Название учреждения, в котором выполнялась данная диссертационная работа


На правах рукописи

Фамилия Имя Отчество

# Название диссертационной работы

Специальность XX.XX.XX ”—
«Название специальности»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
уч. степень, уч. звание
Фамилия Имя Отчество

Город ”— 20XX

# 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.

# ВСТУП

Использовать можно в двух системах:

Первый вариант — это выполняются первые 8 работ и получают нужную оценку.

Второй вариант — каждое задание добавляет балы, общая сумма балов определяет итоговую оценку. (Данная система более правильна и гибка, но требует набирать балы за работу)

# ЛАБОРАТОРНА РОБОТА 1

# ОСНОВЫ ШИФРОВАНИЯ. СИММЕТРИЧНЫЕ ШИФРЫ

**Мета роботи:** Изучить основные типы шифров. Исследование симметричных методов шифрование. Получение практических навыков изученных методов шифровки сообщений.

## 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 Skype, 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) и ассиметричным (с открытым ключом).

**1.1. Симметричный метод.** 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.

**1.2. Асимметричный метод.** 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.

**1.3. Блочные шифры.** 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 *(block size)*.

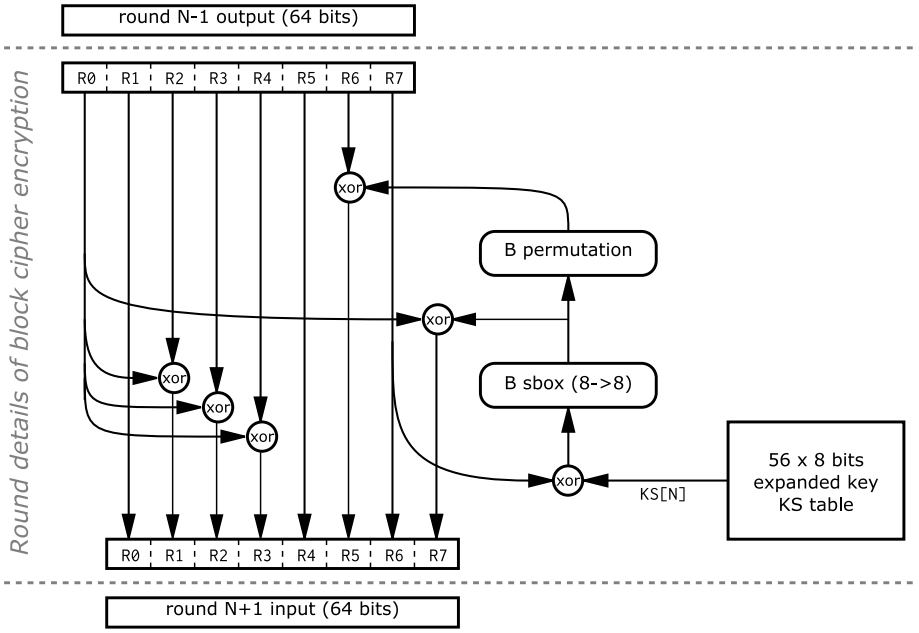


Рисунок 1.1 — Подробный разбор алгоритма блочного скремблирования DVB

**1.4. Потоковые шифры.** Потоковые шифры представляют собой разновидность гаммирования и преобразуют открытый текст в шифрованный

последовательно по 1 биту. Генератор ключевой последовательности выдаёт последовательность бит $k_1, k_2, \ldots, k_i, \ldots$ Эта ключевая последовательность складывается по модулю 2 с последовательностью бит исходного текста $p_1, p_2, \ldots, p_i, \ldots$ для получения шифрованного текста:

$$c_i = p_i \oplus k_i; \tag{1.1}$$

На приемной стороне шифрованный текст складывается по модулю 2 с идентичной ключевой последовательностью для получения исходного текста:

$$c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i; \tag{1.2}$$

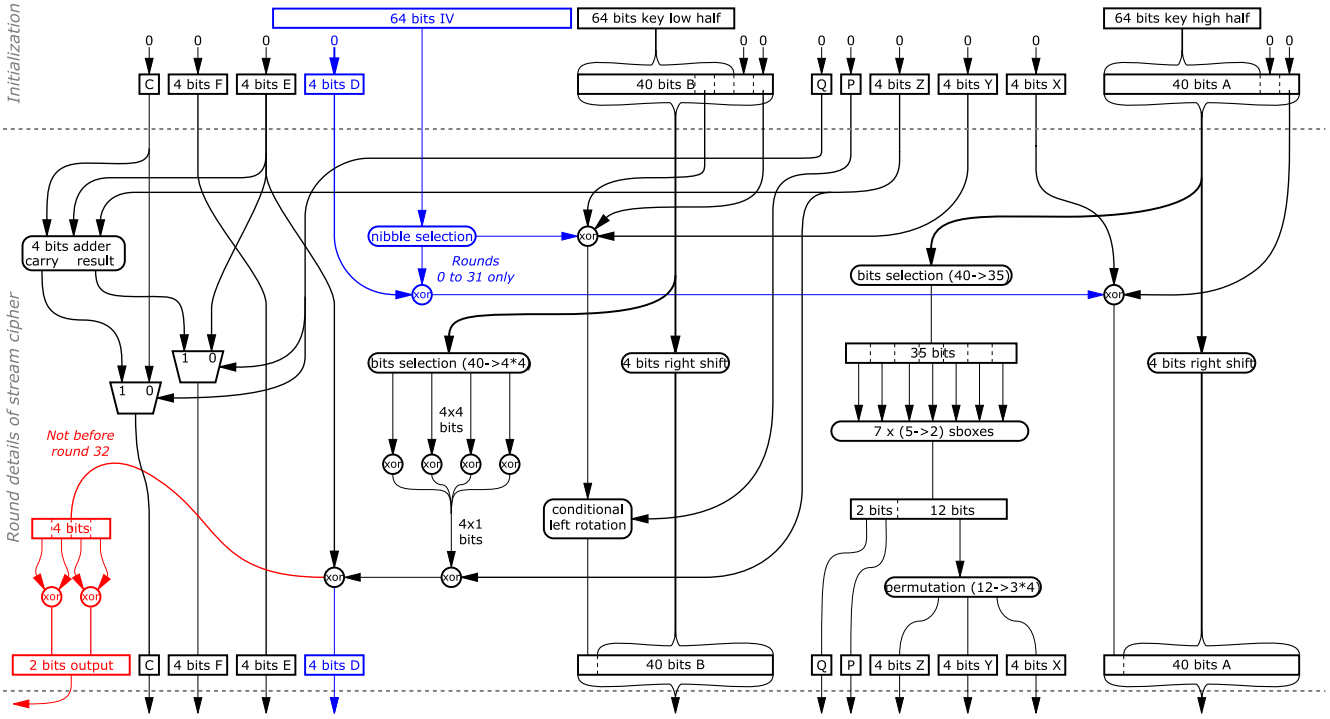


Рисунок 1.2 — Подробный разбор алгоритма потокового скремблирования DVB

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.

**1.5. 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. Сферы применения криптографии.

В настоящее время криптография прочно вошла в нашу жизнь. Перечислим лишь некоторые сферы применения криптографии в современном информатизированном обществе:

- шифрование данных при передаче по открытым каналам связи (например, при совершении покупки в Интернете сведения о сделке);
- обслуживание банковских пластиковых карт;
- хранение и обработка паролей пользователей в сети;
- сдача бухгалтерских и иных отчётов через удалённые каналы связи;
- банковское обслуживание предприятий через локальную или глобальную сеть;
- безопасное от несанкционированного доступа хранение данных на жёстком диске компьютера.

Средства обеспечения информационной безопасности можно разбить на четыре группы:

- организационные средства (действия общего характера, предпринимаемые руководством организации, и конкретные меры безопасности, имеющие дело с людьми);
- законодательные средства (стандарты, законы, нормативные акты и т.д.);
- программно-аппаратные средства (системы идентификации и аутентификации; системы шифрования дисковых данных;
- системы аутентификации электронных данных и т.д.);
- криптографические средства (электронная цифровая подпись, шифрования, аутентификация и др.).

## 3. 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.

## Задания

Выполнение первой работы предусматривает две части. Нужно реализовать шифровку ручными методами, не используя ПО. Вторая часть предусматривает использование специализированного ПО. После выполнения обеих частей требуется сравнить результате, записать выводы.

Выполнение действий должно сопровождаться соответствующими заметками в отчёте.

Уровень 1.

1. Зашифровать 5-ю различными методами свои ФИО.
2. Приложить ключи к отчёту.
3. Составить инструкцию для расшифровки.
4. Выбрать данные соответствующие варианту из табл.
5. Используя данные по варианту расшифровать криптотекст.
6. Записать результаты в отчёт.

Уровень 2.

1. Используя ключ из табл. определить шифр.
2. Написать алгоритм шифровки-дешифровки заданным способом. Алгоритм должен быть представлен в виде подробной блок-схемы.

## Пример выполнения работы

## Вопросы для контроля

1. Криптография и её роль в обществе.
2. Объяснить цель и задачи криптографии.
3. Пояснить какие бывают криптографические методы.
4. Виды криптографии и их классификация.
5. Отличие симметричных и асимметричный шифров.
6. Пояснить что такое исходный текст, шифр, ключ.

7. Принцип подбора ключа в симметричных криптосистемах.
8. Принцип работы симметричных шифров. Приведите примеры.
9. Принцип работы асимметричных шифров. Приведите примеры.
10. Преимущества и недостатки симметричный систем.
11. Шифры одиночной перестановки и перестановки по ключевому слову. Шифр Гронфельда.
12. Шифры двойной перестановки. Шифрование с помощью магического квадрата.
13. Шифр многоалфавитной замены и алгоритм его реализации.
14. Пояснить алгоритм шифрации двойным квадратом. Шифр Enigma.

# ЛАБОРАТОРНА РОБОТА 2

# РАБОТА С PKI

**Мета роботи:** Изучить методы генерации простых чисел, проверку на простоту числа и реализовать алгоритм Диффи-Хелмана.

## Теоретические ведомости

Асимметричные криптографические системы были разработаны в 1970-х гг. Принципиальное отличие асимметричной криптосистемы от криптосистемы симметричного шифрования состоит в том, что для шифрования информации и её последующей расшифровки используются различные ключи: открытый ключ К

1. используется для шифрования информации, вычисляется из секретного ключа к;

2. секретный ключ к используется для расшифровки информации, зашифрованной с помощью парного ему открытого ключа К.

Эти ключи различаются таким образом, что с помощью вычислений нельзя вывести секретный ключ к из открытого ключа К. Поэтому открытый ключ К может свободно передаваться по каналам связи. Асимметричные системы называют также двухключевыми криптографическими системами, или криптосистемами с открытым ключом.

Обобщенная схема асимметричной криптосистемы шифрования с открытым ключом показана на рис ниже

Для криптографического закрытия и последующего расшифровки передаваемой информации используются открытый и секретный ключи получателя В сообщения. В качестве ключа зашифровки должен использоваться открытый ключ получателя, а в качестве ключа расшифровки — его секретный ключ.

Секретный и открытый ключи генерируются попарно. Секретный ключ должен оставаться у его владельца и быть надежно защищен от НСД (аналогично ключу шифрования в симметричных алгоритмах). Копия открытого ключа должна находиться у каждого абонента криптографической сети, с которым обменивается информацией владелец секретного ключа.

