации шифрующими подстановками (это общее свойство блочных шифров) каждый из 64 бит выходного блока может быть искажён с вероятностью $\frac{1}{2}$ при искажении лишь одного случайно выбранного бита входного блока C_i единичная ошибка в блоке C_i порождает, в среднем, ошибок внутри шифроблока P_i на следующие шифроблоки ошибка не распространяется.

При искажении бит выходного блока P_i соответствующий блок C_i расшифровывается некорректно, а остальные блоки расшифровываются верно. Но если бит шифротекста случайно потерян или добавлен, то в силу произошедшего сдвига весь последующий текст расшифровывается некорректно. Для локализации последствий сдвига следует предусмотреть средство контроля границ блоков.

Первые две особенности позволяют активному противнику, контролирующему линию связи, защищаемую шифром в режиме ECB, наблюдать частоты появления отдельных блоков и сообщений. В определённых условиях он может генерировать ложные сообщения, не зная ни ключа, ни алгоритма шифрования, даже если сообщения содержат метки времени. В силу этих серьезных недостатков режим ECB не используется для шифрования длинных сообщений. В этом режиме шифруются лишь короткие сообщения вспомогательного характера: пароли, сеансовые ключи и т.п.

1.2 Режим сцепления шифрованных блоков [CBC]

Технология, свободная от недостатков режима ECB, должна в случае повторения в сообщении уже встречавшегося ранее блока открытого текста генерировать блок шифрованного текста, отличный от сгенерированного ранее. Проще всего добиться этого с помощью ***режима сцепления шифрованных блоков*** (*Cipher Block Chaining*, см. рисунок 5.1)[5].

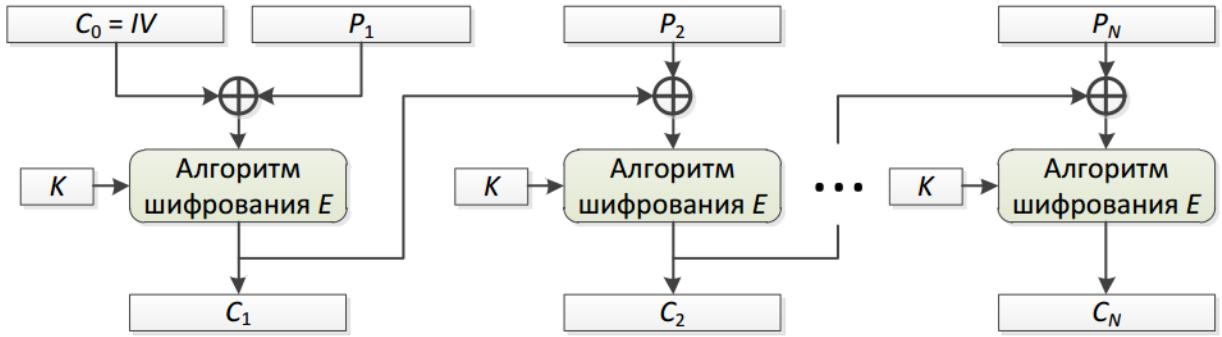

Рисунок 5.1 – Схема шифрования в режиме CBC

Начальный вектор может передаваться в линию связи как в открытом, так и в шифрованном виде (в частности, с помощью режима ECB). Таким образом, если исходный текст состоял из *N* блоков, то результат шифрования в режиме CBC будет содержать *N+1* блоков. Однако важно избегать повторения синхропосылки в разных сообщениях, шифруемых одинаковым ключом. Это затрудняет атаку на шифротекст, основанную на наличии стандартов в начале сообщения. В качестве синхропосылки используется некоторая строка случайных байт либо метка времени.

Искажение одного бита в блоке P_i влечёт искажение примерно половины бит во всех блоках шифротекста, начиная с C_i. Для расшифрования это несущественно, так как восстановленный текст содержит ту же единственную ошибку.

Искажение *k*-го бита в блоке C_i,(такие искажения возможны из-за шумов в линиях связи или сбоев в устройствах хранения) влечёт искажение около

половины бит в блоке P_i, и *k*-го бита в блоке P_{i+1}. Следующие блоки расшифровываются корректно (самовосстанавливаются). В то же время, режим CBC совершенно неустойчив к ошибкам синхронизации.

Симметричный алгоритм шифрования в режиме CBC используется для поблочной передачи данных общего назначения и аутентификации.

1.3 Обратная связь по шифротексту [CFB]

В некоторых практических ситуациях требуется шифровать символы поступающего потока, не дожидаясь, когда сформируется целый блок данных. В таких случаях удобен режим (*Cipher Feedback*), в котором блоки открытого и шифрованного текста имеют длину *j* бит, где *j*– параметр режима, $1 \leq j \leq 64$ Обозначим такой режим шифрования CFB-*j*[5]. В связи с байтным представлением информации параметр выбирается, как правило, равным восьми.

На входе функции шифрования размещается 64-битный регистр сдвига, в котором изначально размещается некоторое значение инициализационного вектора (*IV*). Крайние слева (старшие)*j* бит этого значения связываются операцией XOR с первой порцией открытого текста P_1, в результате чего получается первая порция шифрованного текста C_1, который подаётся на линию передачи данных. Содержимое регистра сдвига смещается влево на *j* бит, а в крайние справа (младшие) *j* бит помещается значение C_1. Затем весь процесс повторяется до тех пор, пока не будут зашифрованы все элементы открытого текста (см. рисунок 5.2)[11].

Алгоритм шифрования в режиме CFB:

$I_1 = IV.$

$O_i = E_k(I_i), \qquad i = \overline{1, N}.$

$C_i = P_i \oplus St_j(O_i), \qquad i = \overline{1, N}.$

$I_i = Ml_{64-j}(I_{i-1}) \,||\, C_{i-1}, \qquad i = \overline{2, N},$

где $St_j(X)$ – крайние слева (старшие) j бит блока X, $Ml_j(X)$ – крайние справа (младшие) j бит блока X, $||$ – операция конкатенации битовых строк.

Алгоритм дешифрования в режиме CFB:

$$I_1 = IV.$$

$$O_i = E_k(I_i), \qquad i = \overline{1,N}.$$

$$P_i = C_i \oplus St_j(O_i), \qquad i = \overline{1,N}.$$

$$I_i = Ml_{64-j}(I_{i-1}) \,||\, C_{i-1}, \qquad i = \overline{2,N}.$$

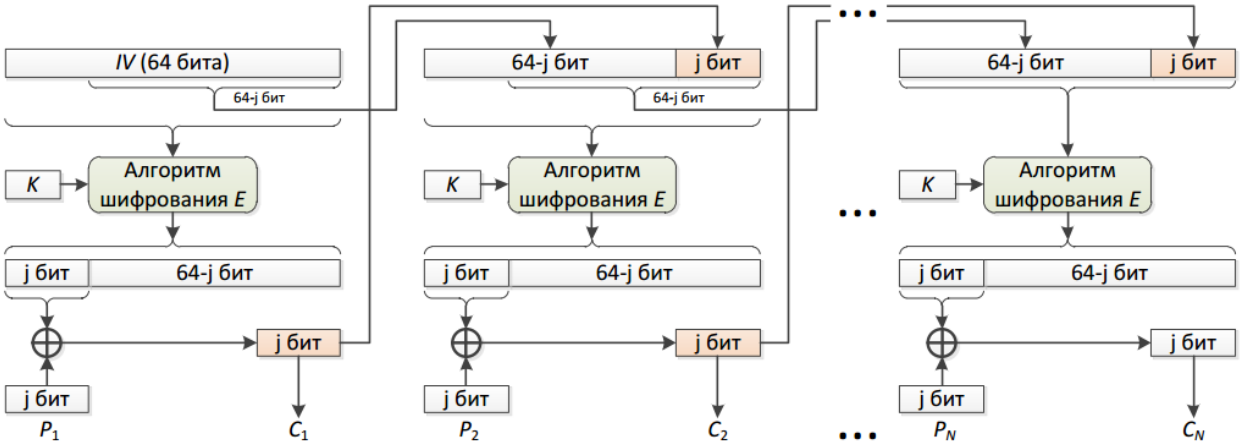

Рисунок 5.2 – Схема шифрования в режиме CFB

Как и в режиме CBC, синхропосылка может передаваться в линию связи в открытом виде. Однако необходимо исключить повторение синхропосылки в различных сообщениях, шифруемых одинаковым ключом.

Искажение одного бита в блоке P_i влечёт искажение одного бита в C_i и в среднем половины бит во всех блоках шифротекста, начиная с C_{i+1}, но при расшифровании получается открытый текст с той же единственной ошибкой.

Искажение k-го бита в блоке C_i влечёт искажение k-го бита в блоке P_i. Затем ошибка поступает в регистр состояний и искажает в среднем половину бит в каждом из следующих блоков, где $\left[\frac{64}{j}\right] \le l$. В дальнейшем блоки расшифровываются корректно.

Режим CFB самостоятельно восстанавливается после ошибок синхронизации.

Симметричный алгоритм шифрования в режиме CFB используется для потоковой передачи данных общего назначения и аутентификации.

1.4 Обратная связь по выходу [OFB]

Блочный шифр в режиме OFB можно рассматривать как синхронный шифр гаммирования, обрабатывающий *j*-битные блоки открытого и шифрованного текста (обозначим такой режим OFB-*j*) [5].

