# Вопросы

💦 - возможно требуется дополнить вопрос

🍏 - закончен

## 💩 - не закончен

**АКЦИЯ: Приведи друга. Приведи друга, и пиши сам чуть меньше**

Прошу оформлять так, чтобы удобно было читать (если скрин формулы - отмасштабируйте перед скрином и фоткайте в высоком разрешении, чтобы не резало), воздержитесь от скринов текста с методы (с лекциями чуть лучше), т.к. низкое разрешение и плохая читаемость на телефоне и т.д.

Если алгоритм - желательно фоточки, если возможно пример. Делайте так, чтобы открывать методу не нужно было, когда видишь вопрос. Желательно отделять подразделы и важные элементы жирным текстом/подчеркиванием/курсивом/цветом



В идеале убирать сложный длиннотекст и заменять попроще и покороче, где это возможно. Не стесняться многоуровневых списков там, где они удобны (классификация и т.д.)

## 💦1. Сущность проблемы информационной безопасности и надежности систем.

***Информация*** – сведения (данные) о внутреннем и окружающем нас мире, воспринимаемые и передаваемые людьми или техническими устройствами.

**Информационная (информационно-вычислительная) система** – организационно упорядоченная совокупность документов, технических средств и информационных технологий, реализующая информационные (информационно-вычислительные) процессы.

**Информационные процессы** – процессы сбора, накопления, хранения, обработки (переработки), передачи и использования информации

**Информационные ресурсы** – отдельные документы или массивы документов в информационных системах.

**Информационные технологии**— процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

**Доступ** – специальный тип взаимодействия между объектом и субъектом, в результате которого создается поток информации от одного к другому.

**Несанкционированный доступ** – доступ к информации, устройствам ее хранения и обработки, а также к каналам передачи, реализуемый без ведома (санкции) владельца и нарушающий тем самым установленные правила доступа.

**Объект** – пассивный компонент системы, хранящий, перерабатывающий, передающий или принимающий информацию; примеры объектов: страницы, файлы, папки, директории, компьютерные программы, устройства (мониторы, диски, принтеры и т.д.)

**Субъект** – активный компонент системы, который может инициировать поток информации; примеры субъектов: пользователь, процесс либо устройство.

**Безопасность ИВС** – свойство системы, выражающееся в способности системы противодействовать попыткам несанкционированного доступа или нанесения ущерба владельцам и пользователям системы при различных умышленных и неумышленных воздействиях не нее.

**Защита информации** – организационные, правовые, программно-технические и иные меры по предотвращению угроз информационной безопасности и устранению их последствий.

**Информационная безопасность** систем – свойство информационной системы или реализуемого в ней процесса, характеризующее способность обеспечить необходимый уровень своей защиты.

## 💦2. Характеристики и параметры ИС и ИВС.

**Информационная безопасность** (information security) — все аспекты, связанные с определением, достижением и поддержанием конфиденциальности, целостности, доступности информации или средств её обработки (**СIA**):

· конфиденциальность (**C**onfidentiality) — состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на нее право;

· целостность (**I**ntegrity) — избежание несанкционированной модификации информации;

· доступность (**A**vailability) — избежание временного или постоянного сокрытия информации от пользователей, получивших права доступа.

## 🍏3. Характеристика угроз безопасности современным ИС и ИВС.

| Тип угрозы | Примеры |
| --- | --- |
| **Имитация**  **(Spoofing)** | Подделка электронных сообщений  Подделка ответных пакетов при аутентификации |
| **Фальсификация (Tampering)** | Модификация данных, передаваемых по сети  Модификация файлов |
| **Отречение (Repudiation)** | Удаление критичного файла или совершение покупки с последующим отказом признавать свои действия |
| **Раскрытие информации (Information disclosure)** | Несанкционированный доступ или незаконная публикация конфиденциальной информации |
| **Отказ в обслуживании (Denial of service)**  **Проблема botnet** | Заполнение сети пакетами «SYN»  Загрузка сетевого ресурса большим количеством поддельных пакетов ICMP |
| **Повышение привилегий**  **(Elevation of privilege)** | Получение системных привилегий через атаку с переполнением буфера  Незаконное получение административных прав либо незаконная их передача с целью наживы |

## 💦4. Характеристика методов и средств защиты информации от несанкционированного доступа.

Классификация методов и средств обеспечения информационной безопасности:

1. Организационные

2. Правовые

3. Технические

4. Аппаратные, программные, аппаратно-программные

Идентификация – идентификатор.