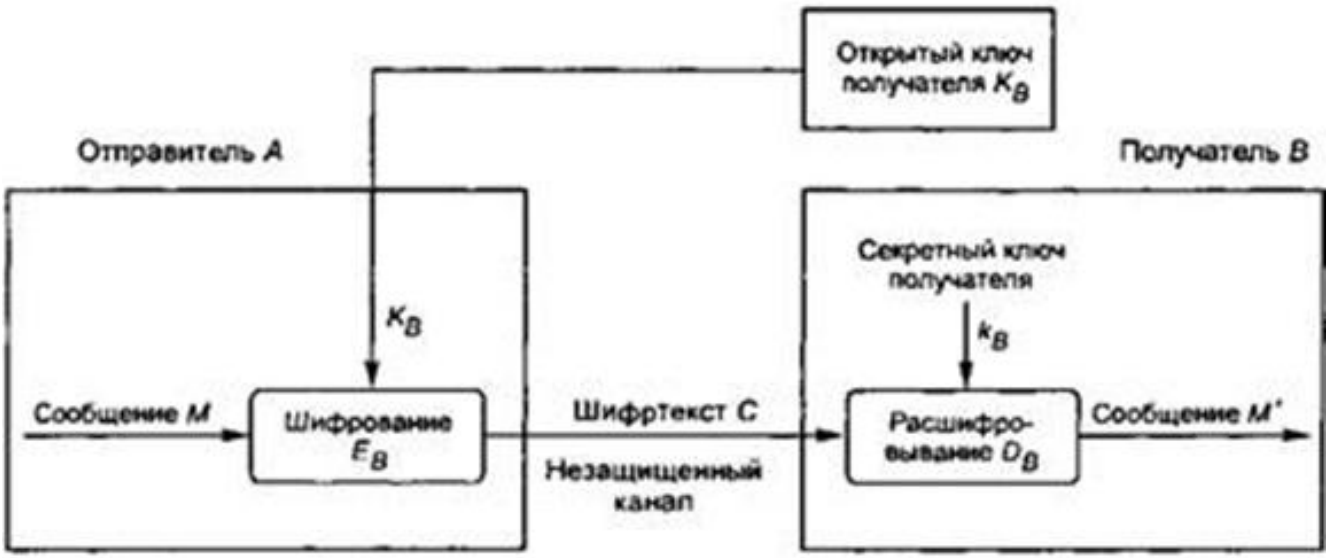


**Рис. 5.3. Обобщенная схема асимметричной криптосистемы шифрования**

Рисунок 2.1 — Обобщённая схема асимметричной криптосистемы шифрования

**1. Передача ключа**

Процесс передачи зашифрованной информации в асимметричной криптосистеме осуществляется следующим образом.

**1.1. Подготовительный этап.**

– абонент В генерирует пару ключей: секретный ключ кв и открытый ключ Кв;
– открытый ключ Кв посылается абоненту А и остальным абонентам (или делается доступным, например на разделяемом ресурсе).

**1.2. Обмен информацией между абонентами А и В.**

– абонент А шифрует сообщение с помощью открытого ключа Кв абонента В и отправляет шифротекст абоненту В;
– абонент В расшифровывает сообщение с помощью своего секретного ключа кв.

Никто другой (в том числе абонент А) не может расшифровать данное сообщение, так как не имеет секретного ключа абонента В. Защита информации в асимметричной криптосистеме основана на секретности ключа кв получателя сообщения.

**1.3. Характерные особенности асимметричных криптосистем.**

– открытый ключ Кв и криптограмма С могут быть отправлены по незащищенным каналам, т. е. противнику известны Кв и С;

– алгоритмы шифрования и расшифровывания: Ев : M → C; DB : C → M являются открытыми.

## 2. Методы генерации простых чисел

Задача генерации простых чисел относится к классическим вычислительным проблемам, известным с античных времен. С появлением криптографии с открытым ключом эта проблема приобрела не только теоретическое, но и большое практическое значение, так как большие случайные простые числа служат параметрами (открытыми или закрытыми) многих асимметричных криптосистем. Поскольку «формула» построения простого числа не известна (хотя её поисками занимались многие выдающиеся математики), то все методы генерации простых чисел действуют по следующей схеме: строится случайное нечетное число n нужной длины и затем проверяется на простоту одним из известных тестов.

В 1976 г. Х. Уильямс привел следующую классификацию методов проверки чисел на простоту (судя по более поздним публикациям, эта классификация не потеряла актуальности по настоящее время):

Детерминированные методы, использующие специальные функции и/или знание полного или частичного разложения числа n – 1 на простые множители. Эти методы математически строго доказывают простоту или составность проверяемого числа n. Методы Монте-Карло (вероятностные) – не требуют разложения на простые множители, работают очень быстро, но не дают строгого доказательства простоты (составности) числа. Методы гипотез – промежуточные между методами Монте-Карло и математически строгими методами. Основаны на истинности какой-либо недоказанной пока гипотезы (например, обобщённой гипотезы Римана): в предположении, что гипотеза верна, эти методы превращаются в детерминированные. Все детерминированные методы, которые были известны до августа 2002 г., имеют экспоненциальную оценку сложности. В работе индийских математиков M. Agrawal, N. Kayal, N. Saxena с замечательным названием «PRIMES is in P» представлен детерминированный полиномиальный тест на простоту; однако на данный момент вследствие своей сложности этот тест имеет скорее теоретическое, чем практическое значение.

## 3. Известные алгоритмы

Извесные алгоритмы сегодня

## 4. Алгоритма Диффи-Хелмана

История, цель и важность

### 4.1. Принцип работы алгоритма. Отобразить принцип работы алгоритма

### 4.2. Пример реализации. блок схема(как там мороки много)

## Задания

Стандартный алгоритм для выполнения работы — Алгоритм Диффи-Хелмана. Не запрещено использовать более новые алгоритмы, соответствующие тем же принципам.

Вариант задания указан в табл..

Уровень 1.

1. Описать последовательность выполнения алгоритма для шифрования задания по варианту указаному выше.
2. Исследовать использование алгоритма в различных сферах ИБ. Указать реальное применение данного алгоритма.

Уровень 2.

1. Составить программу для реализации алгоритма ДХ.
2. Произвести поэтапный обмен между двумя клиентами.
3. Предоставить все промежуточные данные, как на табл.
4. Продемонстрировать шифрованный блок и записать его в отчёт.

Уровень 3.

1. Разработать приложение клиент-клиент обмена информацией между несколькими людьми. Используя асинхронный алгоритм шифрования.
2. Продемонстрировать выполнение приложения.
3. Предоставить исходный код в дополнение отчёта.

Приложение может быть составлено в любом варианте, например: онлайн передача, запись в шифрованный файл, почтовый шифр и другие.

## Пример выполнения работы

## Варианты

## Вопросы для контроля

1. Асимметричные системы.
2. Привести примеры асимметричных алгоритмов.
3. Криптосистемы. Использование асимметричный алгоритмов.
4. Преимущества и недостатки асимметричный систем.
5. Методы подбора ключа в асимметричных криптосистемах.
6. Сферы использование асимметричных алгоритмов.
7. Описать однонаправленные функции. Привести примеры.

# ЛАБОРАТОРНА РОБОТА 3

# CISCO. ТОПОЛОГИЯ СЕТЕЙ

**Мета роботи:** Напомнить основные концепции сети, их связи, уязвимости и основные узлы.

## Теоретические ведомости

### 1. Информационная среда

Сеть со стороны ИБ.

Основые «дыры» в сети

### 2. Пути несанкционированного доступа

**2.1. Средства защиты информации.** (всяко-разно)Ссылка на вики по этой теме

– Маршрутизатор как способ защиты. (обязательно включить) - Фильтрование сети

- экранирование сети

Защита информации в сети Интернет:

## Задания

Уровень 1.

1. Изучить предложенную топологию в Cisco. Представлена на рис.
2. Изменить модель по варианту.
    а) Построить реализацию заданной сети в Cisco или другом ПО.
    б) Указать недостатки заданной системы.

Уровень 2.

1. Настроить маршрутизатор по варианту.
    а) Шаг 1.
    б) Шаг 2.
    в) Шаг 3.
    г) Шаг 4.
    д) Шаг 5.
    е) Шаг 6.

ж) Шаг 7.

и) Шаг 8.

2. Разделение подсети, протокол RIP

## Пример выполнения работы

## Варианты

## Вопросы для контроля

1. Какие есть средства защиты частной сети?
2. Что такое шлюз сетевого уровня?
3. Преимущества и недостатки использования сетевых экранов.
4. Что такое списки доступа? Каковы их цели и применение?
5. Приведите пример списка доступа.
6. Что такое маска подсети?
7. Каков принцип маски и как происходит фильтрация пакетов

# ЛАБОРАТОРНА РОБОТА 4

# ЭЛЕКТРОННО-ЦИФРОВАЯ ПОДПИСЬ

**Мета роботи:** Принцип работы, особенности, создание и использование ЭЦП.

## Теоретические ведомости

В современном мире происходит постепенная замена бумажной технологии обработки информации её электронным аналогом. Всё большее распространение получают автоматизированные системы обработки информации. Тенденция ведет в будущем к полной замене бумажного документооборота электронным. Однако защитных атрибутов бумажных документов, таких как подписей, печатей, водяных знаков и специальной фактуры поверхности, - у электронного представления документа нет. Поэтому возникла задача разработки такого механизма электронной защиты, который смог бы заменить подпись и печать на бумажных документах. Во-первых, такой механизм должен подтверждать, что подписывающее лицо не случайно подписало электронный документ. Во-вторых, он должен подтверждать, что только подписывающее лицо, и только оно, подписало электронный документ. В-третьих, он должен зависеть от содержания подписываемого документа и времени его подписания. В-четвертых, подписывающее лицо не должно иметь возможности в последствии отказаться от факта подписи документа. Таким механизмом стала электронно-цифровая подпись.

**Электронная цифровая подпись (ЭЦП)** — реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе (Федеральный Закон «Об электронно-цифровой подписи»).

**Криптографическое преобразование** — это преобразование информации, основанное на некотором алгоритме, зависящем от изменяемого параметра (обычно называемого секретным ключом), и обладающее свойством невозможности восстановления исходной информации по преобразованной, без знания

действующего ключа, с трудоемкостью меньше заранее заданной.Другими словами, ЭЦП - это часть электронного документа, а фактически - обычный файл, полученный в результате криптографического преобразования документа. Электронная подпись создаётся при помощи так называемого закрытого ключа. Проверяется же отправленный документ открытым ключом - он общедоступен и вместе с документом приходит получателю. То есть, ЭЦП использует т.н. асимметричное криптографическое преобразование, в котором используется пара ключей - открытый (публичный) ключ и секретный (личный, индивидуальный) ключ, известный только одной взаимодействующей стороне. При использовании ЭЦП гарантируется следующее. Во-первых, то, что документ не изменился в процессе пересылки. Если после подписания цифровой подписью документ был искажен, это выяснится при проверке открытым ключом. Во-вторых - гарантируется однозначная идентификация отправителя. В случае с обычной подписью это можно сделать, например, сравнив с подписью в паспорте. В случае с ЭЦП - специальные удостоверяющие центры, выдающие сертификаты ключей.