Режим обратной связи по выходу (*Output Feedback*) во многом подобен режиму CFB. В режиме OFB в регистр сдвига подаётся значение, получаемое на выходе функции шифрования (см. рисунок 5.3), а в режиме CFB в этот регистр подаётся порция шифрованного текста.

Алгоритм шифрования в режиме OFB:

$$I_1 = IV.$$

$$O_i = E_k(I_i), \qquad i = \overline{1,N}.$$

$$C_i = P_i \oplus St_j(O_i), \qquad i = \overline{1,N}.$$

$$I_i = Ml_{64-j}(I_{i-1}) \,||\, St_j(O_{i-1}), \qquad i = \overline{2,N},$$

где $St_j(X)$ – крайние слева (старшие) *j* бит блока X, $Ml_j(X)$ – крайние справа (младшие) *j* бит блока X, $||$ – операция конкатенации битовых строк.

Алгоритм дешифрования в режиме OFB:

$$I_1 = IV.$$

$$O_i = E_k(I_i), \qquad i = \overline{1,N}.$$

$$P_i = C_i \oplus St_j(O_i), \qquad i = \overline{1,N}.$$

$$I_i = Ml_{64-j}(I_{i-1}) \,||\, St_j(O_{i-1}), \qquad i = \overline{2,N}.$$

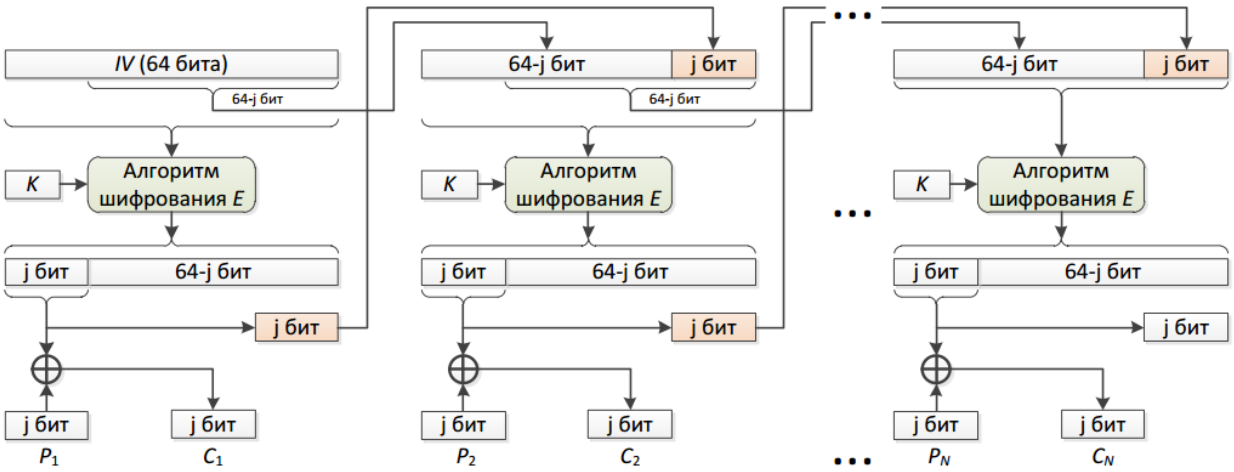

Рисунок 5.3 – Схема шифрования в режиме OFB

Синхропосылка может передаваться в линию связи в открытом виде, но необходимо исключить повторение синхропосылки в разных сообщениях, шифруемых одинаковым ключом.

При использовании режима OFB чрезвычайно важно сохранять синхронизацию. Для этого необходимо предусмотреть средство контроля над синхронизацией и средство восстановления синхронизации в случае её потери.

В режиме OFB ошибки не распространяются, что является позитивным при передаче оцифрованных речевых сигналов или видеоизображений.

Симметричный алгоритм шифрования в режиме работы OFB используется для потоковой передачи данных по каналам с помехами (например, по спутниковой связи).

1.5 Другие режимы шифрования

Разработка других режимов шифрования стимулировалась стремлением устранить некоторые недостатки четырёх основных режимов[7].

BC – ***сцепление блоков***. Этот режим задаётся следующим равенством:

$$C_1 = E_k(P_i \oplus \sum_{j=0}^{i-1} C_j), \qquad i = \overline{1, N},$$

Где $C_0 = IV$ – вектор инициализации. Основной недостаток режима ВС заключается в том, что при расшифровании единичная ошибка в шифротексте влечёт некорректное расшифрование всех последующих блоков шифротекста.

PCBC – ***сцепление блоков шифротекста с распространением ошибки***. Режим задаётся следующим равенством: $C_i = E_k(P_i \oplus C_{i-1}),\ i = \overline{1,N}$, где $C_0 = IV$ и $P_0 = IV'$ – векторы инициализации.

Этот режим используется в протоколе *Kerberos 4* для выполнения за один проход и шифрования, и проверки целостности. В режиме PCBC единичная ошибка в шифротексте влечёт некорректное расшифрование всех последующих блоков шифротекста, что используется для проверки целостности сообщений. Однако если проверка целостности охватывает лишь завершающий отрезок текста, то могут остаться незамеченными перестановки пары шифроблоков в начале текста. Это подозрительное свойство заставило разработчиков отказаться от данного режима в пользу CBC в следующей версии протокола *Kerberos*.

OFBNLF – нелинейная обратная связь по выходу. Этот режим наследует некоторые свойства режимов OFB и ECB: $C_i = E_{ki}(P_i), k_i = E_k(k_{i-1}), i = \overline{1,N}$, где k_0 – вектор инициализации.

Единичная ошибка в шифротексте распространяется только на один блок открытого текста, однако требуется поддержка синхронизации.

Скорость обработки информации определяется не только скоростью шифрования базовым алгоритмом, но и скоростью обновления текущего значения ключа.

2 Кратное шифрование

Криптоаналитикам не удалось разработать практически приемлемый метод дешифрования алгоритма DES, который был бы лучше метода полного пе-

ребора ключей. Вместе с тем короткая длина ключа алгоритма DES не позволяла рассматривать его как надёжное средство защиты информации. Это стимулировало криптографов заняться построением блочного шифра с длинным ключом, использующего в качестве базового элемента алгоритм DES.

Один из методов – многократное шифрование с использованием базового алгоритма. Этот метод применим к любому симметричному блочному шифру, однако его использование приводит к уменьшению скорости шифрования (либо требуется больше аппаратных ресурсов) в соответствующее число раз. Кроме того, важно, чтобы множество шифрующих подстановок не являлось группой (для алгоритма DES это доказано), иначе кратное шифрование сводится к однократному.

2.1 Двойное шифрование

Простейшая схема кратного шифрования [5] – это ***двойное шифрование*** с помощью двух шифрующих подстановок с независимыми ключами:

$$C = E_{k_2}(E_{k_1}(P)).$$

Заметим, что при последовательном использовании алгоритма DES дважды с двумя разными ключами будет получено отображение, отличное от всех тех, которые получаются при однократном применении алгоритма DES, т.е. $k_3 : E_{k_2}(E_{k_1}(P)) = E_{k_3}(P)$. Здесь следует заметить, что, несмотря на многочисленные и вполне очевидные аргументы в пользу такого утверждения, оно было строго доказано только в 1992 году.

Итак, повторное применение алгоритма DES задаёт отображение, неэквивалентное отображениям, получаемым при однократном использовании этого алгоритма. Но для криптоанализа этой схемы существует и другой путь, независящий от конкретных особенностей алгоритма DES, а, наоборот, подходящий для всех блочных шифров.

Соответствующий алгоритм получил название ***метода двусторонней атаки*** (*meet-in-the- middle attack*). Он основан том факте, что из $C = E_{k_2}(E_{k_1}(P))$ следует, что $X = E_{k_1}(P) = D_{k_2}(C)$.

При наличии известной пары (P, C) анализ с помощью данного метода выполняется по следующей схеме. Сначала проводится шифрование P с каждым из2^{56} возможных значений k_1. Полученные результаты сохраняются в таблице, а сама таблица сортируется по значениям X. Затем выполняется дешифрование C для каждого из2^{56} возможных значений k_2. с одновременным сравнением получаемых результатов с результатами, представленными в таблице. Если обнаруживается совпадение, то соответствующие два ключа проверяются на следующей известной паре открытого и шифрованного текста. Если оказывается, что и с новой парой эти два ключа порождают правильный шифрованный текст, они принимаются как истинные ключи.

Оказывается, что оценка для сложности задачи криптоанализа «двойного» алгоритма DES с известным открытым текстом, несмотря на размер ключа в 112 битов, имеет порядок 2^{56}, что ненамного превышает оценку 2^{55}, имеющую место для обычного алгоритма DES.

Другой способ двойного шифрования, называемый ***методом Дэвиса-Прайса***, построен на идеях режима шифрования CBC:

$$C_i = E_{k_2}(P_i \oplus E_{k_1}(C_{i-1})), i = \overline{1, N}.$$

Метод двусторонней атаки позволяет определить ключи и в этом случае, характеристики сложности метода примерно такие же.