Аутентификация – подтверждение аутентичности (пароль, паспорт)

Авторизация – подтверждение привилегии

Методы биометрические аутентификации:

Отпечаток пальцев, сканирование сетчатки глаза, лица.

## 🍏5. Методология разработки и реализации политики безопасности современных ИС и ИВС

**Политика информационной безопасности** (ПИБ) организации или учреждения – совокупность правил, процедур, практических методов, руководящих принципов, документированных управленческих решений, направленных на защиту информации и связанных с ней ресурсов и используемых всеми сотрудниками организации или учреждения в своей деятельности.

Процесс разработки политики информационной безопасности:

1. **Обоснование актуальности**, цели и задачи разработки ПИБ в организации (учреждении).
2. **Объекты защиты**. Описание структуры организации (учреждения)
3. **Основные угрозы и их источники**. Анализ потенциальных угроз
4. **Оценка угроз, рисков и уязвимостей**. Анализ ценности ресурсов, оценка значимости угроз, а также эффективности существующих и планируемых средств защиты
5. **Разработка и введение мер, методов и средств обеспечения требуемого уровня защищенности информационных ресурсов**.

Для создания эффективной системы информационной безопасности организации или учреждения целесообразно разработать:

* концепцию информационной безопасности, которая определяет в целом цели политики и основные ее принципы в увязке со статусом, целями и задачами организации или учреждения;
* стандарты (менеджмента качества) – правила и принципы защиты информации по каждому конкретному направлению деятельности;
* процедуры – описание конкретных действий по защите информации при работе с ней: персональных данных, порядка доступа к информационным носителям, системам и ресурсам;
* инструкции, содержащие подробное описание (алгоритмы) действий по организации информационной защиты и обеспечению разработанных стандартов и процедур;
* план мероприятий по обучению персонала и тестированию знаний сотрудников, имеющих доступ к информационным ресурсам.

Для построения политики ИБ рассматривают следующие направления защиты ИВС:

* защита объектов ИВС;
* защита процессов, процедур и программ обработки информации;
* защита каналов связи;
* подавление побочных электромагнитных излучений;
* управление системой защиты.

Организационная защита обеспечивает:

* организацию охраны, режима, работу с кадрами и с документами;
* использование технических средств безопасности (например, простейших дверных замков, магнитных или иных карт и др.), информационно-аналитическую деятельность по выявлению внутренних и внешних угроз.

**Концепция информационной безопасности** определяет в целом цели политики и основные ее принципы в увязке со статусом, целями и задачами организации или учреждения;

Основными разделами концепции информационной безопасности могут быть следующие:

* определение ИБ (или СУИБ);
* структура информационной системы организации (учреждения) и вытекающая из этого структура системы обеспечения информационной безопасности;
* безопасность информации: принципы и стандарты;
* оценка рисков информационным ресурсам в организации (учреждении);
* описание основных механизмов контроля безопасности;
* обязанности и ответственность каждого отдела, управления или департамента, каждого сотрудника в реализации разработанной и утвержденной политики безопасности;
* обязанности лица (администратора безопасности), ответственного за организацию оперативного контроля и управления политикой безопасности;
* ссылки на документы об информационной безопасности, действующие на территории Республики Беларусь.

**Оценка рисков**

**Фактор, воздействующий на ИВС**, – это явление, действие или процесс, результатом которых может быть утечка, искажение, уничтожение данных, блокировка доступа к ним, повреждение или уничтожение системы защиты.

**Внутренние дестабилизирующие факторы,** влияющие на:

1) на программные средства (ПС):

* некорректный исходный алгоритм;
* неправильно запрограммированный исходный алгоритм (первичные ошибки);

2) на аппаратные средства (АС):

* системные ошибки при постановке задачи проектирования;
* отклонения от технологии изготовления комплектующих изделий и АС в целом;
* нарушение режима эксплуатации, вызванное внутренним состоянием АС.