**Закрытый ключ электронной цифровой подписи** — уникальная последовательность символов, известная владельцу сертификата ключа подписи и предназначенная для создания в электронных документах электронной цифровой подписи с использованием средств электронной подписи;

**Открытый ключ электронной цифровой подписи** — уникальная последовательность символов, соответствующая закрытому ключу электронной цифровой подписи, доступная любому пользователю информационной системы и предназначенная для подтверждения с использованием средств электронной цифровой подписи подлинности электронной цифровой подписи в электронном документе;

**Сертификат ключа подписи** — документ на бумажном носителе или электронный документ с электронной цифровой подписью уполномоченного лица удостоверяющего центра, которые включают в себя открытый ключ электронной цифровой подписи и которые выдаются удостоверяющим центром участнику информационной системы для подтверждения подлинности электронной цифровой подписи и идентификации владельца сертификата ключа подписи;

При использовании асимметричного криптографического преобразования возникает задача обеспечения совместного использования зашифрованной (подписанной) информации, связанная с управлением ключами (генерация, распределение и т.д.). Для решения этой трудоемкой задачи создаются специальные

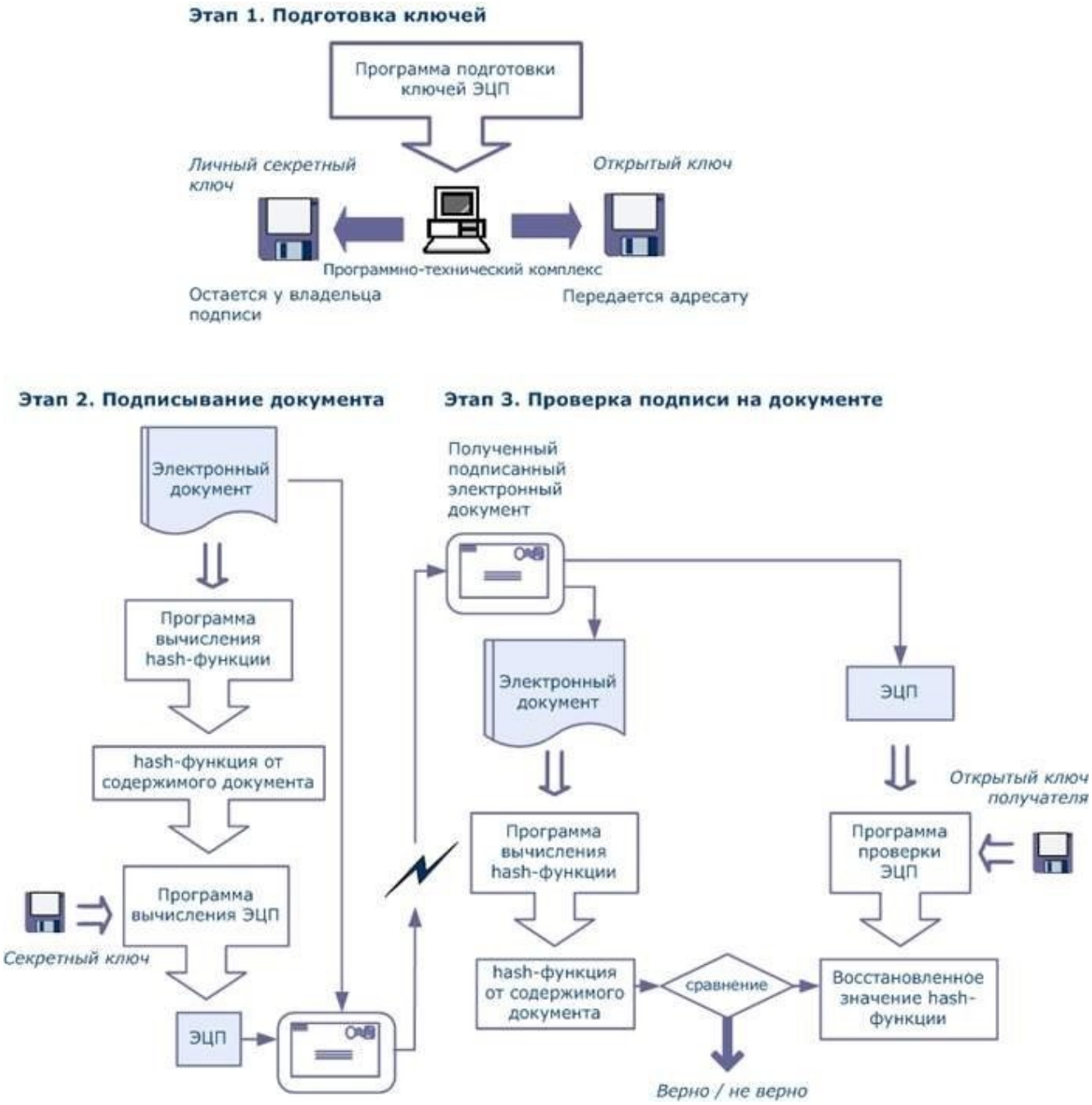


Рисунок 4.1 — Схема механизма ЭЦП, использующего асимметричное криптографическое преобразование

удостоверяющие центры, услуги которых призваны облегчить использование ЭЦП.

**Удостоверяющий центр** — это юридическое лицо, согласно Закону «Об электронно-цифровой подписи» выполняющее следующие функции:

1. изготовление сертификатов ключей подписей;

2. создание (генерация) ключей электронных цифровых подписей по обращению клиентов с гарантией сохранения в тайне закрытого ключа электронной цифровой подписи;

3. приостановка и возобновление действия сертификатов ключей подписей, а также их аннулирование;

4. ведение реестра сертификатов ключей подписей, обеспечение его актуальности и возможности свободного доступа к нему клиентов;

5. проверка уникальности открытых ключей электронных цифровых подписей в реестре сертификатов ключей подписей и архиве удостоверяющего центра;

6. выдача сертификатов ключей подписей в форме документов на бумажных носителях и (или) в форме электронных документов с информацией об их действии;

7. осуществление по обращениям пользователей сертификатов ключей подписей подтверждение подлинности электронной цифровой подписи в электронном документе в отношении выданных им сертификатов ключей подписей;

8. предоставление клиентам иных связанных с использованием электронных цифровых подписей услуги.

Таким образом, задачу подтверждения ЭЦП, а также другие связанные задачи, берет на себя удостоверяющий центр.

**Подтверждение подлинности электронной цифровой подписи в электронном документе** — положительный результат проверки соответствующим сертифицированным средством электронной цифровой подписи с использованием сертификата ключа подписи принадлежности электронной цифровой подписи в электронном документе владельцу сертификата ключа подписи и отсутствия искажений в подписанном данной электронной цифровой подписью электронном документе.

## 1. Структура сертификата

Формат сертификата открытого ключа определен в рекомендациях Международного Союза по телекоммуникациям ITU (X.509) и документе RFC 3280


Рисунок 4.2 — Реализация механизма ЭЦП через Удостоверяющий центр

Certificate & CRL Profile организации инженерной поддержки Интернета Internet Engineering Task Force (IETF). IETF является открытым интернациональным сообществом исследователей, разработчиков сетевых протоколов, операторов и производителей, занимающихся проблемами развития сетей Интернет и обеспечением непрерывного функционирования существующей инфраструктуры. В настоящее время основным принятым форматом является формат версии 3, позволяющий задавать дополнения, с помощью которых реализуется определенная политика безопасности в системе. Несмотря на то, что документ RFC 3820 адресован Интернет-сообществу, формат сертификата открытого ключа предоставляет гибкий механизм передачи разнообразной информации и применяется в корпоративных PKI.

Сертификат открытого ключа подписи или шифрования представляет собой структурированную двоичную запись в формате абстрактной синтаксической нотации ASN.1. Сертификат содержит элементы данных, сопровождаемые цифровой подписью издателя сертификата (см. рис. 6.1). В сертификате имеется десять основных полей: шесть обязательных и четыре опциональных. Большая часть информации, указываемой в сертификате, не является обязательной, а

содержание обязательных полей сертификата может варьироваться. К обязательным полям относятся:

– серийный номер сертификата Certificate Serial Number;
– идентификатор алгоритма подписи Signature Algorithm Identifier;
– имя издателя Issuer Name;
– период действия Validity (Not Before/After);
– открытый ключ субъекта Subject Public Key Information;
– имя субъекта сертификата Subject Name.

Под субъектом сертификата понимается сторона, которая контролирует секретный ключ, соответствующий данному открытому ключу. Наличие необязательных полей характерно для сертификатов версий 2 и 3, к необязательным полям сертификата относятся номер версии, два уникальных идентификатора и дополнения. Структура сертификата представлена на рис. ниже


Рисунок 4.3 — Структура сертификата

Издатель сертификатов присваивает каждому выпускаемому сертификату серийный номер Certificate Serial Number, который должен быть уникален. Комбинация имени издателя и серийного номера однозначно идентифицирует каждый сертификат.

**1.1.** Поля **`Signature Algorithm Identifier.`** указывается идентификатор алгоритма ЭЦП, который использовался издателем сертификата для подписи сертификата, например ГОСТ Р 34.10-94 (см. рис. 6.2).

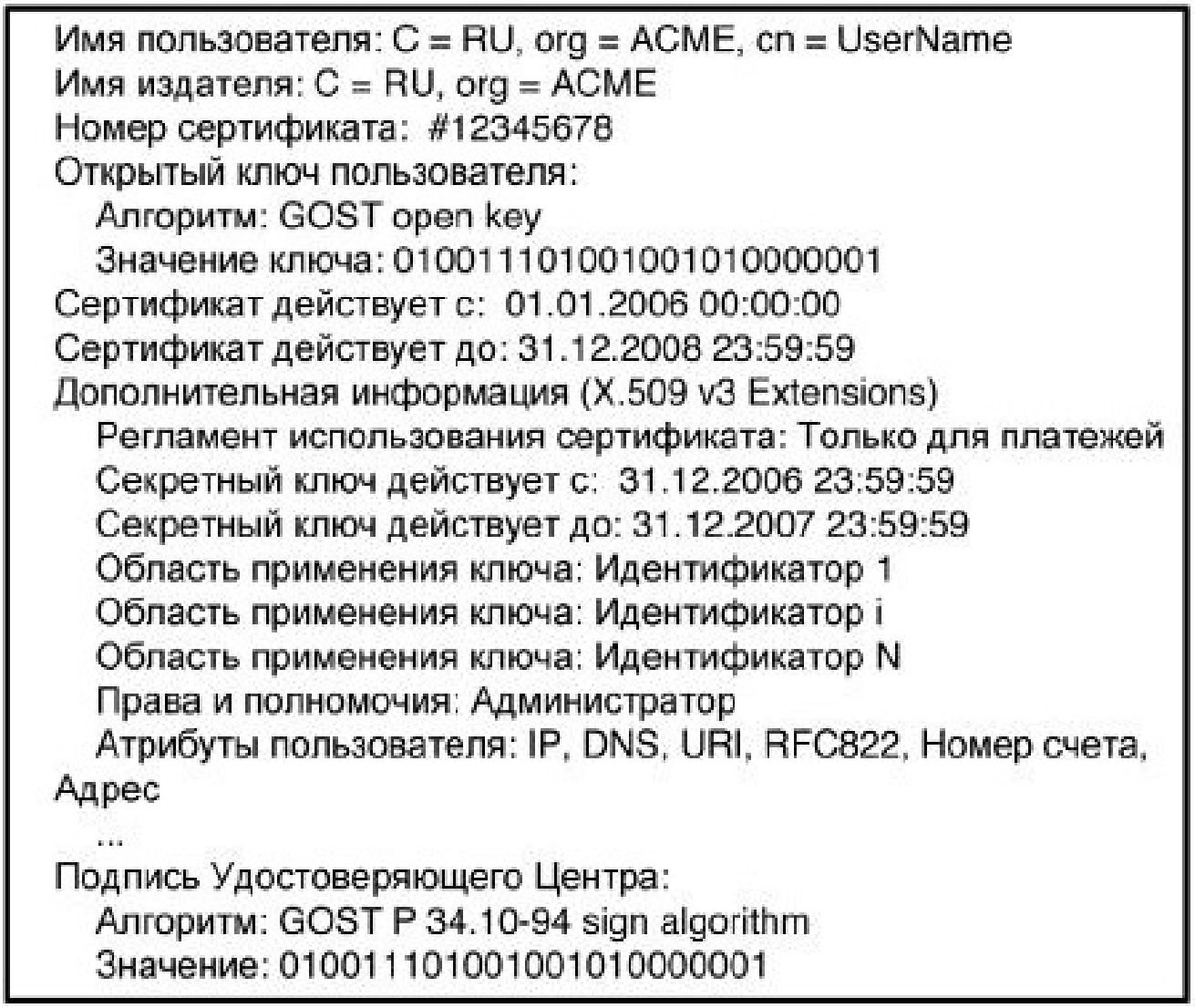


Рисунок 4.4 — Пример сертификата формата X.509

**1.2.** Поля **`Issuer Name.`** содержит отличительное имя (формата X.500) третьей доверенной стороны, то есть издателя, который выпустил этот сертификат. В поле ”Validity” (Not Before/After) указываются даты начала и окончания периода действия сертификата.

**1.3.** Поля **`Subject Name.`** содержит отличительное имя субъекта, то есть владельца секретного ключа, соответствующего открытому ключу данного сертификата. Субъектом сертификата может выступать УЦ, РЦ или конечный субъект.

**1.4.** Поля **Subject Public Key Information.** содержит информацию об открытом ключе субъекта: сам открытый ключ, необязательные параметры и идентификатор алгоритма генерации ключа. Это поле всегда должно содержать значение. Открытый ключ и необязательные параметры алгоритма используются для верификации цифровой подписи (если субъектом сертификата является УЦ) или управления ключами.