2.2 Тройное шифрование

Более стойкие схемы используют тройное шифрование. Очевидным решением проблемы противодействия методам двусторонней атаки является тройное шифрование с использованием трёх разных ключей[5]. Это увеличивает сложность задачи криптоанализа с известным открытым текстом до 2^{112},

что выходит за рамки как сегодняшних реальных возможностей, так и возможностей обозримого будущего. Однако недостаток такого подхода – требование использовать ключ размером $56 \times 3 = 168$ бит, который можно считать уж слишком громоздким.

Схему тройного шифрования Тачмена с парой независимых ключей k_1 и k_2 называют ***режимом EDE*** (зашифрование-расшифрование-зашифрование):

$$C = E_{k_1}(D_{k_2}(E_{k_1}(P)).$$

Тройной алгоритм DES с двумя ключами стал довольно популярной альтернативой стандартному алгоритму DES, рекомендованной стандартами управления ключами ANSI X9.17 и ISO 8732.

При одинаковых ключах эта схема равносильна однократному шифрованию, что позволяет совместно использовать эти схемы в общей сети. Несмотря на чередование ключей, исключающее стандартный метод согласования, Меркл и Хеллман разработали оригинальный вариант метода согласования памяти и времени, требующий выполнения порядка 2^{56} операций и некоторого количества подобранных блоков открытого текста.

Хотя данные методы криптоанализа и оказываются несостоятельными с практической точки зрения, использование тройного DES с двумя ключами может вызывать некоторые опасения относительно его криптоаналитической стойкости. Поэтому многие специалисты склоняются к мысли, что более надёжной альтернативой является ***тройной алгоритм DES с тремя ключами***. Последний использует ключ, эффективная длина которого составляет 168 бит, и выполняет преобразование, задаваемое формулой:

$$C = E_{k_3}(D_{k_2}(E_{k_1}(P)).$$

Тройной алгоритм DES с тремя ключами реализован во многих приложениях, ориентированных на работу в сети *Internet*, в том числе в **PGP** и **S/MIME**.

Рассмотренные схемы кратного шифрования могут сочетаться с различными режимами шифрования.

Задание

I. Реализовать приложение для шифрования, позволяющее выполнять следующие действия:
1. Шифровать данные с использованием заданного в варианте режима шифрования, применённого для того симметричного алгоритма, который был реализован в предыдущей лабораторной работе:
1) шифруемый текст должен храниться в одном файле, ключ шифрования – в другом, а вектор инициализации – в третьем;
2) зашифрованный текст должен сохраняться в файл;
3) в процессе шифрования предусмотреть возможность просмотра и изменения ключа, вектора инициализации, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде.
2. Шифровать данные по заданной в варианте схеме кратного шифрования.

II. Реализовать приложение для дешифрования, позволяющее выполнять следующие действия:
1. Дешифровать данные с использованием заданного в варианте режима шифрования, применённого для того симметричного алгоритма, который был реализован в предыдущей лабораторной работе:
1) зашифрованный текст должен храниться в одном файле, ключ – в другом, а вектор инициализации – в третьем;
2) расшифрованный текст должен сохраняться в файл;
3) в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, вектора инициализации, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.
2. Дешифровать данные по заданной в варианте схеме кратного шифрования.

Варианты

Таблица 5.1 – Варианты режимов шифрования

Вариант	Режим шифрования	Схема кратного шифрования
1	CFB	Схема с тремя ключами
2	BC	Метод Дэвиса-Прайса
3	PCBC	EDE
4	OFB	Простая двойная
5	CBC	Схема с тремя ключами
6	OFBNLF	Метод Дэвиса-Прайса
7	OFB	Схема с тремя ключами
8	CFB	EDE
9	CBC	Метод Дэвиса-Прайса
10	BC	Простая двойная
11	OFB	EDE
12	PCBC	Схема с тремя ключами
13	OFBNLF	Метод Дэвиса-Прайса
14	CFB	Простая двойная
15	CBC	EDE

Вопросы для самоконтроля

1) Режим электронной шифровальной книги (ECB).
2) Режим сцепления шифрованных блоков (CBC).
3) Режим обратной связи по шифротексту (CFB).
4) Режим обратной связи по выходу (OFB).
5) Режим сцепления блоков (BC).
6) Режим сцепления блоков шифротекста с распр. ошибки (PCBC).
7) Режим нелинейной обратной связи по выходу (OFBNLF).
8) Двойное шифрование.
9) Метод двусторонней атаки.
10) Тройное шифрование.

6 ЛИНЕЙНЫЙ КРИПТОАНАЛИЗ.

ОЦЕНКА СВОЙСТВ АЛГОРИТМОВ

Цель: Изучить основные методы линейного криптоанализа. Познакомиться с критериями оценки свойств лавинного эффекта.

Теоретические ведомости

1 Оценка надёжности шифров

Методы оценки качества криптоалгоритмов, используемые на практике:

1) всевозможные попытки их вскрытия;
2) анализ сложности алгоритма дешифрования;
3) оценка статистической безопасности шифра.

В первом случае многое зависит от квалификации, опыта, интуиции криптоаналитика и от правильной оценки возможностей противника. Обычно считается, что противник знает шифр, имеет возможность его изучения, знает некоторые характеристики открытых защищаемых данных, например тематику сообщений, их стиль, стандарты, форматы и т. п. Рассмотрим следующие примеры возможностей противника:

1) противник может перехватывать все зашифрованные сообщения, но не имеет соответствующих им открытых текстов;
2) противник может перехватывать все зашифрованные сообщения и добывать соответствующие им открытые тексты;
3) противник имеет доступ к шифру (но не ключам!) и поэтому может зашифровывать и расшифровывать любую информацию.

Во втором случае оценку стойкости шифра заменяют оценкой минимальной сложности алгоритма его вскрытия. Однако получение строго доказуемых оценок нижней границы сложности алгоритмов рассматриваемого типа не представляется возможным. Иными словами, всегда возможна ситуация, когда алгоритм вскрытия шифра, сложность которого анализируется, оказывается вовсе не самым эффективным.

Сложность вычислительных алгоритмов можно оценивать числом выполняемых элементарных операций, при этом, естественно, необходимо учитывать их стоимость и затраты на их выполнение. В общем случае это число должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютерных систем. Качественный шифр невозможно раскрыть способом более эффективным, чем полный перебор по всему ключевому пространству, при этом криптограф должен рассчитывать только на то, что у противника не хватит времени и ресурсов, чтобы это сделать.

Алгоритм полного перебора по всему ключевому пространству это пример так называемого экспоненциального алгоритма. Если сложность алгоритма выражается неким многочленом (полиномом) от n, где n - число элементарных операций, такой алгоритм носит название полиномиального.

В третьем случае считается, что надежная криптосистема с точки зрения противника является «чёрным ящиком», входная и выходная информационные последовательности которого взаимно независимы, при этом выходная зашифрованная последовательность является псевдослучайной. Поэтому смысл испытаний заключается в проведении статистических тестов, устанавливающих зависимость изменений в зашифрованном тексте от изменений символов или битов в исходном тексте или ключе, а также анализирующих, насколько выходная зашифрованная последовательность по своим статистическим свойствам приближается к истинно случайной последовательности. Случайность текста шифровки можно приближённо оценивать степенью её сжатия при использовании алгоритма Лемпела-Зива, применяемого в архиваторах IBM PC.

Если степень сжатия больше 10%, то можно считать криптосистему несостоятельной.

2 Критерии оценки свойств «лавинного эффекта»

Пусть $X^{(i)} = X \oplus E_i$, т.е. бинарный вектор, полученный инвертированием *i*-го бита в векторе $X \in \mathbb{U}$. Тогда бинарный вектор $Y^{(i)} \oplus Y = F(X^{(i)}, K) \oplus F(X, K)$ называется ***лавинным вектором по компоненту*** (здесь *F* – функция шифрования). Пусть для рассматриваемых критериев *X* – это 1 блок открытого текста, размерность вектора *X* равна *n*, а *Y* – *m* и пусть $\mathbb{U}$ – это множество, состоящее из всех блоков открытого текста.

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

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

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

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

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

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

2) Степень полноты преобразования:

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

3) Степень лавинного эффекта:

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

4) Степень соответствия строгому лавинному критерию:

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

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

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

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

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

Задание

Исследовать лавинный эффект (исследования проводить на одном блоке текста):

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

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

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

Исследовать лавинный эффект для реализованного режима шифрования (рассматривать

текст из трёх блоков):

1) построить графики зависимости числа изменённых бит в блоках , , от позиции изменившегося бита в открытом тексте (3 отдельных графика или 3 зависимости на 1 графике);

2) построить графики зависимости числа изменённых бит в блоках C_1, C_2, C_3 от позиции изменившегося бита в ключе (3 отдельных графика или 3 зависимости на 1 графике);

3) построить графики зависимости числа изменённых бит в блоках C_1, C_2, C_3 от позиции изменившегося бита в векторе инициализации (3 отдельных графика или 3 зависимости на 1 графике);

4) построить графики зависимости числа изменённых бит в блоках C_1, C_2, C_3 от позиции изменившегося бита в зашифрованном тексте (3 отдельных графика или 3 зависимости на 1 графике).

3. Исследовать лавинный эффект для реализованной схемы кратного шифрования (рассматривать текст из 1 блока).