**Внешние дестабилизирующие факторы,** влияющие на:

1) на программные средства:

* неквалифицированные пользователи;
* несанкционированный доступ к ПС с целью модификации кода;

2) на аппаратные средства:

* внешние климатические условия;
* электромагнитные и ионизирующие помехи;
* перебои в электроснабжении;
* недостаточная квалификация обслуживающего персонала.

**Риски и их оценка**

**Риск = Ущерб\*Вероятность**

**Ущерб следует из угрозы** *(Т.е. Риск > Угроза, для понимания разницы между Риском и Угрозой)*

**Мероприятия по оценке рисков**:

* идентификация значимых угроз и уязвимостей для идентифицированных ресурсов;
* оценка вероятности возникновения угроз и уязвимостей;
* вычисление рисков; оценивание рисков по заранее определенной шкале риска.

**Угрозы**:

* Естественные - физические и природные воздействия
* Искусственные - вызваны деятельностью человека

---

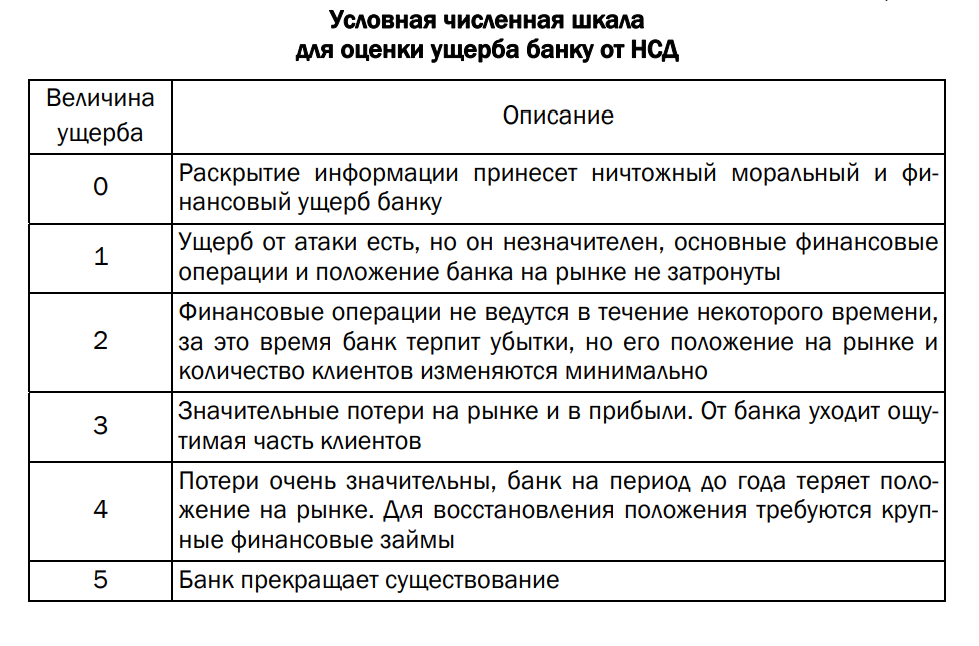
* Преднамеренные
* Непреднамеренные

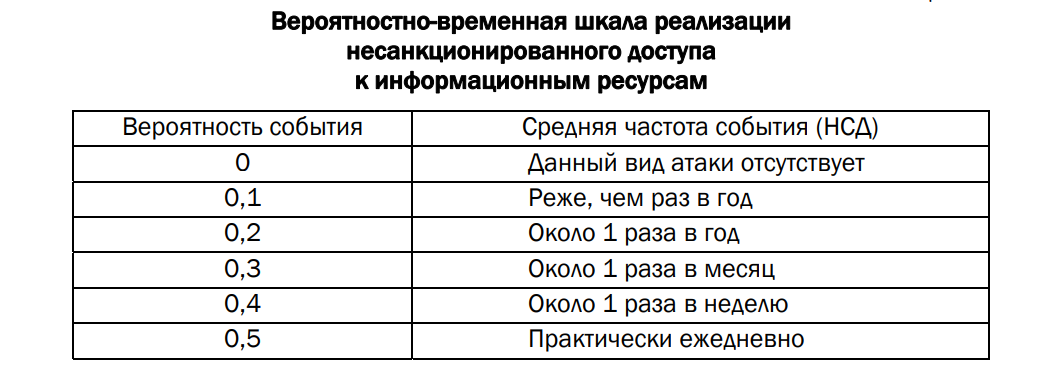
Цель оценивания рисков состоит в определении характеристик рисков для информационной системы и ее ресурсов. На основе таких данных могут быть выбраны необходимые средства управления ИБ.