**1.5.** Поля **Issuer Unique Identifier** и **Subject Unique Identifier.** Данные поля являются необязательными. Они информируют об уникальных идентификаторах субъекта, издателя сертификата и предназначены для управления многократным использованием имен субъектов и издателей. Вследствие неэффективности подобного механизма управления Интернет-стандарты профилей сертификата и списка аннулированных сертификатов не рекомендуют использовать в сертификатах эти поля.

**1.6. Дополнения сертификата .** Важная информация находится также в дополнениях сертификата. Они позволяют включать в сертификат информацию, которая отсутствует в основном содержании, определять валидность сертификата и наличие у владельца сертификата прав доступа к той или иной системе. Кроме того, в дополнениях содержится технологическая информация, позволяющая легко проверить подлинность сертификата. Каждая организация может использовать свои частные дополнения, удовлетворяющие конкретным требованиям ведения бизнеса. Однако большинство требований включено в стандартные дополнения, поддержку которых обеспечивают коммерческие программные продукты.

Опциональное поле ”Extensions” (дополнения) появляется в сертификатах третьей версии. Каждое дополнение состоит из идентификатора типа дополнения Extension identifier, признака критичности Criticality flag и собственно значения дополнения Extension value. Идентификатор типа дополнения задает формат и семантику значения дополнения. Признак критичности сообщает приложению, использующему данный сертификат, существенна ли информация о назначении сертификата и может ли приложение игнорировать данный тип дополнения. Если дополнение задано как критичное, а приложение не распознает данный тип дополнения, то сертификат не должен использоваться приложением. Приложение может игнорировать нераспознанное некритичное дополнение и использовать сертификат.

Дополнения сертификатов X.509 определены рекомендациями X.509 версии 3 Международного Союза по телекоммуникациям и документом RFC 3280. Все эти дополнения можно разделить на две категории: ограничивающие и информационные. Первые ограничивают область применения ключа, определённого сертификатом, или самого сертификата. Вторые содержат дополнительную информацию, которая может быть использована в прикладном программном обеспечении пользователем сертификата.

К ограничивающим дополнениям относятся:

1. основные ограничения (Basic Constraints);
2. назначение ключа (Key Usage);
3. расширенное назначение ключа (Extended Key Usage);
4. политики применения сертификата (Certificates Policies, Policy Mappings, Policy Constraints);
5. ограничения на имена (Name Constraints).

К информационным дополнениям относятся:

1. идентификаторы ключей (Subject Key Identifier, Authority Key Identifier);
2. альтернативные имена (Subject Alternative Name, Issuer Alternative Name);
3. пункт распространения списка аннулированных сертификатов (CRL Distribution Point, Issuing Distribution Point);
4. способ доступа к информации УЦ (Authority Access Info).

## 2. Преимущества и недостатки ЭЦП

Недостаток метода: хотя сообщение надежно шифруется, но «засвечиваются» получатель и отправитель самим фактом пересылки шифрованного сообщения.

Общая идея криптографической системы с открытым ключом заключается в использовании при зашифровке сообщения такой функции от открытого ключа и сообщения (шифр -функции), которую алгоритмически очень трудно обратить, то есть вычислить по значению функции её аргумент, даже зная значение ключа.

**2.1. Особенности системы .** Преимущество асимметричных шифров перед симметричными шифрами состоит в отсутствии необходимости передачи секретного ключа. Сторона, желающая принимать зашифрованные тексты, в соответствии с используемым алгоритмом вырабатывает пару «открытый ключ – закрытый ключ». Значения ключей связаны между собой, однако вычисление

одного значения из другого должно быть невозможным с практической точки зрения. Открытый ключ публикуется в открытых справочниках и используется для шифрования информации контрагентом. Закрытый ключ держится в секрете и используется для расшифровывания сообщения, переданного владельцу пары ключей. Начало асимметричным шифрам было положено в 1976 году в работе Уитфилда Диффи и Мартина Хеллмана «Новые направления в современной криптографии». Они предложили систему обмена общим секретным ключом на основе проблемы дискретного логарифма. Вообще, в основу известных асимметричных криптосистем кладётся одна из сложных математических проблем, которая позволяет строить односторонние функции и функции-ловушки. Например, криптосистема Ривеста-Шамира-Адельмана использует проблему факторизации больших чисел, а криптосистемы Меркля-Хеллмана и Хора-Ривеста опираются на так называемую задачу об укладке рюкзака.

**Недостатки** — асимметричные криптосистемы требуют существенно больших вычислительных ресурсов. Кроме того, необходимо обеспечить аутентичность (подлинность) самих публичных ключей, для чего обычно используют сертификаты.

**Гибридная (или комбинированная) криптосистема** — это система шифрования, обладающая всеми достоинствами криптосистемы с открытым ключом, но лишенная её основного недостатка – низкой скорости шифрования.

*Принцип:* Криптографические системы используют преимущества двух основных криптосистем: симметричной и асимметричной криптографии. На этом принципе построены такие программы, как PGP и GnuPG.

**2.2. Основной недостаток.** асимметричной криптографии состоит в низкой скорости из-за сложных вычислений, требуемых её алгоритмами, в то время как симметричная криптография традиционно показывает блестящую скорость работы. Однако симметричные криптосистемы имеет один существенный недостаток – её использование предполагает наличие защищённого канала для передачи ключей. Для преодоления этого недостатка прибегают к асимметричным криптосистемам, которые используют пару ключей: открытый и закрытый.

*Шифрование:* Большинство шифровальных систем работают следующим образом. Для симметричного алгоритма (3DES, IDEA, AES или любого другого) генерируется случайный ключ. Такой ключ, как правило, имеет размер от 128 до 512 бит (в зависимости от алгоритма). Затем используется симметричный алгоритм для шифрования сообщения. В случае блочного шифрования необходимо

использовать режим шифрования (например, CBC), что позволит шифровать сообщение с длиной, превышающей длину блока. Что касается самого случайного ключа, он должен быть зашифрован с помощью открытого ключа получателя сообщения, и именно на этом этапе применяется криптосистема с открытым ключом (RSA или Алгоритм Диффи – Хеллмана). Поскольку случайный ключ короткий, его шифрование занимает немного времени. Шифрование набора сообщений с помощью асимметричного алгоритма – это задача вычислительно более сложная, поэтому здесь предпочтительнее использовать симметричное шифрование. Затем достаточно отправить сообщение, зашифрованное симметричным алгоритмом, а также соответствующий ключ в зашифрованном виде. Получатель сначала расшифровывает ключ с помощью своего секретного ключа, а затем с помощью полученного ключа получает и всё сообщение.

### 2.3. Цифровая подпись обеспечивает.

1. Удостоверение источника документа. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.

2. Защиту от изменений документа. При любом случайном или преднамеренном изменении документа (или подписи) изменится шифр, следовательно, подпись станет недействительной.

3. Невозможность отказа от авторства. Так как создать корректную подпись можно лишь, зная закрытый ключ, а он известен только владельцу, то владелец не может отказаться от своей подписи под документом.

### 2.4. Возможные угрозы цифровой подписи.

1. Злоумышленник может попытаться подделать подпись для выбранного им документа.

2. Злоумышленник может попытаться подобрать документ к данной подписи, чтобы подпись к нему подходила.

3. Злоумышленник, укравший закрытый ключ, может подписать любой документ от имени владельца ключа.

4. Злоумышленник может обманом заставить владельца подписать какой-либо документ, например используя протокол слепой подписи.

5. Злоумышленник может подменить открытый ключ владельца на свой собственный, выдавая себя за него.

**Важной проблемой** всей криптографии с открытым ключом, в том числе и систем ЭЦП, является управление открытыми ключами. Необходимо обеспечить доступ любого пользователя к подлинному открытому ключу любого другого пользователя, защитить эти ключи от подмены злоумышленником, а также организовать отзыв ключа в случае его компрометации.

Задача защиты ключей от подмены решается с помощью сертификатов. Сертификат позволяет удостоверить заключённые в нём данные о владельце и его открытый ключ подписью какого-либо доверенного лица. В централизованных системах сертификатов используются центры сертификации, поддерживаемые доверенными организациями. В децентрализованных системах путём перекрёстного подписывания сертификатов знакомых и доверенных людей каждым пользователем строится сеть доверия.

Управлением ключами занимаются центры распространения сертификатов. Обратившись к такому центру, пользователь может получить сертификат какого-либо пользователя, а также проверить, не отозван ли ещё тот или иной открытый ключ.

## 3. Применение ЭЦП

Электронная цифровая подпись может быть использована в самых разнообразных сферах деятельности. ЭЦП, как аналог собственноручной подписи, имеет целый ряд преимуществ, обусловленных её электронной природой.

1. Юридически значимый электронный документооборот между субъектами хозяйственной деятельности. Электронный документооборот с использованием ЭЦП позволяет вести деловую переписку, а также заключать договора и обмениваться другими документами по электронной почте. При этом такие электронные документы будут иметь такую же юридическую силу, как и документы на бумаге, а невозможность подделки ЭЦП будет дополнительной гарантией вашего бизнеса.

2. Эффективная обработка и хранение электронных документов. Электронный архив. Переход на электронный документооборот с использованием ЭЦП позволяет значительно повысить эффективность работы с документами в организации/предприятии. Электронными документами проще обмениваться, особенно в случае географически удаленной сети филиалов или представительств. Они занимают физически намного меньше места, что позволяет всегда иметь под рукой

все необходимые документы. Использование функций шифрования позволяет гарантированно защитить документы от несанкционированного доступа не только во время передачи, но и при хранении в электронном архиве.

3. Все это можно реализовать посредством специального программного обеспечения, которое поможет минимизировать проблемы при переходе на электронный документооборот и сделает работу с электронными документами интуитивно понятной и простой.

4. Подача отчетности в электронном виде в государственные учреждения и организации. Применение ЭЦП позволяет значительно упростить и ускорить обмен документами между субъектами хозяйственной деятельности и контролирующими государственными структурами. Электронная форма подача отчетности подразумевает возможность обмена всеми необходимыми документами через Интернет, например с помощью электронной почты. При этом документы подписываются с помощью ЭЦП (что дает им юридическую силу) и шифруются (что позволяет защитить их от несанкционированного доступа). Подача отчетности в электронном виде позволяет сэкономить время, расходы на доставку и снизить риски связанные с человеческим фактором. К тому же проверка и обработка электронных документов может быть полностью автоматизирована, что минимизирует ошибки при проверке документа.

5. Аутентификация и разграничение доступа. Технология ЭЦП имеет применение и в областях, не связанных с электронным документооборотом. Электронные ключи могут использоваться для получения доступа к управлению системами/механизмами требующими надежной аутентификации и протоколирования выполняемых операций.

### 3.1. Текущие требования к стойкости ЭЦП. пока пусто

## 4. Проверка подлинности ЭЦП

Одним из способов проверки подлинности ЭЦП является предоставление сертификата, подтверждающего, что подпись и сертификат на момент подачи документов актуальны и не утратили свою силу.

Сертификат ключа проверки электронной подписи может представлять собой как электронный документ, так и удостоверение в бумажном виде, которые подтверждают законное право владельца на электронную подпись и на ключ её проверки.

Квалифицированные сертификаты ключа проверки подлинности электронной подписи могут быть выданы лишь таким удостоверяющим центром, который обязательно имеет аккредитацию или уполномочен на это федеральным органом.

Дополнить информацию для Украины

## Задания

Уровень 1.

1. Выбрать и отобразить реестр сертификатов на Вашем устройстве.
2. Опишите состав сертификата.
3. Выберите шифрованный блок данных.
4. Опишите полный путь данного сертификата к main-центру раздачи разрешений.

Уровень 2.

1. Создайте самоподписной сертификат.
2. Внесите его в свой реестр.
3. Приложите данные в отчёт.

Уровень 3.

1. Напишите алгоритм для создания подписанного сертификата.
2. Создайте сертификат подписанный ЭЦП из прошлого задания.
3. Приложите пример работы программы и подписанный сертификат.

## Пример выполнения работы

Пример будет включать создание ЭПЦ как самостоятельно, так и используя ПО.

## Варианты

## Вопросы для контроля

1. Что такое электронно-цифровая подпись?
2. Каков принцип работы ЭЦП?