Ход работы

Вопросы для самоконтроля

7 КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Цель: Изучить криптографические протоколы. Освоить методы генерации и проверки больших чисел. Познакомиться с теоремой Эйлера. Изучить схему обмена ключами Диффи-Хеллмана.

Теоретические ведомости

В криптографических системах с открытым ключом пользователи обладают собственным открытым и частным закрытым ключами. К открытому ключу имеют доступ все пользователи, и информация шифруется именно с его помощью. А вот для расшифровки необходим частный ключ, находящийся у конечного пользователя. В отличие от криптограмм с секретным ключом в такой системе участниками являются не две, а три стороны. Третья может представлять собой сотового провайдера или, например, банк. Однако эта сторона не заинтересована в хищении информации, поскольку она заинтересована в правильном функционировании системы и получении положительных результатов.

1 Криптографические протоколы

В современной криптографии большое внимание уделяется не только созданию и исследованию шифров, но и разработке криптографических протоколов.

Протокол – это последовательность шагов, которые предпринимают две или большее количество сторон для совместного решения задачи. Все шаги следуют в порядке строгой очередности, и ни один из них не может быть сделан прежде, чем закончится предыдущий. Кроме того, любой протокол подразумевает участие, по крайней мере, двух сторон.

Криптографический протокол – это такая процедура взаимодействия двух или более абонентов с использованием криптографических средств, в результате которой абоненты достигают своей цели, а их противники - не достигают. В основе протокола лежит набор правил, регламентирующих использование криптографических преобразований и алгоритмов в информационных процессах. Каждый криптографический протокол предназначен для решения определённой задачи.

Рассмотрим простейший протокол для обмена конфиденциальными сообщениями между двумя сторонами, которые будем называть абонент №1 и абонент №2. Пусть абонент №1 желает передать зашифрованное сообщение абоненту №2. В этом случае их последовательность действий должна быть следующей.

1) Абоненты выбирают систему шифрования (например, шифр Цезаря).

2) Абоненты договариваются о ключе шифрования.

3) Абонент №1 шифрует исходное сообщение с помощью ключа выбранным методом и получает зашифрованное сообщение.

4) Зашифрованное сообщение пересылается абоненту №2.

5) Абонент №2 расшифровывает зашифрованное сообщение с помощью ключа и получает открытое сообщение.

Этот протокол достаточно прост, однако он может действительно использоваться на практике. Криптографические протоколы могут быть простыми и сложными в зависимости от назначения.

2 Обмен ключами по схеме Диффи-Хеллмана

2.1 Генерация большого простого числа

Любая криптосистема основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в системе, в которой количе-

ство пользователей составляет десятки и сотни, управление ключами – серьёзная проблема. Если не обеспечено достаточно надёжное управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации. В этом случае необходимо введение какой-либо случайной величины в процесс шифрования.

В частности, для реализации алгоритма RSA требуются большие простые числа. Где их взять?

Простых чисел не так мало, как кажется. Например, существует приблизительно 10^{151} простых чисел длиною от 1 бита до 512 включительно. Для чисел, близких к n , вероятность того, что выбранное число окажется простым, равна $\frac{1}{\ln n}$. Поэтому полное число простых чисел, меньших n, равно $\frac{n}{\ln n}$. Считается, что вероятность выбора двумя людьми одного и того же большого простого числа пренебрежимо мала.

Существуют различные вероятностные проверки чисел на простоту, определяющие с заданной степенью достоверности, является ли число простым. При условии, что эта степень достоверности велика, такие способы достаточно хороши. Такие простые числа часто называют «*промышленными простыми*», т.е. они просты с контролируемой возможностью ошибки.

Повсеместно используется алгоритм, разработанный Майклом Рабином по идеям Гари Миллера.

2.2 Тест Рабина-Миллера

Выбрать для проверки случайное число p[1,2]. Вычислить b как наибольшее число делений $p-1$ на 2, т.е. b – наибольший показатель степени числа 2, на которую делится $p-1$ без остатка.

Затем вычислить m такое, что $p = 1 + 2^b \cdot m$.

1. Выбрать случайное число a, меньшее p.

2. Установить $j = 0$ и $z = a^m \bmod p$.

3. Если $z = 1$ или $z = p - 1$, то p проходит проверку и может быть простым числом.

4. Если $j > 0$ и $z = 1$, то p не является простым числом.

5. Установить $j = j + 1$.

6. Если $j < b$ и $z < p - 1$, установить $z = z^2 \bmod p$ и вернуться на пункт 4.

7. Если $z = p - 1$, то p проходит проверку и может быть простым числом.

8. Если $j = b$ и $z \neq p - 1$, то p не является простым числом.

Повторить эту проверку нужно t раз.

Доказано, что в этом тесте вероятность прохождения проверки составным числом убывает быстрее, чем в прочих. Гарантируется, что 75 % возможных значений окажутся показателями того, что выбранное число p – составное. Это значит, что вероятность принять составное число p за простое не превышает величины $\left(\frac{1}{4}\right)^t$.

2.3 Алгоритм генерации простого числа

Сгенерировать случайное *n*-битное число p.

Установить его старший и младший биты равными 1. Старший бит будет гарантировать требуемую длину искомого числа, а младший бит – обеспечивать его нечётность.

Убедиться, что p не делится на небольшие простые числа: $3,\ 5,\ 7,\ 11$ и т.д. Наиболее эффективна проверка на делимость на все простые числа, меньшие 2000.

Выполнить тест Рабина-Миллера минимум 5 раз.

Если не прошло хотя бы одну проверку в пунктах 3 или 4, то оно не является простым.

Проверка, что случайное нечётное p не делится на 3, 5 и 7, отсекает 54% нечётных чисел. Проверка делимости на все простые числа, меньшие 256, отсекает 80% составных нечётных чисел.

Даже если составное число не было выявлено, это будет выявлено далее. Шифрование и дешифрование не будут работать[6, 7].

2.4 Построение первообразного корня по модулю n

В силу теоремы Эйлера для любых взаимно простых и выполняется соотношение:

$$a^{\varphi(n)} \equiv 1(\bmod n), \tag{7.1}$$

где $\varphi(n)$ обозначает функцию Эйлера, значение которой равно количеству положительных целых значений, меньших n и взаимно простых с n. Для простого числа p выполняется:

$$\varphi(p) = p - 1.$$

Если предположить, что два числа p и q – простые, тогда для $n = p^{\alpha} \cdot q^{\beta}$ функция Эйлера будет иметь вид:

$$\varphi(n) = \varphi(p^{\alpha} \cdot q^{\beta}) = \varphi(p^{\alpha}) \cdot \varphi(q^{\beta}) = \left[(p-1) \cdot p^{\alpha-1}\right] \cdot \left[(q-1) \cdot q^{\beta-1}\right]. \tag{7.2}$$

Рассмотрим более общее соотношение, чем (7.1).

Говорят, что число a, взаимно простое с модулем n, **принадлежит показателю** m, если m – такое наименьшее натуральное число, что выполняется сравнение:

$$a^{m} \equiv 1(\bmod n). \tag{7.3}$$

Если a и n – взаимно простые, то существует, по крайней мере, одно число $m = \varphi(n)$, удовлетворяющее (7.3). Наименьшее из положительных чисел m, для которых выполняется (7.3), является длиной периода последовательности, генерируемой степенями a.

Справедливы следующие свойства [6].

Свойство 1. Числа $a^0, a^1, \cdots, a^{m-1}$ попарно несравнимы по модулю n.

Свойство 2. $a^{\gamma} \equiv a^{\gamma'}(\bmod n) \Leftrightarrow \gamma \equiv \gamma'(\bmod m)$.

Доказательство: разделим γ и γ' на m с остатками: $\gamma = m \cdot q + r, \gamma' = = m \cdot q' + r'$. Тогда: $a^{\gamma} \equiv a^{\gamma'} \Leftrightarrow a^{m \cdot q + r} \equiv a^{m \cdot q' + r'} \Leftrightarrow a^{r} \equiv a^{r'} \Leftrightarrow r \equiv r'$.

Отсюда вытекает следующее свойство.

Свойство 3. Число a, принадлежащее показателю $\varphi(n)$, называется ***первообразным корнем*** по модулю n.

Свойство 4. По любому простому модулю p существует первообразный корень. Первообразные корни существуют по модулям 2, 3, p^{α}, $2 \cdot p^{\alpha}$, где $p-$ нечётное простое число, а $\alpha \in N$.

Свойство 5. Пусть $c = \varphi(n)$ и $q_1, q_2, \ldots, q_k$ – различные простые делители числа c. Число a, взаимно простое с модулем n, будет первообразным корнем тогда и только тогда, когда не выполнено ни одно из следующих сравнений: $a^{c/q_1} \equiv 1(\mod n), a^{c/q_2} \equiv 1(\mod n), \ldots, a^{c/q_k} \equiv 1(\mod n)$.

Доказательство: необходимость следует из того, что $a^{\varphi(n)} \equiv 1(\mod n)$ и сравнение не имеет места при меньших показателях степени.