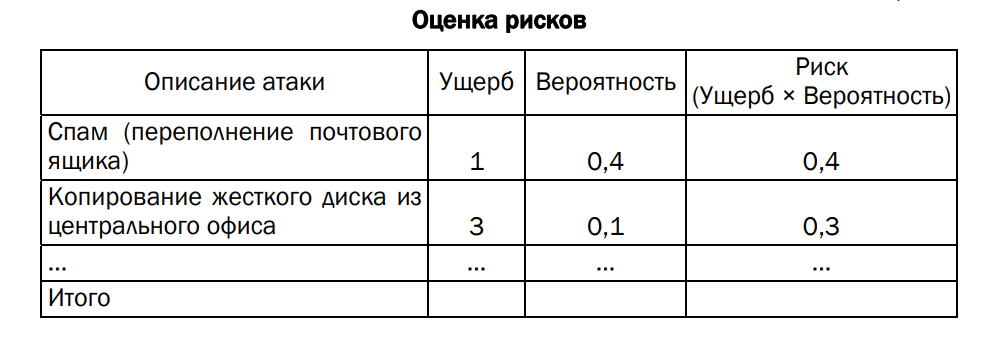
При оценивании рисков учитываются:

* ценность ресурсов;
* оценка значимости угроз;
* эффективность существующих и планируемых средств защиты.

**Шкала численной оценки ущерба от несанкционированного доступа (НСД)**







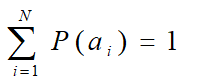
## 🍏6. Энтропия источника сообщения. Энтропия Шеннона.

***Алфавит*** – конечная совокупность символов (знаков), с помощью которых можно представить любое сообщение в ИС.

Мощность алфавита – количество элементов (конечно/бесконечно) - **N(A).**

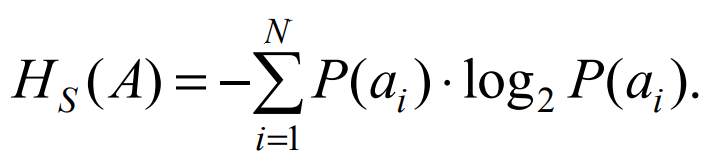
Вероятность того, что произвольный символ ξ произвольного документа (текст, база данных, текст программы) будет буквой «аi»:

P (ξ = аi) = p(аi)



(Введено Шенноном) **Энтропия** – характеристика информативности информации.

***Энтропия Шеннона:***



***В двоичном алфавите стремится к 1, а вероятности 0 и 1 к 0.5***

**Энтропия алфавита** - информационная характеристика алфавита - показывает, какое количество информации (бит) приходится в среднем на один символ алфавита.

*-упомянуть что Энтропия в целом также является мерой защищенности пароля*

**Чем больше возможных исходов, тем больше энтропия алфавита.**

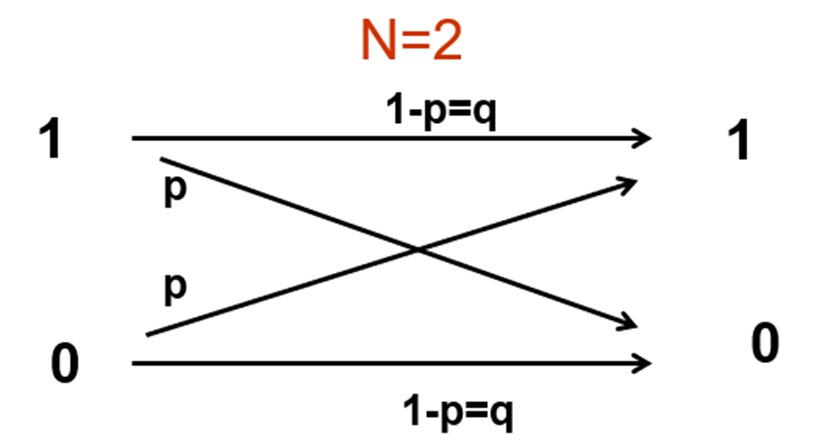
## 🍏7. Энтропия источника сообщения. Энтропия Хартли.