3. Почему ЭЦП используется в большинстве систем проверки документации?
4. Опишите этапы шифрования-дешифровки.
5. Как подпись обеспечивает целостность данных?
6. Случаи небезопасного использования ЭПЦ?
7. Где на компьютере могут храниться цифровые подписи?
8. Как проверить надёжность ЭПЦ?

# ЛАБОРАТОРНА РОБОТА 5

# СИСТЕМЫ АВТОРИЗАЦИИ ПОЛЬЗОВАТЕЛЯ

**Мета роботи:** Освоить основные принципы систем автоматической авторизации пользователя.

## Теоретические ведомости

*«Нужно заменить систему KERBEROS на другую с подобным принципом работы»*

## Задания

## Пример выполнения работы

## Варианты

## Вопросы для контроля

# ЛАБОРАТОРНА РОБОТА 6

# ПАКЕТЫ АНТИВИРУСНОЙ ЗАЩИТЫ

**Мета роботи:** Изучение способов защиты системы с помощью различного ПО. Научиться выбирать ПО для защиты системы.

## Теоретические ведомости

**Вредоносная программа** — любое программное обеспечение, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ, и/или владельцу сети ЭВМ, путём копирования, искажения, удаления или подмены информации.

### 1. История

Сказать, где и когда появился первый вирус, невозможно, поскольку таких данных в природе не существует. Если на «компьютере» Чарльза Бэббиджа, «отца» первой вычислительной машины, вирусов ещё не было, к середине семидесятых годов прошлого века они стали весьма распространенным и неприятным для большинства явлением. Тем не менее, предпосылки к их созданию появились практически сразу же с созданием первых ЭВМ.

Еще в 1940 году математик Джон фон Нейман написал книгу, в которой были описаны самовоспроизводящиеся математические автоматы, то есть принципы, которые легли в основу всех вирусов. В 1959 году американский научный журнал «Scientific American» опубликовал статью Л. Пенроуза, рассказывавшую о самостоятельно распространяющихся биологических структурах. Автор рассмотрел способности подобных структур к мутациям, активации и размножению. Другой ученый, Ф. Шталь, полученные из этой статьи знания реализовал на практике. Работая оператором в научно-исследовательской лаборатории, он имел доступ к мощнейшей для того времени ЭВМ – IBM 650. Эксперимент очень удивил Шталя, превзойдя все его ожидания. Получившийся в результате «мутации» математических алгоритмов электронный «зверек» удалил все следы своих «родителей», присутствовавших в системе, после чего самоуничтожился.

Естественно, все вышеперечисленные труды и опыты были направлены не для того, чтобы нынешние вирусописатели ежедневно выбрасывали в Интернет мегабайты новой «заразы». Изначально эти исследования, относившиеся к области создания искусственного интеллекта, представляли собой академический интерес. Однако любое открытие, сделанное в мирных целях, может быть без особых трудностей превращено в мощное оружие разрушения.

В 1961 году среди компьютерщиков была очень популярна игра «Darwin». Её сюжет и смысл были просты: игрок руководил «расой», которая должна была уничтожить своих конкурентов. Выигрывал тот, кто захватит всю отданную под игровой процесс оперативную память. Особых действий в игре не требовалось: необходимо было лишь размножить принадлежащих к своей расе на свободные ячейки ОЗУ или же захватить ячейки противника. Подобный алгоритм очень похож на логику работы деструктивных программ.

Широкое распространение компьютерных сетей стало катализатором появления на свет первых деструктивных программ — **компьютерных вирусов**.

**1.1. 70-е годы: начало.** Появление первого в мире компьютерного вируса было зафиксировано в начале 70-х годов прошлого столетия, когда на просторах военной компьютерной сети APRAnet – прародительницы современного Интернета – была найдена программа Creeper. Вирус был написан для распространенной в те времена операционной системы Tenex, в которую он проникал, распространяясь через модемную связь. На экран зараженных компьютеров периодически выводилась надпись: «I’M THE CREEPER: CATCH ME IF YOU CAN». Деструктивных действий Creeper не совершал, ограничиваясь лишь этим сообщением, раздражавшим пользователей. Чуть позже для него было написано «противоядие» – программа Reaper, находившая файл с вирусом и удалявшая его. Распространялась она, кстати, аналогичным с Creeper способом. Можно сказать, что первый в мире антивирус был создан «по аналогии с вредоносной программой».

В 1974 году «частым гостем» на различных серверах была программа с милым для животноводов названием Rabbit. «Кролик» ничего, кроме распространения и размножения самого себя, не делал. Программа самовоспроизводилась с огромной скоростью, постепенно занимая все системные ресурсы. Иногда Rabbit даже вызывал сбой в работе серверов.

Другой пример – логическая игра Pervading Animal для операционной системы Exec 8, смысл которой заключался в отгадывании пользователем загаданного программой животного. Если ему это не удавалось, игра предлагала модернизировать её, после чего появлялась возможность задавать дополнительные наводящие вопросы.

Модифицированная версия программы странным образом начинала копироваться в другие директории, в результате чего через некоторое время во всех папках жёсткого диска содержалась копия Pervading Animal. Так как в то время каждый килобайт пространства был «на вес золота», подобное поведение игры мало кого обрадовало. До сих пор не ясно, была ли это ошибка программистов либо же задумка вирусописателей. Впрочем, проблема была быстро решена – новая версия операционной системы Exec 8 базировалась на другом типе файловой системы, на которой программа засорять файловое пространство больше не могла.

**1.2. 80-е: первые эпидемии.** К восьмидесятым годам прошлого столетия компьютер перестал быть роскошью, доступной лишь избранным. Владельцев ПК становится все больше, кроме того, обмен информацией между пользователями с помощью электронных досок объявлений (BBS – Buletin Board System) достиг международного масштаба.

В 1981 году произошла первая по-настоящему массовая вирусная эпидемия. Пострадали широко распространенные в то время компьютеры Apple II. Вирус Elk Clone записывался в загрузочные секторы дискет в момент обращения к ним пользователя. Elk Clone искажал изображение на мониторе, выводил на экран различные текстовые сообщения, заставлял мигать текст. Неискушенные пользователи впадали от действий вируса в оцепенение, в то время как он сам продолжал «перемещаться» с одного компьютера на другой.

В 1983 году американский программист Лен Эйделман впервые употребил термин «вирус», которым он обозначил саморазмножающиеся программы.

Спустя год Фред Коэн, один из самых авторитетных вирусологов, дал четкое определение понятию «компьютерный вирус»: «программа, способная «заражать» другие программы при помощи их модификации с целью внедрения своих копий».

В 1986 году 19-летний пакистанец Басит Фарук Алви написал вирус Brain. Также как и Elk Clone, Brain поражал загрузочные сектора дискет. Программа не была ориентирована на какие-либо разрушительные функции, она лишь меняла метку всех дискет на «(C)Brain». Как утверждает автор, он преследовал только

одну цель – выяснить уровень компьютерного пиратства у себя в стране. Но уже через несколько недель после активации вируса зараженными оказались тысячи компьютеров по всему миру, что вызвало настоящий переполох среди пользователей и бурю обсуждений в средствах массовой информации. В Brain был впервые использован прием, когда при чтении зараженного сектора диска вирус подставлял вместо этого раздела незараженный.

В 1988 году была создана первая вредоносная программа, которая не просто заражала компьютер, но и наносила ему реальный вред. Этот вирус был создан в Лехайском университете, в котором, кстати, работал ранее упоминавшийся Фред Коэн. Вирус Lehigh уничтожал информацию на дисках, поражая системные файлы COMMAND.COM. Наличие квалифицированных специалистов в университете оказалось спасением – за стены учебного заведения он так и не пробрался. Впрочем, немалую роль в устранении угрозы эпидемии сыграл и алгоритм самого Lehigh – во время форматирования винчестеров он самоуничтожался вместе с остальной информацией.

В это же время начало активно развиваться программное обеспечение, защищавшее компьютеры от вирусов. Антивирусные программы того времени представляли собой простенькие сканеры, которые посредством контекстного поиска пытались обнаружить вирусный код в программах. Другим распространенным «лекарством» от вредоносных программ того времени были «иммунизаторы». Этот тип ПО модифицировал все программы таким образом, чтобы вирусы считали их уже зараженными и не выполняли по отношению к ним никаких действий. После того как количество вирусов возросло в тысячи раз, использование иммунизаторов было уже бесполезным.

Антивирусные фирмы чаще всего состояли из двух-трех человек и свои продукты продавали за символическую сумму либо раздавали бесплатно. Но распространенность защитных программ была очень низка, да и непрерывное появление новых вирусов делало их бессильными. Интернет в то время ещё не успел «вырваться» из «объятий» учёных и военных, а обновляться без наличия глобальной сети было практически невозможно.

В середине 80-х годов появился термин «Virus Hoax» – «вирусная мистификация». В конце восьмидесятых пользователи панически боялись вирусов: мифы о программах, выводящих из строя аппаратную часть ПК, будоражили ум каждого владельца компьютера. Virus Hoax были ничем иным, как ложными слухами о новых компьютерных эпидемиях. Вспоминается история, когда один шутник

разослал на разные BBS сообщения о появлении нового вируса, который распространялся через модемы, работавшие со скоростью передачи информации 2400 бит в секунду. Чтобы не заразиться вирусом, автор рекомендовал перейти на модемы со скоростью 1200 бит/с. И что вы думаете? Масса пользователей выбросила более быстрые модемы ради своей «безопасности».

В 1988 произошла первая эпидемия, вызванная сетевым компьютерным вирусом. Впоследствии такие вирусы стали именоваться «червями». Созданная неким Робертом Моррисом программа поражала компьютеры, работавшие под ОС UNIX. В планы создателя не входило нанесение вреда системе, червь должен был лишь проникнуть в сеть ARPAnet и оставаться там незамеченным. Вирус обладал возможностью вскрытия паролей в ОС, и в списке выполнявшихся процессов детище Морриса отображалось в виде обычного пользовательского процесса. Червь стремительно саморазмножался и пожирал все свободные ресурсы компьютера, вследствие чего из строя выходили целые серверы. Некоторые из них смогли вернуться к работе лишь через пять суток, поскольку вакцины против червя не существовало. За время своего «хождения по миру» вирус поразил около 6000 компьютерных систем, затронув даже компьютеры исследовательского центра NASA. Роберт Моррис отделался 400 часами общественных работ, но вошел в историю как автор первого разрушительного сетевого червя.

**1.3. 90-е: полиморфные вирусы.** В начале 90-х годов прошлого столетия английская компания Sophos, в которой работали Ян Храске, Эд Уайлдинг и Питер Лэймер, приступила к выпуску журнала «Virus Bulletin». Virus Bulletin рассказывал о компьютерных вирусах, а также обо всех аспектах защиты от них. Авторами журнала являлись программисты, руководители антивирусных компаний, разработчики ПО. Журнал был некоммерческим: за всю его историю в нем не было напечатано ни одного рекламного объявления. Из-за этого Virus Bulletin не получил широкой распространенности. Его читателями были в основном профессионалы в сфере IT (информационных технологий), а также сотрудники компьютерных фирм.

В 1990 году появился новый тип вредоносных программ – полиморфные вирусы. «Полиморфизмом» была названа технология, при которой вирус нельзя было обнаружить сканером, искавшим вирусы с помощью фрагментов уже известного вредоносного кода. Полиморфизм позволяет программам генерировать код во время исполнения, в результате чего копия вируса на каждом новом заражённом компьютере будет отличаться от предыдущей. Первым подобным вирусом

стал Chameleon, написанный Марком Вашбёрном. После появления полиморфных программ неотъемлемой частью антивируса стал эмулятор для дешифрации кодов, впервые использованный Евгением Касперским.

В этом же году в Болгарии, которая тогда была центром мирового вирусописания, появилась специализированная BBS, с которой каждый желающий мог скачать вредоносные программы. Конференции, посвященные программированию вирусов, появились и в UseNet.

В это же время была опубликована книга «Маленькая Черная Книжка о Компьютерных Вирусах» Марка Людвига. Она стала «Библией» всех создателей вирусов. Была сформирована так называемая «VX-сцена» – сообщество программистов, специализирующихся на создании компьютерных вирусов.