Достаточность: допустим, что не удовлетворяет ни одному из сравнений и пусть a принадлежит показателю $m < c$. Тогда $m \mid c \Rightarrow c = m \cdot u$. Обозначим через q простой делитель u. Тогда легко получить противоречие:

$$a^{c/q} \equiv a^{mu/q} \equiv (a^m)^{u/q} \equiv 1(\mod n).$$

Если некоторая последовательность имеет длину $\varphi(n)$, тогда целое число a генерирует своими степенями множество всех ненулевых вычетов по модулю n. Такое целое число называют ***первообразным корнем по модулю n***. Для числа n их количество равно $\varphi(\varphi(n-1))$, где $\varphi()-$ функция Эйлера.

Пример:

Пусть $n = 41$. Имеем $c = \varphi(41) = 40 = 2^3 \cdot 5$. Итак, первообразный корень не должен удовлетворять двум сравнениям:

$$a^8 \equiv 1(\mathrm{mod}\,41),\ a^{20} \equiv 1(\mathrm{mod}\,41).$$

Испытываем числа $2,3,4,\ldots: 2^8 \equiv 10, 2^{20} \equiv 1, 3^8 \equiv 1, 4^8 \equiv 18, 4^{20} \equiv 1,$ $5^8 \equiv 18, 5^{20} \equiv 1, 6^8 \equiv 10, 6^{20} \equiv 40$.Отсюда видно, что $6-$ наименьший первообразный корень по модулю 41.

2.5 Алгоритм обмена ключами по схеме Диффи-Хеллмана

В 1976 году была опубликована работа молодых американских математиков У. Диффи и М.Э. Хеллмана «Новые направления в криптографии». В ней они предложили конкретную конструкцию так называемого «открытого распределения ключей»[7, 13].

Цель алгоритма состоит в том, чтобы два участника могли безопасно обменяться ключом, который в дальнейшем может использоваться в каком-либо алгоритме симметричного шифрования. Сам алгоритм Диффи-Хеллмана может применяться только для обмена ключами. Алгоритм основан на трудности вычислений дискретных логарифмов.

Безопасность обмена ключами в алгоритме Диффи-Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, но очень трудно вычислить дискретные логарифмы. Для больших простых чисел задача считается неразрешимой.

Предположим, что двум абонентам необходимо провести конфиденциальную переписку, а в их распоряжении нет первоначально оговоренного секретного ключа. Однако между ними существует канал, защищённый от модификации, т.е. данные, передаваемые по нему, могут быть прослушаны, но не изменены (такие условия имеют место довольно часто). В этом случае две стороны могут создать одинаковый секретный ключ, ни разу не передав его по сети, по следующему алгоритму (см. рисунок 7.1).

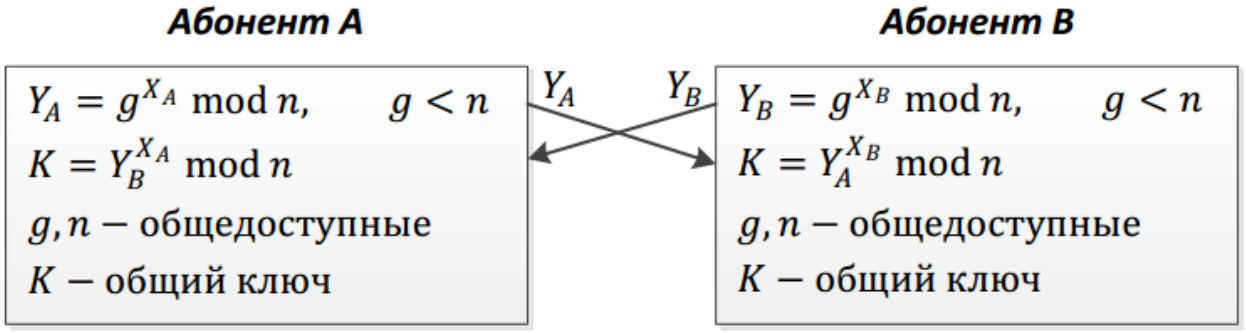

Рисунок 7.1 – Обмен ключами по схеме Диффи-Хеллмана

Алгоритм заключается в следующем:

1) Задаются глобальные открытые элементы:

n – случайное большое простое число;

g – первообразный корень n.

2) Вычисляется ключ абонентом *A*:

a) выбирается большое секретное число $X_A (X_A < n)$;

b) вычисление открытого значения Y_A : $Y_A = g^{X_A} \bmod n$.

3) Вычисляется ключ абонентом *B*:

a) выбирается большое секретное число $X_B (X_B < n)$;

b) вычисление открытого значения Y_B : $Y_B = g^{X_B} \bmod n$.

4) Вычисляется секретный ключ абонентом *A*: $K = Y_B^{X_A} \bmod n$.

5) Вычисляется секретный ключ абонентом *B*: $K = Y_A^{X_B} \bmod n$.

Необходимо ещё раз отметить, что алгоритм Диффи-Хеллмана работает только на линиях связи, надёжно защищённых от модификации. Если бы он был применим на любых открытых каналах, то давно снял бы проблему распространения ключей и, возможно, заменил бы собой всю асимметричную криптографию.

Пример:

Пусть $n = 97$ и $g = 5$. Абонент *A* сгенерировал случайное число $X_A = 36$. Абонент *B* сгенерировал случайное число $X_B = 58$. Эти элементы они держат в секрете. Далее каждый из них вычисляет новый элемент:

$Y_A = 5^{36} \bmod 97 = 50, \;\; Y_B = 5^{58} \bmod 97 = 44.$

Затем они обмениваются этими элементами по каналу связи. Теперь абонент *A*, получив Y_B и зная свой секретный элемент X_A, вычисляет общий ключ: $K_A = 44^{36} \bmod 97 = 75.$ Аналогично поступает абонент *B*: $K_B = 50^{36} \bmod 97 = 75.$

Задание

Вопросы для самоконтроля

8 ЭЛЕКТРОННО-ЦИФРОВАЯ ПОДПИСЬ

Цель: Изучить устройство и принцип работы цифровых ключей. Рассмотреть встроенные в систему ЭЦП и создать свою подпись.

Теоретические ведомости

SSL (Secure Socket Layer) – криптографический протокол, который подразумевает более безопасную связь. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений.

Цифровой сертификат – файл, который уникальным образом идентифицирует серверы. Обычно цифровой сертификат подписывается и заверяется специализированными центрами. Их называют центрами сертификации или удостоверяющими центрами.[12]

Протокол SSL обеспечивает защищённый обмен данных за счёт двух следующих элементов:

– Аутентификация;

– Шифрование.

SSL использует асимметричную криптографию для аутентификации ключей обмена, симметричный шифр для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. Протокол SSL предоставляет «*безопасный канал*», который имеет три основных свойства:

1) *Канал является частным.* Шифрование используется для всех сообщений после простого диалога, для определения секретного ключа.

2) *Канал аутентифицирован.* Серверная сторона диалога всегда аутентифицируется, а клиентская делает это опционально.

3) *Канал надёжен*. Транспортировка сообщений включает в себя проверку целостности.

1 Принцип работы сертификата

Рисунок 8.1 –Предоставление доступа в SSL-соединения

Для доступа защищённого доступа к сайту пользователь совершает шаги, изображённые на рисунке 8.1:
– Пользователь заходит на защищённый сайт;
– Выполняется проверка DNS и определение IP-адреса хоста веб-сайта;
– Запись веб-сайта найдена, переход на веб-сервер хоста;
– Запрос безопасного SSL-соединения с хоста веб-сайта;
– Хост отвечает валидным SSL-сертификатом;
– Устанавливается защищённое соединение, передаваемые данные шифруются.

2 Виды электронной подписи

Существует три вида ЭП. Самая обычная, которой мы часто пользуемся, не имеет тех степеней защиты информации, как две другие – Усиленные. Они различаются в статусе и область их применения неодинакова. Разберемся в их отличиях:

Простая ЭП предполагает использование логина и пароля. При доступе к услугам, для подтверждения операции, может запрашиваться одноразовый код, отправляемый через CMS-сообщение либо почту. С подобными видами идентификации приходится сталкиваться часто. Для этого не надо обращаться в специализированные центры.

Усиленная неквалифицированная подпись – этот атрибут не только идентифицирует отправителя, но и фиксирует изменения подписанного документа. Получают УНП в удостоверяющем центре. Область применения НЭП ограничена. Государственные и муниципальные документы, содержащие тайну, подписывать ею нельзя.

Усиленная квалифицированная ЭП имеет самую высокую степень защиты на законодательном уровне. Электронные документы приравниваются к бумажным со всеми атрибутами визирования, имеют такую же юридическую силу. В сертификате, который выдается вместе с ключом, содержится информация по его проверке. Чтобы проводить юридически-значимые операции, необходимо использование этого ключа (подписи)[4].

Задание

Для выполнения заданий по аутентификации посредством сертификатов X.509 следует подробно ознакомиться с историей, принципом работы, актуальной версией, а также применением в сети. Некоторая информация приведена в литературе[1, 7–9, 12].

Выполнение работы проходит в командной строке(PowerShell), используем привилегированный режим. Выполнение работы альтернативными методами не запрещается.

1) Изучить содержание сервиса «Управление сертификатами пользователей».

2) Создать самоподписанный(корневой) сертификат.

3) Создать дочерний сертификат.

4) Экспорт публичных ключей сертификата.

5) Экспорт приватного ключа дочернего сертификата.

6) Написать отчёт, где провести анализ и сравнение корневого сертификата, сертификата клиента и их зависимости.