(см в.6)

При p(ai) = const - вероятности всех символов одинаковые – **Энтропия Хартли**

| **Hc (A) = log2 N** |
| --- |

## 🍏8. Двоичный канал передачи информации.



Двоичный канал передачи информации строится на основе двоичного алфавита: А = {0, 1}. ДСК - канал в котором вероятности равны: 1 -> 1 = 0 -> 0 и 1 -> 0 = 0 -> 1.

Записывается как:

p(1) = p(0), p(0 | 1) = p(1 | 0)

2 свойства:

* Дискретность - значения конечны (0, 1)
* Симметричность(?)

## 🍏9. Энтропия двоичного алфавита.

***Энтропия Хартли:***

log2(2) = 1

***Энтропия Шеннона:***

**В двоичном алфавите стремится к 1, а вероятности 0 и 1 к 0.5 -**

можно представить p(0) = 1-p(1) и подставить в формулу

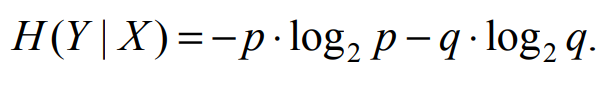
H(A2) = -(1-р(1))\*log2(1-р(1)) - р(1)\*log2(р(1)) (см 10)

(также см 10 про эффективную энтропию)

## 💦10. Условная энтропия. Энтропийная оценка потерь при передаче информации.

Если р > 0 (ДСК, p - вероятность ошибки), то это можно трактовать как неоднозначность (по Шеннону – *equivocation*) между переданным и принятым сообщениями.

Эта неоднозначность определяется как **условная энтропия (частная)**:

****

**Условная энтропия** – количество теряемой информации при передаче каждого символа сообщения.

Шеннон показал, что эффективная информация на выходе канала относительно входной в расчете на 1 символ (**Эффективная энтропия** алфавита) составляет:

**Не = H(X) – H(Y|X) *(для двоичного H(X) = 1)***

**Интересные факты**, но Урбанович спрашивал часто:

При вероятности ошибки **p = 0**, условная энтропия ДСК будет H(Y|X) = 0, эффективная - He = 1. То есть **информация не теряется**, а эффективная энтропия *= энтропии Хартли.*

При вероятности ошибки **p = 1** (все сообщения инвертированы), условная вероятность **ТАКЖЕ** будет = 0, потому что по факту мы можем все сообщения инвертировать назад, и получится что **никакой информации потеряно не было**. Эффективная энтропия тогда также = 1 (равняется энтропии Хартли)

## 💦11. Методы и средства структурной, информационной и временной избыточности в ИВС

***Используемые методы:***

1) увеличение наработки

2) снижение интенсивности отказов

3) улучшение восстанавливаемости

4) резервирование

Третью и четвертую группы можно объединить под единым названием – **избыточных методов:**

* **простая структурная избыточность** - структурное резервирование
* системы с **временной избыточностью** – системы с повторениями (передачи)
  + Реализуется введением в структуру ТС накопительного звена, позволяющего в течение определенного времени выполнять основную функцию при отказе элемента за накопительным звеном. Если избыточное время будет выше времени восстановления отказавшего элемента, то функция по назначению будет выполняться непрерывно и даже при отказе.
* **информационная избыточность** – обусловливает возможность применения функций сжатия информации

## 💦12. Помехоустойчивое кодирование информации. Основные понятия. Назначение.

**(См 14)**

**Надежность** является комплексным свойством, включающим в себя единичные свойства: ***безотказность, ремонтопригодность, сохраняемость, долговечность.***