**1.4. Конструкторы вредоносных программ.** В 1992 году хакер, известный под ником Dark Avenger, выпустил в свет утилиту MtE (Mutation Engine). С её помощью любой, даже самый примитивный вирус можно было сделать полиморфным. Этим же человеком был впервые создан вирус Peach, наделенный способностью обходить антивирусное ПО. Peach удалял базу изменений программы Central Point AntiVirus. Эта программа, не находив свою базу данных, считала, что запущена впервые, и создавала её вновь. Таким образом, вирус обходил защиту и продолжал заражать систему.

Группа программистов, известная в сети, как Nowhere Man, выпустила конструктор вирусов VCL (Virus Creation Laboratory). Отныне любой школьник, даже не владеющий языками программирования, мог вооружиться конструктором и собрать вирус любого типа и разрушительной силы. С появлением VCL и так немалый «поток» новых компьютерных вредителей стал просто огромным.

**1.5. Первый арестованный вирусописатель.** На протяжении 1993-94 годов свет увидели новые конструкторы вирусов: PS-MPC и G2. Сгенерированные ими вредоносные программы стали самой распространенной опасностью в Интернете.

В это же время состоялся настоящий «бум» среди производителей антивирусов – их программы, наконец-то, стали обязательной составляющей к практически любой ОС. На рынок безопасности решила проникнуть даже Microsoft, выпустившая Microsoft AntiVirus (MSAV). Первоначально программа была популярна, но впоследствии крупнейший производитель программного обеспечения в мире прекратил разработку продукта.

Лидерство в этой области постепенно завоевала компания Symantec, частью которой стали крупнейшие производители антивирусного программного обеспечения: Central Point и Fifth Generation Systems.

Эпидемия нового полиморфного вируса, Pathogen, уже не была событием из ряда вон выходящим, к подобного рода событиям все уже начали привыкать. Однако это был первый вирус, автор которого был найден и осуждён. Безработный Кристофер Пайл за создание вредоносных программ был приговорен к 18 месяцам тюремного заключения.

**1.6. Атака на Microsoft.** В 1995 году все разосланные бета-тестерам диски с операционной системой Windows 95 были заражены загрузочным вирусом Form. К счастью, один из них обнаружил неладное, и на прилавки магазинов поступила нормальная, незараженная система.

В августе того же года появился первый макровирус, написанный на языке WordBasic, встроенном в текстовый редактор MS Word. Макровирусом Concept были заражены сотни тысяч компьютеров по всему земному шару, вследствие чего он еще долго лидировал в статистических исследованиях компьютерных журналов.

В 1996 году первую эпидемию пережили пользователи Windows 95 – их компьютеры были поражены загрузочным вирусом Boza. В июле того же года создатели макровирусов переключились с Word на редактор электронных таблиц MS Excel, выпустив для него вирус Laroux.

Не заставили себя ждать и резидентные вирусы, использующие сервисы «нулевого кольца» ОС. Win95.Punch загружался в систему как VxD-драйвер, перехватывал обращения к файлам и заражал их.

**1.7. Антивирусные склоки.** К 1997 году операционная система Linux, ранее считавшаяся оплотом «чистоты и стабильности», больше не являлась платформой, свободной от вирусов. Linux.Bliss, распространявшийся посредством конференций UseNet, заражал исполняемые файлы этой ОС.

В этом же году было отмечено появление двух новых типов червей, распространявшихся через IRC и FTP. Особо большим их количеством мог «похвастаться» IRC, во многом из-за своей популярности, а также многочисленных «дыр» mIRC – основного клиента подобных сетей.

Под конец XX века в погоне за лидерством стали нередки скандалы среди производителей антивирусов. Так, представители компании McAfee объявили о

том, что ее программисты обнаружили ошибку в антивирусе фирмы Dr.Solomon's. Суть заявления сводилась к тому, что Dr.Solomon's мог находить новые и технически совершенные вирусы только в специальном «усиленном» режиме, в который переключался лишь после нахождения обычных, примитивных червей. В результате антивирус показывал хорошие скоростные результаты при сканировании незараженных дисков, и отличные показатели обнаружения при работе с зараженными файлами. В ответ Dr.Solomon's подала иск в суд на McAfee, причиной которого стала её «некорректно построенная рекламная компания». В итоге вся «заварушка» завершилась покупкой McAfee контрольного пакета акций Dr.Solomon's.

Спустя некоторое время публичное заявление сделали тайваньские разработчики из фирмы Trend Micro, обвинившие McAfee и Symantec в якобы «нарушении их патента на сканирование данных». Миру были сразу представлены доказательства о «безгрешности» компаний, однако Trend Micro добилась своего, получив отменную бесплатную рекламу в средствах массовой информации.

**1.8. Наиболее разрушительные вирусы.** Продолжать подробную историю компьютерных вирусов вплоть до наших дней не имеет смысла, поскольку ежегодно появляются сотни и тысячи новых вредоносных программ. Я ограничусь лишь кратким рассказом о самых известных вирусах, появившихся после 1997 года:

**CIH** (1998) – ущерб, нанесенный вирусом, составил порядка 80 миллионов долларов. Вирус был написан программистом из Тайваня, и стал одним из самых разрушительных в истории. «Чих» заражал исполняемые файлы и активировался каждый год 26 апреля – в день годовщины аварии на Чернобыльской АЭС. CIH перезаписывал FlashBIOS, после чего материнские платы становились непригодными к использованию. Первый и последний вирус, который наносил вред аппаратной части ПК.

**Melissa** (1999) – 26 марта 1999 года этот макровирус, распространявшийся по электронной почте, заразил около 20

**ILOVEYOU** (2000) – скрипт, написанный на макроязыке Visual Basic. Так же как и Melissa, распространялся по электронной почте с темой письма «I love you». Вирус рассылал свои копии по всем данным адресной книги почтового клиента. Все логины и пароли, найденные червем на компьютере, отсылались автору

программы. Последний, кстати, и не пытался скрываться: он является жителем Филиппин, где наказаний за компьютерные преступления не предусмотрено.

**Code Red** (2001) – сетевой червь, использующий ошибку в сетевом сервисе Microsoft IIS. В заданный день зараженные компьютеры должны были начать DDOS-атаку по списку различных серверов, среди которых были системы правительства США. Огромные масштабы эпидемии и как итог – убытки в 2,5 миллиарда долларов.

**Blaster** (2003) – сетевой червь, выводивший на зараженных компьютерах сообщение о необходимости перезагрузки. Через пару дней после его выпуска в Интернет (11 августа) были заражены миллионы компьютеров по всему миру.

**Sobig.F** (2003) – сетевой червь, распространявшийся по электронной почте. Размножавшийся с огромной скоростью вирус скачивал на заражённый компьютер дополнительные файлы, «сжигая» трафик и ресурсы системы. Интересная особенность – 10 сентября вирус прекращал свою деятельность, больше не представляя угрозы для пользователя. Автор Sobig.F, за информацию о котором Microsoft предлагала 250 тыс. долларов, не найден до сих пор.

**Bagle** (2004) – сетевой червь, распространявшийся по классическому способу, используя файловые вложения в электронных письмах. На заражённом компьютере устанавливалась специальная «лазейка», через которую злоумышленник получал полный доступ к системе. Вирус имеет более ста модификаций.

**MyDoom** (2004) – в январе 2004 года этот вирус молниеносно распространился по всему Интернету, в результате чего средняя скорость загрузки сайтов в глобальной сети уменьшилась на 50

**Sasser** (2004) – вирус вызвал «перерыв» в работе французских спутниковых каналов передачи данных, отменил рейсы некоторых авиакомпаний, не говоря уже об обычных компьютерах, чья работа была полностью приостановлена. Распространялся Sasser благодаря ошибке в системе безопасности Windows 2000 и XP, запуская на зараженном компьютере сканер портов. Вирус был написан 17-летним немецким школьником. Интересен тот факт, что парень запустил вирус в сеть в День своего совершеннолетия.

## 2. Классификация

У компаний-разработчиков антивирусного программного обеспечения существуют собственные классификации и номенклатуры вредоносных программ.

Приведённая в этой статье классификация основана на номенклатуре «Лаборатории Касперского».

**2.1. По вредоносной нагрузке.**

1. Помехи в работе заражённого компьютера: начиная от открытия-закрытия поддона CD-ROM и заканчивая уничтожением данных и поломкой аппаратного обеспечения. Поломками известен, в частности, Win32.CIH.

2. Блокировка антивирусных сайтов, антивирусного ПО и административных функций ОС с целью усложнить лечение.

3. Саботирование промышленных процессов, управляемых компьютером (этим известен червь Stuxnet).

4. Инсталляция другого вредоносного ПО.

5. Загрузка из сети (downloader).

6. Распаковка другой вредоносной программы, уже содержащейся внутри файла (dropper).

7. Кража, мошенничество, вымогательство и шпионаж за пользователем. Для кражи может применяться сканирование жёсткого диска, регистрация нажатий клавиш (Keylogger) и перенаправление пользователя на поддельные сайты, в точности повторяющие исходные ресурсы.

8. Похищение данных, представляющих ценность или тайну.

9. Кража аккаунтов различных служб (электронной почты, мессенджеров, игровых серверов…). Аккаунты применяются для рассылки спама. Также через электронную почту зачастую можно заполучить пароли от других аккаунтов, а виртуальное имущество в MMOG — продать.

10. Кража аккаунтов платёжных систем.

11. Блокировка компьютера, шифрование файлов пользователя с целью шантажа и вымогательстваденежных средств (см. Ransomware). В большинстве случаев после оплаты компьютер или не разблокируется, или вскоре блокируется второй раз.

12. Использование телефонного модема для совершения дорогостоящих звонков, что влечёт за собой значительные суммы в телефонных счетах.

13. Платное ПО, имитирующее, например, антивирус, но ничего полезного не делающее (fraudware или scareware (англ.)русск.; см. тж лжеантивирус).

14. Прочая незаконная деятельность:

15. Получение несанкционированного (и/или дарового) доступа к ресурсам самого компьютера или третьим ресурсам, доступным через него, в том числе

прямое управление компьютером (так называемый backdoor).

16. Организация на компьютере открытых релеев и общедоступных прокси-серверов.

17. Заражённый компьютер (в составе ботнета) может быть использован для проведения DDoS-атак.

18. Сбор адресов электронной почты и распространение спама, в том числе в составе ботнета.

19. Накрутка электронных голосований, щелчков по рекламным баннерам.

20. Генерация монет платёжной системы Bitcoin.

21. Шуточное ПО, делающее какие-либо беспокоящие пользователя вещи.

22. Adware — программное обеспечение, показывающее рекламу.

23. Spyware — программное обеспечение, занимающееся массовым сбором малоценной информации — например, конфигурации компьютера, каталогов диска, активности пользователя…

24. «Отравленные» документы, дестабилизирующие ПО, открывающее их (например, архив размером меньше мегабайта может содержать гигабайты данных и надолго «завесить» архиватор).

25. Программы удалённого администрирования могут применяться как для того, чтобы дистанционно решать проблемы с компьютером, так и для неблаговидных целей.

26. Руткит нужен, чтобы скрывать другое вредоносное ПО от посторонних глаз. Это возможно благодаря тесной интеграции руткита с операционной системой.

27. Иногда вредоносное ПО для собственного «жизнеобеспечения» устанавливает дополнительные утилиты: IRC-клиенты, программные маршрутизаторы, открытые библиотеки перехвата клавиатуры… Такое ПО вредоносным не является, но из-за того, что за ним часто стоит истинно вредоносная программа, детектируется антивирусами. Бывает даже, что вредоносным является только скрипт из одной строчки, а остальные программы вполне легитимны.

28. Файлы, не являющиеся истинно вредоносными, но в большинстве случаев нежелательные

**2.2. По методу размножения.**

1. Эксплойт — теоретически безобидный набор данных (например, графический файл или сетевой пакет), некорректно воспринимаемый программой, работающей с такими данными. Здесь вред наносит не сам файл, а неадекватное

поведение ПО с ошибкой. Также эксплойтом называют программу для генерации подобных «отравленных» данных.

2. Логическая бомба в программе срабатывает при определённом условии, и неотделима от полезной программы-носителя.