Ход работы

1 Управление сертификатами пользователей

Данный сервис включён в состав Windows. Открыть его можно из «Панели управления», открыв раздел «Управление сертификатами пользователей».

Альтернативным способом запуска является запуск через командную строку. Написав команду *«certmgr»* перед нами будет отображено окно, рисунке 8.2

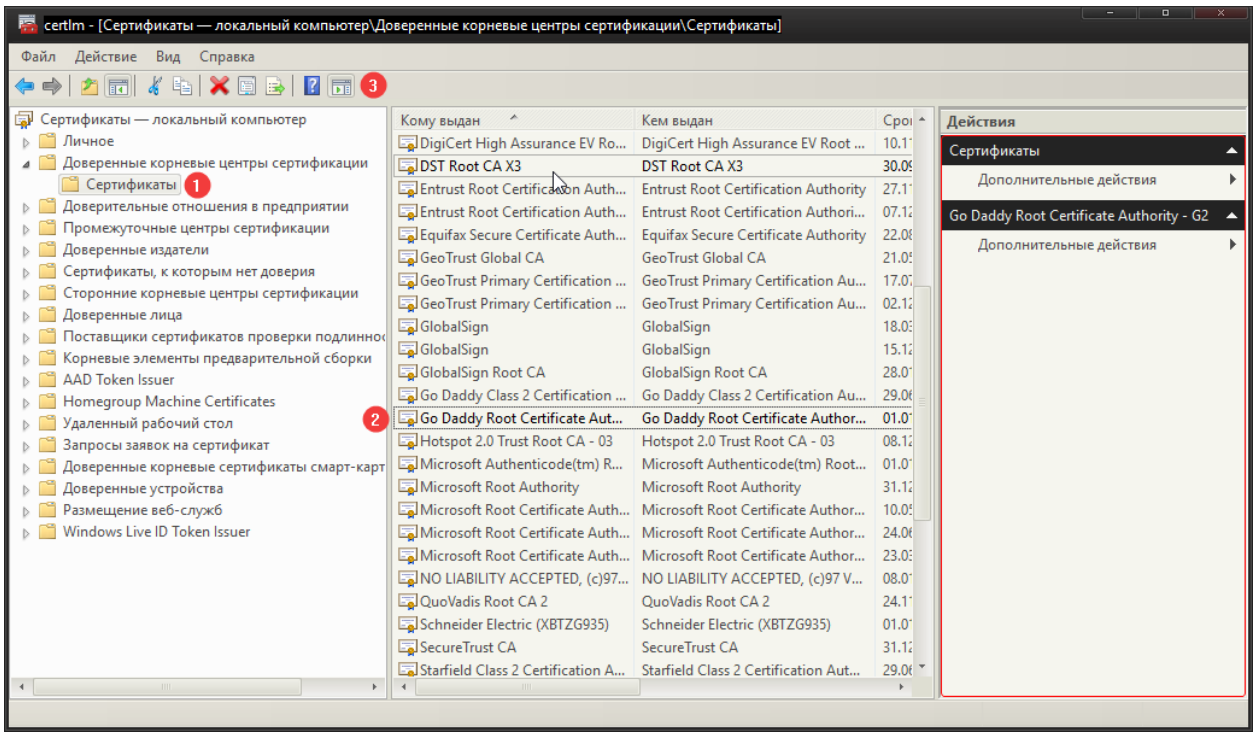

Рисунок 8.2 –Окно «управление сертификатами пользователей»

где:

1 – дерево папок, в котором выбраны корневые сертификационные центры;

2 – выбран сертификат «Go Daddy», является одним из самых больших СЦ;

3 – панель дополнительных параметров, для быстрого доступа к действиям.

В свойствах сертификата указано множество информации, основные данные это: кому выдан сертификат; кем выдан; период активности сертификата; данные о субъекте; открытый ключ и надёжность ключа.

Свойства сертификата показаны на рисунке 8.3.

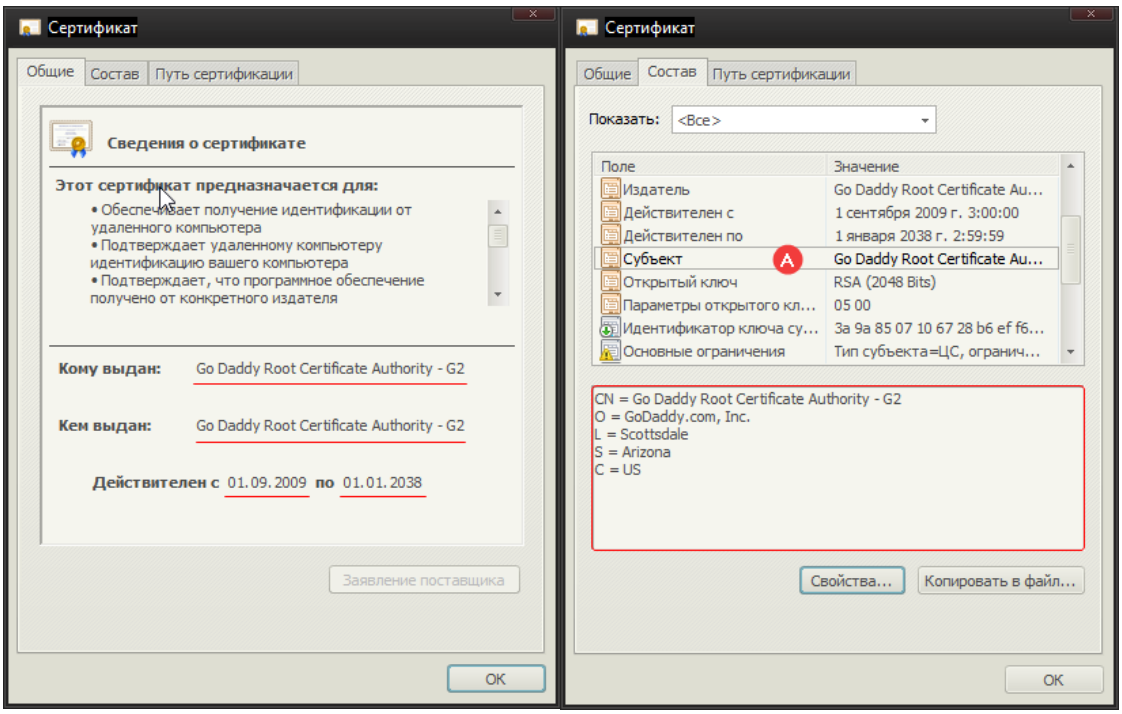

Рисунок 8.3 –Свойства сертификата

2 Создание корневого сертификата

Данный вид сертификатов ещё называют «самоподписными».

1) Открыть окно управления сертификатами пользователя, выбрать папку «Личное» → «Сертификаты». С помощью данного окна мы будем проверять созданные сертификаты.

2) Запустить «PowerShell ISE» с привилегиями администратора. Перед нами будет представлено окно 8.4. Чтобы выполнять несколько команд вместе используются скрипты. Для создания скрипта нужно в меню «Файл» выбрать «Создать». Для выполнения скрипта «Файл» → «Выполнить».

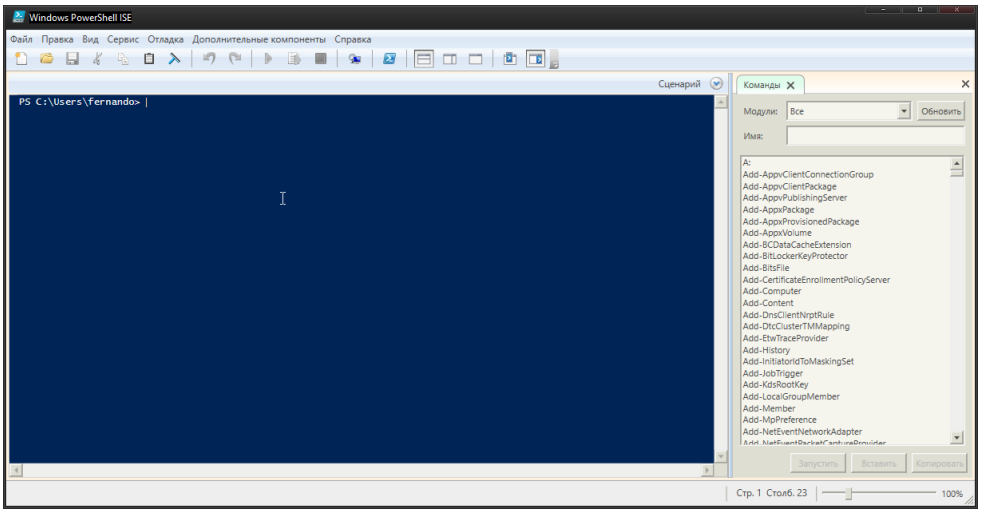

Рисунок 8.4 –Окно PowerShell ISE

3) После чего в области ввода записать:

```
New-SelfSignedCertificate -Type Custom -KeySpec Signature `
-Subject "CN=P2SRootCert" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `
-CertStoreLocation Cert:\CurrentUser\My -KeyUsageProperty Sign `
-KeyUsage CertSign
```

Таким образом мы создаём новый корневой сертификат «P2SRootCert», который устанавливается в папку «Личное» → «Сертификаты». Длина ключа 2048 Bits. Действителен год с даты создания. Со всеми ключами команды *«New-SelfSignedCertificate»* можно ознакомиться на странице документации[10].