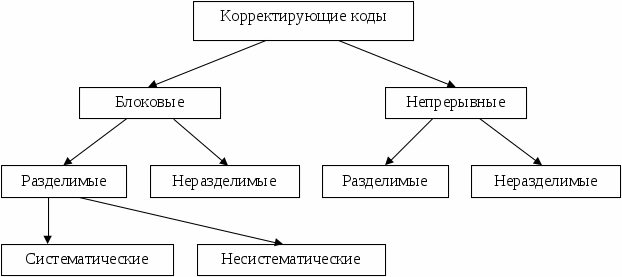
**Безотказность** – это свойство технического объекта непрерывно сохранять работоспособное состояние в течение некоторого времени (или наработки). Наработка, как правило, измеряется в единицах времени.

**Ремонтопригодность** – это свойство технического объекта, заключающееся в приспособленности к поддержанию и восстановлению работоспособного состояния путем технического обслуживания, ремонта (или с помощью дополнительных, избыточных технических средств, функционирующих параллельно с объектом).

**Назначение** - обеспечение качества и надежности ИВС.

## 💦13. Помехоустойчивое кодирование информации. Классификация кодов

***(См 14)***

******

***Классификация кодов (требуется адекватно реформатнуть это)***

* **Блочные коды** —каждому сообщению из k (Xk)символов (бит) сопоставляется блок из n символов (кодовый вектор Xn длиной n=k + r).
* **Непрерывные** (рекуррентные, цепные, свёрточные) коды -непрерывная последовательность символов, не разделяемая на блоки. Передаваемая последовательность образуется путём размещения в определённом порядке проверочных символов между информационными символами исходной послед-ти.
* **Систематические** коды характеризуются тем, что сумма по модулю 2 двух разрешённых кодовых комбинаций кодов снова даёт разрешённую кодовую комбинацию.
* **Несистематические коды** не обладают отмеченными выше свойствами (к ним относятся итеративные коды).
* **Линейные** коды – проверочные (избыточные) символы вычисляются как линейная комбинация информационных символов; для кодов принимается обозначение [n, k]-код.
* **Нелинейные** коды являются противоположностью линейным.
* **Циклические коды** – относятся к линейным систематическим.
  + Основное свойство, давшее им название, состоит в том, что каждый вектор, получаемый из исходного кодового вектора путём циклической перестановки его символов, также является разрешённым кодовым вектором. Принято описывать циклические коды при помощи порождающих полиномов G(X) степени r

**Линейные блочные** коды – это класс кодов с контролем четности, которые можно описать парой чисел (п, k).

## 💦14. Теоретические основы избыточного кодирования информации

**Информационная избыточность сообщений R** определяется по формуле:

| **R = 1 - H / log2 k**  где k — число букв алфавита, а Н — энтропия источника на букву сообщения |
| --- |

Обычно данные делятся на блоки длиной k - размерность кода. Код обозначают записью (n, k) - т.е. код использует n символов для хранения k символов сообщения.

| Сообщение **Xk** (**k** – длина сообщения, символов или бит), называется **информационным словом**.  Избыточные символы длиной **r** символов (бит) составляющие избыточное слово **Xr.** Также называются **битами четности, контрольными битами, паритетами**… Не несут новой информации!  Слово **Xn** длиной **n=k+r** символов **Xn = XkXr** называется **кодовым словом**.  **Rи=r/k** — **относительная избыточность кода**  **Rс=k/n** - **cкорость кода** или **степень кодирования**  **Вес Хемминга** произвольного двоичного слова Х (w(X)) равен количеству ненулевых символов в слове.  w(X=1101) = 3  **Расстояние Хемминга** или кодовое расстояние (d) между двумя произвольными словами (X,Y) одинаковой длины равно количеству позиций, в которых X и Y отличаются между собой.  **d (X,Y) = w(X xor Y)**  **Пример:**  X=101,Y=111. d(X,Y) = 1. |
| --- |

| **d - расстояние Хемминга** между двумя **кодовыми словами**  Количество **ошибок, обнаруживаемое кодом**:    Корректирующие способности кода определяются исключительно **минимальным кодовым расстоянием (dmin)** между двумя произвольными кодовыми словами, принадлежащими коду: |
| --- |

| **Теорема. Минимальное кодовое расстояние** линейного кода равно **минимальному весу ненулевых кодовых слов** |
| --- |