3. Троянская программа. По своему действию является противоположностью вирусам и червям. Его предлагают загрузить под видом законного приложения, однако вместо заявленной функциональности он делает то, что нужно злоумышленникам. Трояны не самовоспроизводятся и не распространяются сами по себе.Нынешние трояны эволюционировали до таких сложных форм, как, например, бэкдор (троян, пытающийся взять на себя администрирование компьютера) и троян-загрузчик (устанавливает на компьютер жертвы вредоносный код).

4. Компьютерный вирус размножается в пределах компьютера и через сменные диски. Размножение через сеть возможно, если пользователь сам выложит заражённый файл в сеть. Вирусы, в свою очередь, делятся по типу заражаемых файлов (файловые, загрузочные, макро-, автозапускающиеся); по способу прикрепления к файлам (паразитирующие, «спутники» и перезаписывающие) и т. д.

5. Сетевой червь способен самостоятельно размножаться по сети. Делятся на IRC-, почтовые, размножающиеся с помощью эксплойтов и т. д.

6. Загрузчик — является небольшой частью кода, используемой для дальнейшей загрузки и установки полной версии вредоноса. После того как загрузчик попадает в систему путем сохранения вложения электронного письма или, например, при просмотре зараженной картинки, он соединяется с удаленным сервером и загружает весь вредонос.

## 3. Виды антивирусной защиты

**Современные антивирусы** — это комплексные программные пакеты, как правило, содержащие несколько взаимосвязанных и взаимодополняющих модулей, нацеленные на борьбу со всем спектром компьютерных угроз.

В современных антивирусах могут задействоваться следующие виды антивирусной защиты:

**Сравнение с вирусным образцом** — вирусной сигнатурой кода, шаблоном поведения вредоносной программы или цифровым отпечатком в «черном» списке известных угроз. Эта разновидность антивирусной защиты заключается в исследовании подозрительной программы на наличие признаков, характерных для вредоносного ПО. Например, реализуя данный вид защиты, антивирус ищет

**сигнатуры** – последовательности кода, уникальные для определённого вируса.

**Поведенческий мониторинг** — разновидность антивирусной защиты, основанная на проверке объектов во время осуществления чтения, записи и других операций. Для проведения мониторинга антивирусная программа располагается в оперативной памяти и действует как обработчик системных событий. При старте какой-либо операции, которая может привести к заражению, антивирусный монитор запускает проверку обрабатываемого объекта (документа, программы и т.д.).

**Обнаружение изменений** — вид антивирусной защиты, базирующийся на контроле целостности программных компонентов компьютера. При заражении вирусы модифицируют файлы, системный реестр или загрузочные сектора диска. Антивирусная программа определяет, был ли изменен объект с помощью подсчета кодов циклического контроля (CRC-сумм) и других методов.

**Эвристический анализ**. Данный вид антивирусной защиты основан на том, что выполняемые вирусами действия и их последовательность отличаются от поведения большинства программ. Поэтому анализ последовательностей команд и системных вызовов подозрительного программного обеспечения помогает выносить правильное решение о его вредоносности.

**Лечение** — разновидность антивирусной защиты, состоящая в удалении вредоносных объектов и восстановлении нормальных параметров компьютерной системы.

**Репутационный сервис** — новейший вид антивирусной защиты, получивший распространение в последние годы и базирующийся на проверке репутации программ, веб-ресурсов и почтовых систем. Такая проверка проводится с использованием «облачных» серверов репутации, поддерживаемых ведущими разработчиками антивирусного ПО, и основана на постоянно обновляемых списках «легитимных», вредоносных и подозрительных ресурсов. Преимуществом репутационных сервисов является очень высокая скорость реакции на появление новых угроз.

Существуют и устаревшие, теперь уже редко используемые виды антивирусной защиты, например, иммунизация, которая заключается в том, что в памяти компьютера размещается программа, сообщающая вирусам, избегающим повторного заражения, о том, что система уже инфицирована.

Реализуют антивирусную защиту следующие модули:

– Антивирусный сканер

– Антивирусный монитор, использующий многочисленные технологии защиты
– Поведенческий блокиратор
– Антивирусный ревизор или система контроля CRC
– Антивирусный фаг или доктор.

## Задания

Уровень 1.

1. По варианту выбрать систему.
2. Исследовать систему на стойкость, защиту.
3. Описать основные угрозы и способы взлома выбранной системы.
4. Анализ и выбор ПО для комплексной защиты.

Предоставить результат в виде отчета с подробным описанием слабых сторон системы, указать способы защиты и написать инструкцию для выбранного ПО.

## Пример выполнения работы

## Варианты

## Вопросы для контроля

# ЛАБОРАТОРНА РОБОТА 7

# ПАССИВНЫЙ АНАЛИЗ ДАННЫХ

**Мета роботи:** Применение методов и технологий испытания программного и аппаратного уровней комплексной защиты информации для проведения атаки на КИС с целью установления уязвимостей.

## Теоретические ведомости

Анализ трафика и сбор критичной информации программами пассивного анализа является одним из методов получения критичной информации о корпоративной информационной системе. Для реализации необходимо иметь специализированное программное обеспечение.

В сфере обеспечения ИБ необходимо обнаруживать и нейтрализовать вредоносный код прежде, чем он нанесет ущерб информационной системе. Проанализировав нестандартное «поведение» трафика, можно определить его как угрожающее и, блокировав действие соответствующих программ, сообщить пользователю об инциденте.

Проведение анализа трафика и сбор критичной информации с применением программ пассивного анализа (программ-снифферов и программ обнаружения вторжений) включает:

- определение слабых мест в защите сервисов: FTP, TFTP, SSH, Finger, HTTP, IMAP SMTP, NetBIOS/SMB, RPC;
- выявление слабых мест сетевых информационных служб (NIS);
- проверка на возможность IP-спуфинга;
- проверка маршрутизации из источника rlogin, rsh и telnet;
- проверка IP-переадресации (forwarding);
- проверка сетевых масок и временных меток (timestamp) ICMP;
- проверка инкапсуляции пакета MBONE;
- проверка инкапсуляции APPLETALK IP, IPX, X.25, FR;
- проверки резервированных разрядов и паритет-протоколов;
- проверка специализированных фильтров;
- проверка фильтров с возможностью нулевой длины TCP и IP;
- проверка на передачу сверхнормативных пакетов;
- проверка опций post-EOL для TCP и IP;

– проверка наличия в Web-сервисах уязвимых сценариев, на базе Basic, Script, JavaScript, Perl и ActiveXo;
– проверка программного обеспечения на закрытие всех известных уязвимостей данной платформы.

## Задания

1. Выбрать задание по варианту из табл.
2. Провести анализ интернет трафика.
3. Провести выборку на массиве данных.
4. Запустить фишинговую угрозу.
5. Собрать дополнительные данные и найти вирус с помощью данного метода.
6. Отобразить результаты в отчёте.

## Пример выполнения работы

## Варианты

## Вопросы для контроля

1. Что такое анализ данных в ИБ?
2. Какие бывают методы анализа?
3. Какое ПО для анализа данных вы знаете?
4. Как работает метод пассивного анализа?
5. Как работает метод активного анализа?
6. Какова надёжность методов анализа данных?
7. Какие особенности, достоинства и недостатки анализа вы знаете?

# ЛАБОРАТОРНА РОБОТА 8

# АРХИВАЦИЯ ДАННЫХ

**Мета роботи:** Выполнить исследование алгоритмов архивации данных. Использовать алгоритм Хаффмана и Лемпеля-Зива для архивации. Восстановить данные после

## Теоретические ведомости

### 1. Архивация данных

**Архивация (сжатие данных)** — есть процесс представления информации в ином виде (перекодирования) с потенциальным уменьшением объёма, требуемого для её хранения. Существует множество классов различных алгоритмов сжатия данных, каждый из которых ориентирован на свою область применения. Источник

Основоположником науки о сжатии информации принято считать *Клода Шеннона*. Его теорема об оптимальном кодировании показывает, к чему нужно стремиться при кодировании информации и насколько та или иная информация при этом сожмется. Кроме того, им были проведены опыты по эмпирической оценке избыточности английского текста. Шенон предлагал людям угадывать следующую букву и оценивал вероятность правильного угадывания. На основе ряда опытов он пришел к выводу, что количество информации в английском тексте колеблется в пределах 0,6–1,3 бита на символ. Несмотря на то, что результаты исследований Шеннона были по-настоящему востребованы лишь десятилетия спустя, трудно переоценить их значение.

**Сжатие данных** — это процесс, обеспечивающий уменьшение объёма данных путём сокращения их избыточности. Сжатие данных связано с компактным расположением порций данных стандартного размера. Сжатие данных можно разделить на два основных типа:

**Сжатие без потерь** (полностью обратимое) — это метод сжатия данных, при котором ранее закодированная порция данных восстанавливается после их распаковки полностью без внесения изменений. Для каждого типа данных, как правило, существуют свои оптимальные алгоритмы сжатия без потерь.

**Сжатие с потерями** — это метод сжатия данных, при котором для обеспечения максимальной степени сжатия исходного массива данных часть содержащихся в нём данных отбрасывается. Для текстовых, числовых и табличных данных использование программ, реализующих подобные методы сжатия, является неприемлемыми. В основном такие алгоритмы применяются для сжатия аудио и видеоданных, статических изображений.

## 2. Алгоритмы архивации данных

**Алгоритм сжатия данных** — это алгоритм, который устраняет избыточность записи данных.

**Отношение сжатия** — одна из наиболее часто используемых величин для обозначения эффективности метода сжатия.

$$\text{Отношение сжатия} = \frac{\text{размер выходного потока}}{\text{размер входного потока}} \tag{8.1}$$

Значение 0,6 означает, что данные занимают 60% от первоначального объема. Значения больше 1 означают, что выходной поток больше входного (отрицательное сжатие, или расширение).

**Коэффициент сжатия** — величина, обратная отношению сжатия.

$$\text{Коэффициент сжатия} = \frac{\text{размер входного потока}}{\text{размер выходного потока}} \tag{8.2}$$

Значения больше 1 обозначают сжатие, а значения меньше 1 расширение.

**Средняя длина кодового слова** – это величина, которая вычисляется как взвешенная вероятностями сумма длин всех кодовых слов.

$$L_{cp} = p_1 \cdot L_1 + p_2 \cdot L_2 + ... + p_n \cdot L_n, \tag{8.3}$$

где $p_n$ – вероятности кодовых слов, $L_1$,$L_2$,$L_3$ – длины кодовых слов.

**Статистические методы** — методы сжатия, присваивающие коды переменной длины символам входного потока, причем более короткие коды присваиваются символам или группам символам, имеющим большую вероятность появления во входном потоке. Лучшие статистические методы применяют кодирование Хаффмана.

**Словарное сжатие** — это методы сжатия, хранящие фрагменты данных в ”словаре” (некоторая структура данных). Если строка новых данных, поступающих на вход, идентична какому-либо фрагменту, уже находящемуся в словаре, в

выходной поток помещается указатель на этот фрагмент. Лучшие словарные методы применяют метод Зива-Лемпела.

Рассмотрим несколько известных алгоритмов сжатия данных более подробно.

**2.1. Алгоритм Хаффмана.** В основе алгоритма Хаффмана лежит идея кодирования битовыми группами. Сначала проводится частотный анализ входной последовательности данных, то есть устанавливается частота вхождения каждого символа, встречащегося в ней. После этого, символы сортируются по уменьшению частоты вхождения.

Основная идея состоит в следующем: чем чаще встречается символ, тем меньшим количеством бит он кодируется. Результат кодирования заносится в словарь, необходимый для декодирования. Рассмотрим простой пример, иллюстрирующий работу алгоритма Хаффмана.

Пусть задан текст «beep boop beer!», рассмотрим таблицу с частотами всех символов:

Таблиця 8.1 — Частота символов

| **Символ** | ’b’ | ’e’ | ’p’ | ’ ’ | ’o’ | ’r’ | ’!’ |
|---|---|---|---|---|---|---|---|
| **Частота** | 3 | 4 | 2 | 2 | 2 | 1 | 1 |

По частоте использования

| **Символ** | ’r’ | ’!’ | ’p’ | ’o’ | ’ ’ | ’b’ | ’e’ |
|---|---|---|---|---|---|---|---|

После этого создадим элементы бинарного дерева для каждого символа и представим их как очередь с приоритетом, в качестве которого будем использовать частоту.