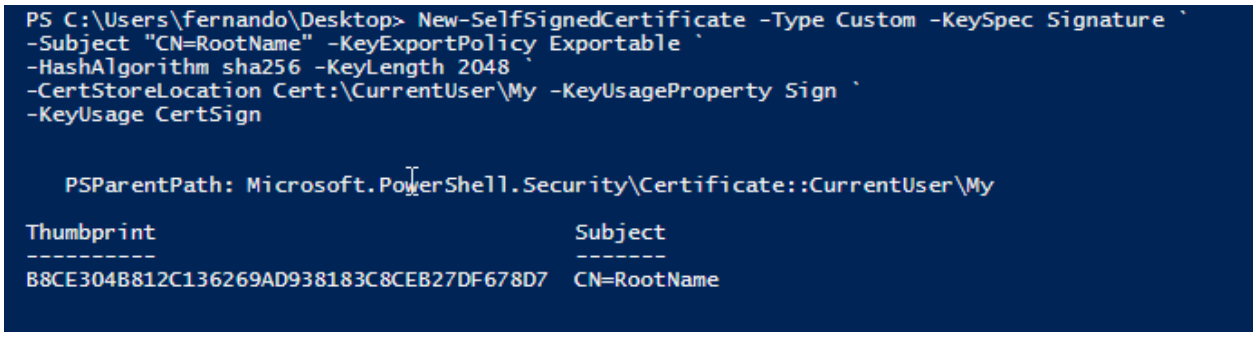

Рисунок 8.5

4) Чтобы получить список сертификатов, установленных на компьютере, нужно использовать команду *«Get-ChildItem»*. Используя параметр *«-Path»* можно указать поиск в определённом каталоге. Давайте отобразим все сертификаты пользовательской папке:

```
Get-ChildItem -Path "Cert:\CurrentUser\My"
```

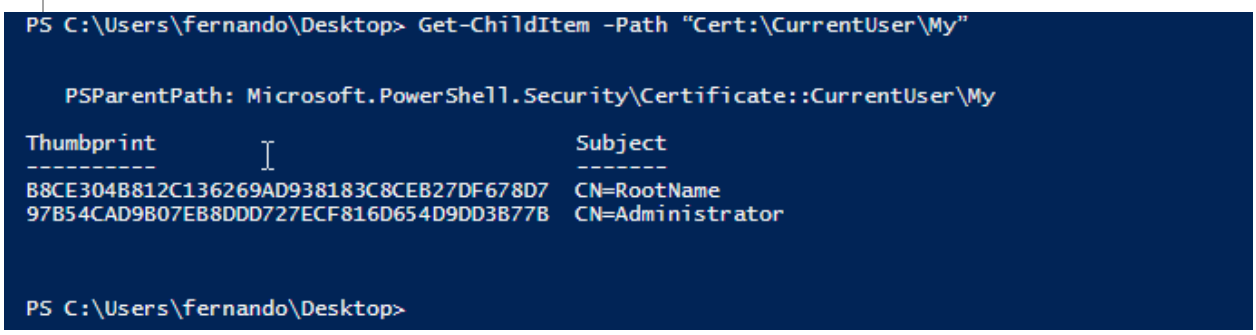

Рисунок 8.6

В результате мы получаем список из отпечатка и имени.

5) Объявите переменную для корневого сертификата, используя отпечаток из предыдущего шага. Замените THUMBPRINT отпечатком корневого сертификата, на основе которого требуется создать дочерний сертификат.

```
1 | $cert = Get-ChildItem -Path "Cert:\CurrentUser\My\B8CE304B...8D7"
```

или можно вынести отпечаток в отдельную переменную, как показано ниже:

```
2 | $ThumbPr = "B8CE304B812C136269AD938183C8CEB27DF678D7"
3 | $cert = Get-ChildItem -Path "Cert:\CurrentUser\My\$ThumbPr"
```

6) Если выполнить этот пример, не изменив его, то будет создан сертификат клиента *P2SChildCert*. Если требуется указать другое имя дочернего сертификата, измените значение *CN*. Не изменяйте *TextExtension* при выполнении данного примера. Сертификат клиента, который создается, автоматически устанавливается в папку:
«Сертификаты–текущий пользователь» → *«Личное»* → *«Сертификаты»*.

3 Экспорт открытого ключа

1) Чтобы экспортировать CER-файл для корневого корневого сертификата, откройте раздел *«Управление сертификатами пользователей»*. Найдите корневой самозаверяющий сертификат и щелкните его правой кнопкой мыши. Щелкните *«Все задачи»* → *«Экспорт»*. Откроется мастера экспорта сертификатов (рис. 8.7).

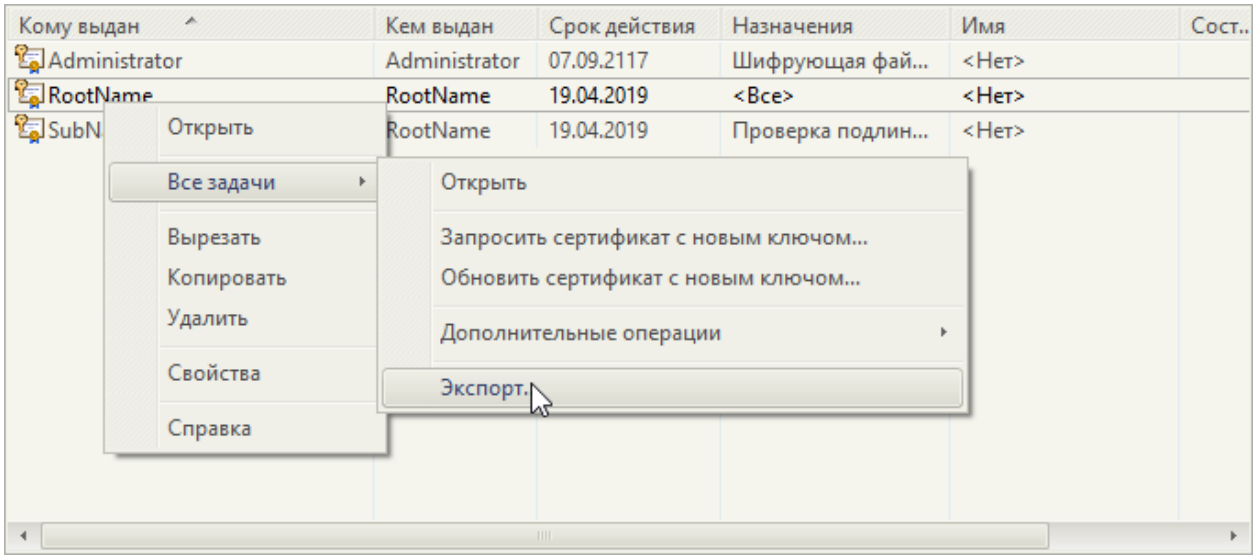

Рисунок 8.7 –Опции экспорта сертификата

Видим окно, изображённое на рисунке 8.8.

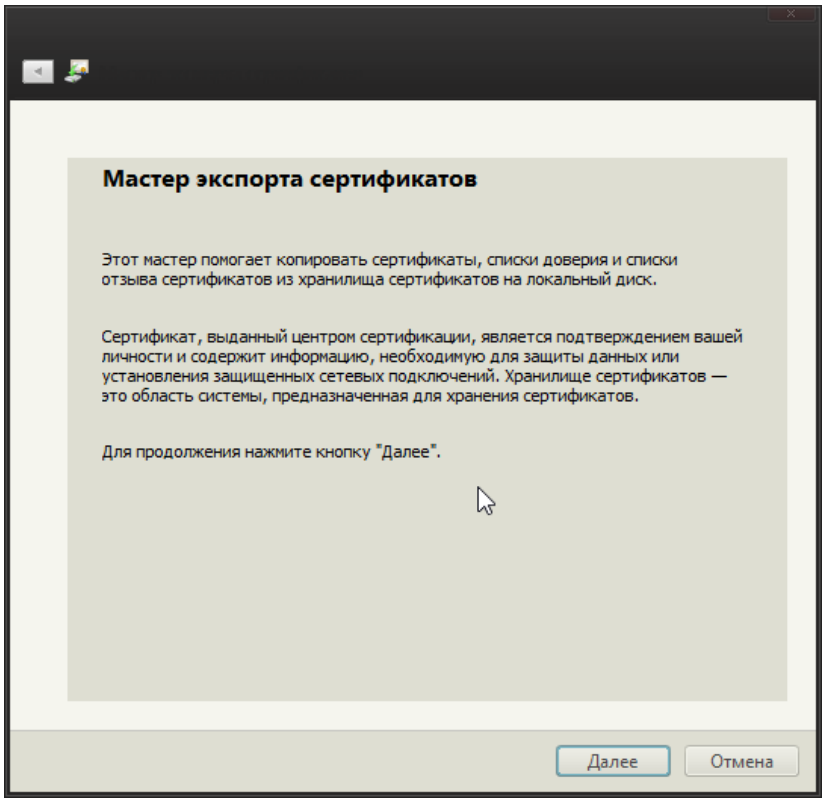

Рисунок 8.8 –Окно мастера экспорта сертификатов

2) На следующем окне выбираем «Нет, не экспортировать закрытый ключ»

3) В данном примере мы будем экспортировать файл в кодировке *Base-64*.