Возьмём первые два элемента из очереди и создадим третий, который будет их родителем. Этот новый элемент поместим в очередь с приоритетом, равным сумме приоритетов двух его потомков. Иначе говоря, равным сумме их частот.

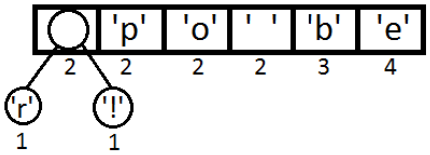


Рисунок 8.1 — 2

Далее будем повторять шаги, аналогичные предыдущему:

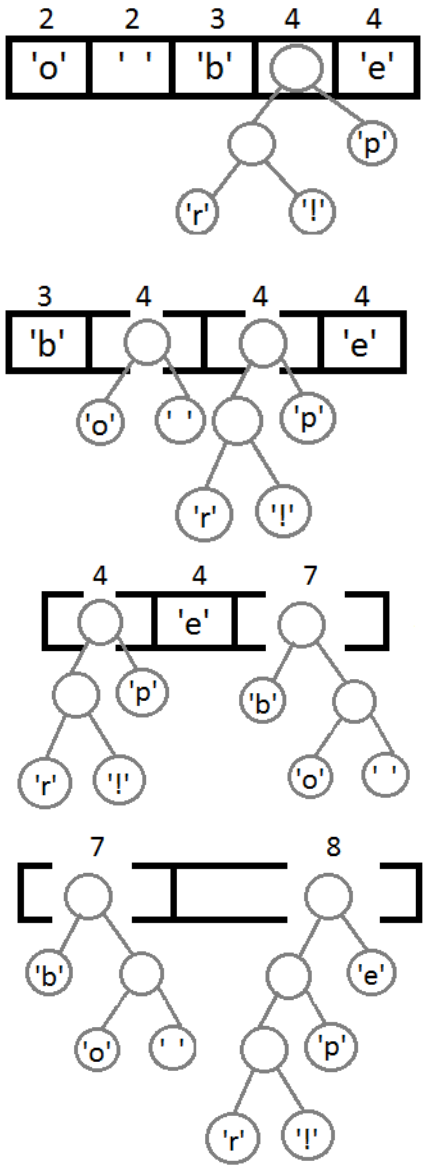


Рисунок 8.2 — Построение дерева

Теперь, после объединения последних двух элементов с помощью их нового родителя, мы получим итоговое бинарное дерево: Осталось присвоить каждому символу его код. Для этого запустим обход в глубину и каждый раз, рассматривая правое поддерево, будем записывать в код 1, а рассматривая левое поддерево — 0.

В результате соответствие символов кодовым значениям получится следующим:

Таблица 8.2 — Кодовые значения символов

| **Сивол** | ‘b’ | ‘e’ | ‘p’ | ‘ ‘ | ‘o’ | ‘r’ | ‘!’ |
|---|---|---|---|---|---|---|---|
| **Кодовое значение** | 00 | 11 | 101 | 011 | 010 | 1000 | 1001 |

Декодирование битов происходит следующим образом: нужно обходить дерево, отбрасывая левое поддерево, если встретилась единица и правое, если

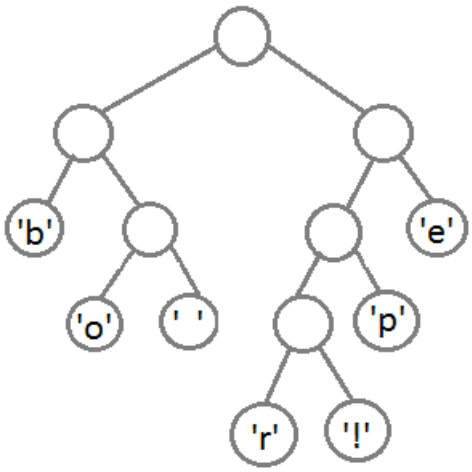


Рисунок 8.3 — 7

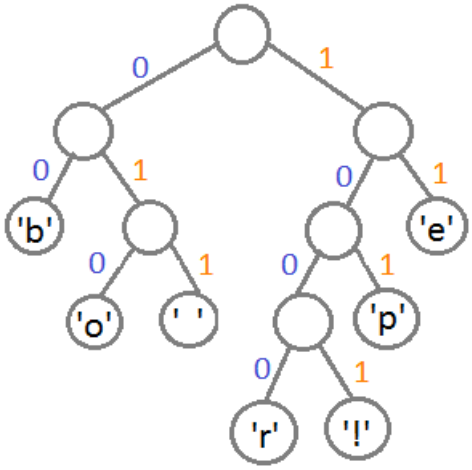


Рисунок 8.4 — 8

встретился 0. Продолжать обход нужно до тех пор, пока не встретим лист, т.е. искомое значение закодированного символа. Например, закодированной строке «101 11 101 11» и нашему дереву декодирования соответствует строка «pepe»

**Входная строка:** «beep boop beer!»

**Входная строка в двоичном виде:** 0110 0010 0110 0101 0110 0101 0111 0000 0010 0000 0110 0010 0110 1111 0110 1111 0111 0000 0010 0000 0110 0010 0110 0101 0110 0101 0111 0010 0010 0001

**Закодированная строка:** 0011 1110 1011 0001 0010 1010 1100 1111 1000 1001

Разница между ASCII-кодировкой строки и её же видом в коде Хаффмана очевидна.

Алгоритм Хаффмана универсальный, его можно применять для сжатия данных любых типов, но он малоэффективен для файлов маленьких размеров (за счет

необходимости сохранения словаря). В настоящее время данный метод практически не применяется в чистом виде, обычно используется как один из этапов сжатия в более сложных схемах. Это единственный алгоритм, который не увеличивает размер исходных данных в худшем случае (если не считать необходимости хранить таблицу перекодировки вместе с файлом).

**2.2. Алгоритм Лемпеля-Зива.** Процесс сжатия выглядит следующим образом. Последовательно считываются символы входного потока и происходит проверка, существует ли в созданной таблице строк такая строка. Если такая строка существует, считывается следующий символ, а если строка не существует, в поток заносится код для предыдущей найденной строки, строка заносится в таблицу, а поиск начинается снова. Например, если сжимают байтовые данные (текст), то строк в таблице окажется 256 (от «0» до «255»). Если используется 10-битный код, то под коды для строк остаются значения в диапазоне от 256 до 1023. Новые строки формируют таблицу последовательно, т. е. можно считать индекс строки ее кодом. Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту. Алгоритм генерирует однозначно декодируемый код за счет того, что каждый раз, когда генерируется новый код, новая строка добавляется в таблицу строк. LZW постоянно проверяет, является ли строка уже известной, и, если так, выводит существующий код без генерации нового. Таким образом, каждая строка будет храниться в единственном экземпляре и иметь свой уникальный номер. Следовательно, при дешифровании при получении нового кода генерируется новая строка, а при получении уже известного, строка ивлекается из словаря. [ссылка]

**Кодирование**

Пусть мы сжимаем последовательность «abacabadabacabae».

1. Тогда, согласно изложенному выше алгоритму, мы добавим к изначально пустой строке «a» и проверим, есть ли строка «a» в таблице. Поскольку мы при инициализации занесли в таблицу все строки из одного символа, то строка «a» есть в таблице.
2. Далее мы читаем следующий символ «b» из входного потока и проверяем, есть ли строка «ab» в таблице. Такой строки в таблице пока нет.
3. Добавляем в таблицу <5> «ab». В поток: <0>;
4. «ba» — нет. В таблицу: <6> «ba». В поток: <1>;
5. «ac» — нет. В таблицу: <7> «ac». В поток: <0>;

6. «ca» — нет. В таблицу: <8> «ca». В поток: <2>;
7. «ab» — есть в таблице; «aba» — нет. В таблицу: <9> «aba». В поток: <5>;
8. «ad» — нет. В таблицу: <10> «ad». В поток: <0>;
9. «da» — нет. В таблицу: <11> «da». В поток: <3>;
10. «aba» — есть в таблице; «abac» — нет. В таблицу: <12> «abac». В поток: <9>;
11. «ca» — есть в таблице; «cab» — нет. В таблицу: <13> «cab». В поток: <8>;
12. «ba» — есть в таблице; «bae» — нет. В таблицу: <14> «bae». В поток: <6>;
13. И, наконец последняя строка «e», за ней идет конец сообщения, поэтому мы просто выводим в поток <4>.

| Текущая строка | Текущий символ | Следующий символ | Вывод | | Словарь | |
|---|---|---|---|---|---|---|
| | | | Код | Биты | | |
| ab | a | b | 0 | 000 | 5: | ab |
| ba | b | a | 1 | 001 | 6: | ba |
| ac | a | c | 0 | 000 | 7: | ac |
| ca | c | a | 2 | 010 | 8: | ca |
| ab | a | b | - | - | - | - |
| aba | b | a | 5 | 101 | 9: | aba |
| ad | a | d | 0 | 000 | 10: | ad |
| da | d | a | 3 | 011 | 11: | da |
| ab | a | b | - | - | - | - |
| aba | b | a | - | - | - | - |
| abac | a | c | 9 | 1001 | 12: | abac |
| ca | c | a | - | - | - | - |
| cab | a | b | 8 | 1000 | 13: | cab |
| ba | b | a | - | - | - | - |
| bae | a | e | 6 | 0110 | 14: | bae |
| e | e | - | 4 | 0100 | - | - |

Рисунок 8.5 — 9

Мы получаем закодированное сообщение «0 1 0 2 5 0 3 9 8 6 4», что на 11-бит короче.

**Декодирование**

Особенность LZW заключается в том, что для декомпрессии нам не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов. Теперь представим, что мы получили закодированное сообщение, приведённое выше, и нам нужно его декодировать. Прежде всего, нам нужно знать начальный словарь, а последующие записи словаря мы можем реконструировать уже на ходу, поскольку они являются просто конкатенацией предыдущих записей.

| Данные | | На выходе | Новая запись | |
|---|---|---|---|---|
| Биты | Код | | Полная | Частичная |
| 000 | 0 | a | - - | 5: a? |
| 001 | 1 | b | 5: ab | 6: b? |
| 000 | 0 | a | 6: ba | 7: a? |
| 010 | 2 | c | 7: ac | 8: c? |
| 101 | 5 | ab | 8: ca | 9: ab? |
| 000 | 0 | a | 9: aba | 10: a? |
| 011 | 3 | d | 10: ad | 11: d? |
| 1001 | 9 | aba | 11: da | 12: aba? |
| 1000 | 8 | ca | 12: abac | 13: ca? |
| 0110 | 6 | ba | 13: cab | 14: ba? |
| 0100 | 4 | e | 14: bae | - - |

Рисунок 8.6 — 10

**Достоинства и недостатки**

+ Не требует вычисления вероятностей встречаемости символов или кодов.
+ Для декомпрессии не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.
+ Данный тип компрессии не вносит искажений в исходный графический файл, и подходит для сжатия растровых данных любого типа.
- Алгоритм не проводит анализ входных данных поэтому не оптимален.

## Задания

1. Взять данные соответственно варианту из табл.
2. Удалить лишнюю информацию методом Хаффмана.
3. Провести операцию методом Лемпеля-Зива.
4. Сравнить результаты проведённых операций.
5. Описать актуальность архивации.
6. Сделать выводы по применению методов сжатия в различных криптосистемах.

## Пример выполнения работы

## Варианты

## Вопросы для контроля

1. Что такое архивация данных?
2. Цель архивации?
3. Какие Вы знаете методы архивации?
4. Опишите принцип дерева Хаффмана.
5. Опишите алгоритм LZ77 или его аналог.
6. Сферы применения заданных алгоритмов.
7. Как выбрать алгоритм, если данные заранее известны?

# РЕКОМЕНДАЦІЇ

# ДОДАТОК А

# ДОПОЛНЕНИЕПЕРВОЕ

## Очередной подраздел приложения

## И ещё один подраздел приложения