4) В следующем окне требуется выбрать место сохранения сертификата и указать имя.

После создания экспорта открытого ключа сертификата мы можем открыть его в текстовом редакторе, например, в «Блокноте» или «Notepad++». Результат будет приблизительно таким, как изображено на рисунке 8.9.

```
-----BEGIN CERTIFICATE-----
MIIC4TCCAcmgAwIBAgIQZLEMAOUXh6JDlOBbUvUXyDANBgkqhkiG9w0BAQsFADAT
MREwDwYDVQQDDAhSb290TmFtZTAeFw0xODA0MTkxNDU2NDBaFw0xOTA0MTkxNTE2
NDBaMBMxETAPBgNVBAMMCFJvb3ROYW1lMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAr5ElUErsn2amDrD48NlmZWhjHvjWBVQEaZ23M0QyHaRHEfM/xPM4
DBbiCnh43mpJo1FhDBt18vU4LfBscjrN9bnxEm9i7TyK2pxTHJ5UnMK1PzraDIp7
GLnrd9ZHsWPLQF9Lx1/Cb1QIGcSZmL9L6Hwx7atKdL0JnzGDYYmBrX7mUes+6ID5
mCOCK4aTV6Sq8fxxpYnmsjsICz5kAk9QXqupQoVfWgYP3YpO5KK0NVT52LiEPOBn
+RH1TKKYKDmLNcUnexc8ESMJkRrHbElWxz8+r246zhiozeK14RDlRWdAsG7vHhMT
l0nQT7TNKqG3zUQyadZ7zSEuyD04mJj4WQIDAQABozEwLzAOBgNVHQ8BAf8EBAMC
AgQwHQYDVR0OBBYEFFn+xKv/c+d7Fjfi/3fe+AjHdZwCMA0GCSqGSIb3DQEBCwUA
A4IBAQBSA4wM3MEiQx76dJ41vjXyMtLW+V0/spTs/oZeqfFX9MTv+x8VtRu+v+DF
5Jbk5svhRmW/ZhkDp0t+SZvihy5EkxsjzHVbtN5mYrKCBIV9QqL+I5VXdfg5eMDp
gsW3pMjPdRvLta2Ef4Y9T/c1ZsU8KvL4CmyQ+nYcHe8eZ2s/8RW6hvVB4KqCYNa/
Yt4o42Tu5EYe1Ee4L7Cp78Wp2mgXArzjIoEP0j2GmvikFqle/CcFzLW9K6f5wlhi
YJC2EiYIZAEtFq6jsyYXlFepbL6X20j/xIeBMLbISaXBm160O8PhU9hP38r84jHP
co8LYj5ZzkJcJ8+xPZHsl0m5JpUs
-----END CERTIFICATE-----
```

Рисунок 8.9 –Содержимое сертификата открытого в Notepad++

4 Экспорт приватного ключа дочернего сертификата

1) Для экспорта выбираем требуемый сертификат, в примере это *«SubName»*. Так же, как и в прошлом задании выбираем «*Экспорт…»*.

2) Выбираем пункт «Да, экспортировать закрытый ключ».

3) Включаем *«конфиденциальность сертификата»*, рисунок 8.10.

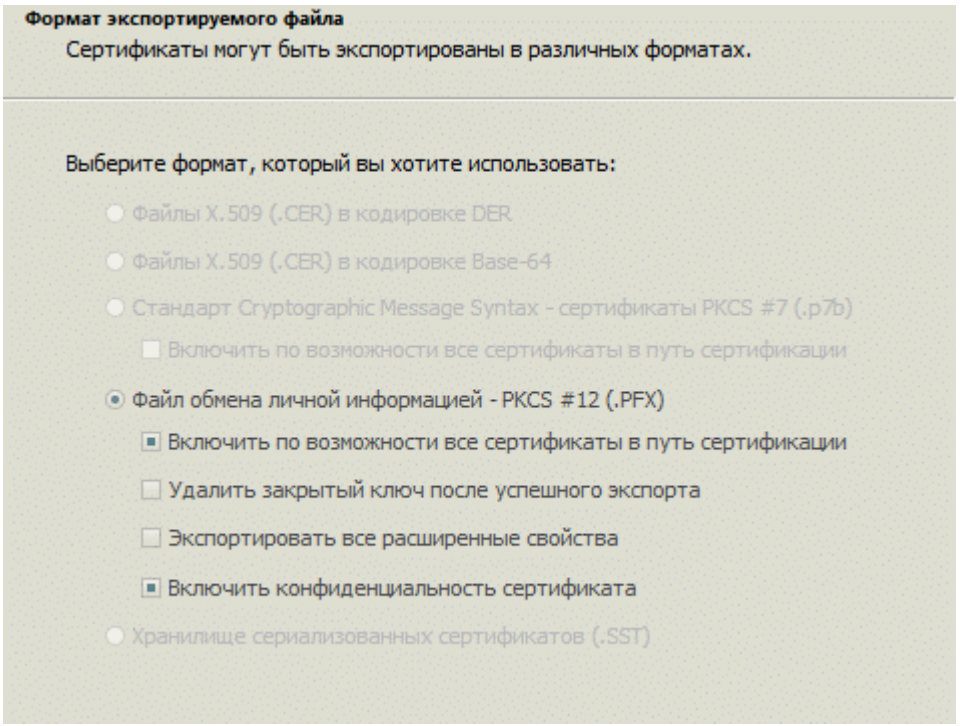

Рисунок 8.10 – Параметры экспорта сертификата с приватным ключом

4) Добавляем способ защиты. Можно использовать групповую политику пользователей или же пароли. Первый вариант удобен, если заранее известны клиенты. В нашем случае используем пароль для доступа к сертификату.

После выполнения всех действий мы получим два файла. Публичный(.cer) и приватный(.pfx), изображены на рисунке 8.11. Публичный используется для шифрования данных и распространения в Интернете. Приватный же сохраняется только у лиц(владельцев) для дешифровки файлов.

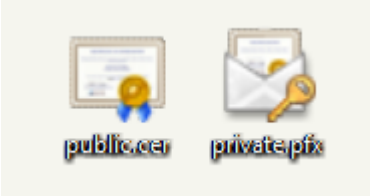

Рисунок 8.11 – Иконки сертификатов

Вопросы для самоконтроля

1) Что такое ЭЦП?

2) Из чего состоит электронная подпись?

3) В каких сферах используется ЭЦП? Приведите примеры.

4) Опишите принцип работы электронной подписи.

5) Зачем нужны публичный и приватный ключи? В чём их различие?

6) Что такое «корневой» сертификат?

7) Центр сертификации – это?

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комарова А.В., Менщиков А.А., Коробейников А.Г. Анализ и сравнение алгоритмов электронной цифровой подписи ГОСТ р 34. 10-1994, ГОСТ р 34. 10-2001 и ГОСТ р 34. 10-2012 // Вопросы кибербезопасности. 2017. № 1 (19).

2. НГТУ Гаммирование. Моделирование работы скремблера // Лабораторные работы. 2012. С. 7.

3. НОУ «ИНТУИТ» Лекция 4: Методы криптоанализа // 03.2015 [Электронный ресурс]. URL: https://www.intuit.ru/studies/courses/600/456/lecture/10198 (дата обращения: 08.04.2018).

4. ПОРТАЛ GOSUSLUGI.RU Как создать и получить электронную подпись для госуслуг | InfoGosuslugi.ru // 2016-11-04 [Электронный ресурс]. URL: http://infogosuslugi.ru/portal-gosuslugi/elektronnaya-podpis-ecp-dlya-gosuslug-sozdanie-i-poluchenie.html (дата обращения: 18.04.2018).

5. Столлингс В. Криптография и защита сетей. Принципы и практика / В. Столлингс, М.: Издательский дом «Вильямс», 2001. 672 с.

6. Харин Ю.С. [и др.]. Математические и компьютерные основы криптологии // Минск: Новое знание. 2003.

7. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер, М.: Триумф, 2002. 816 с.

8. Freier A., Karlton P., Kocher P. The secure sockets layer (SSL) protocol version 3.0 2011.

9. Microsoft Get-ChildItem для Certificate // 2014-10 [Электронный ресурс]. URL: https://technet.microsoft.com/ru-ru/library/hh847761.aspx (дата обращения: 18.04.2018).

10. Microsoft New-SelfSignedCertificate [Электронный ресурс]. URL:

https://docs.microsoft.com/en-us/powershell/module/pkiclient/new-selfsignedcertificate?view=win10-ps (дата обращения: 19.04.2018).

11. PUB F. DES Modes of Operation 1980.

12. REG.RU Цифровой сертификат безопасности: для чего это нужно? / Блог компании REG.RU / Хабрахабр // 2016-04-04 [Электронный ресурс]. URL: https://habrahabr.ru/company/regru/blog/280878/ (дата обращения: 18.04.2018).

13. Rescorla E. Diffie-hellman key agreement method 1999.

14. Wiki inform // 08.01.2018 [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Frequency_analysis (дата обращения: 07.04.2018).

ДОДАТОК Б