Выбор кода определяется вероятностью ошибки в канале, р (чем больше р, тем больше d , т.е. следует выбирать код с большим количеством избыточных битов, однако это снижает Rс )

| **Определение**. Пропускная способность ДСК с вероятностью ошибки р равна  С(р) = 1 +р log2 р + q log2 q |
| --- |

| **Теорема Шеннона**. Для любого ДСК и любого ε >0 существует (n,k)- двоичный код со скоростью Rс, если Rс< C(p), n достаточно велико и р < ε. |
| --- |

**Основная проблема теории кодирования**:

Найти коды с большими d И Rс (задача оптимизационная) - т.е. коды исправляющие большое количество ошибок при как можно меньшей избыточности

## 🍏15. Алгоритм использования корректирующего кода

На стороне ИС

1.Построить проверочную матрицу Hn,k для заданного k (Хk)

2.Вычислить символы избыточного слова Хr

**HT \* *Xn = H \* (Xn)T = 0***

3.Cформировать кодовое слово Хn = х1,х2,…,хk,xk+1,…xk+r и осуществить его передачу

На стороне ПС

1.Получение сообщения (Yn = y1,y2,…,yk,yk+1,…yk+r )

2.Вычисление синдрома (на основе (5) ), используя ту же Hn,k :

S =HT \*Yn = H \*(Yn)T =HT \*(Хn +En) =HT \*Хn + HT \*En = HT \*En (10)

для этого вычисляем Yr’= yri’  : yri’ = yk+i ‘= Σ hij \*yj (11)

и далее: S = s1,s2,…sr  , где si = yk+i + yri’ (12)

4. Анализ (декодирование синдрома) – определение местоположения ошибочного бита (En посредством S xor hm )

5. Исправление ошибки: Хn = Yn + En

## 🍏16. Декодирование кодовых слов. Поиск и исправление ошибок.

При декодировании кодового слова необходимо в первую очередь определить **синдром**, чтобы понять, были ли допущены ошибки при передаче.

**Поиск ошибки**

Прежде всего нужно вычислить новый набор избыточных символов Y’r, используя кодовое слово Yn и проверочную матрицу H

Y’r1 = H11 \* Y1 xor H12 \* Y2 xor … xor H1k \* Yk и тд

Далее вычисляем сам синдром по следующей формуле



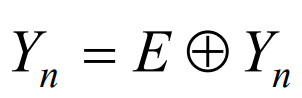
Ненулевой синдром => ошибки есть, нулевой => ошибок нет.

Находим столбец проверочной матрицы, который равен S, если такой присутствует, значит ошибка выявлена верно и ошибочным является символ на позиции с номером столбца.

**Исправление ошибки**

В результате декодирования синдрома получается вектор ошибки E

Исправление ошибочного бита достигается простым сложением по модулю 2 вектора Е и кодового слова Yn:



**Пример**

Есть кодовое слово Yn = 1001**0**00 (ошибочный символ - 3 слева)

Yr = 000

Y’r1 = 0\*1 xor 1\*0 xor 1\*0 xor 1\*1 xor 1\*0 xor 0\*0 xor 0\*0 = 1

Y’r2 = 0

Y’r3 = 0

=> Y’r = 100

S = Yr xor Y’r = 000 xor 100 = 100 => есть ошибка

Ищем в проверочной матрице столбец 100

****

Столбец h5 = S = 100 => ошибка на 5 позиции кодового слова => E = 0000100

Исправляем ошибку:

Yn = E xor Yn = 0000100 xor 1001000 = 1001100

## 🍏17. Код Хемминга с минимальным кодовым расстоянием dmin=3.

Код Хемминга относится к классу линейных блочных кодов.

**Линейные блочные коды** – это класс кодов с контролем четности, которые можно описать парой чисел (n, k).

Для формирования r проверочных символов (кодирования), т. е. вычисления проверочного слова Xr, используется порождающая матрица G: совокупность базисных векторов будем далее записывать в виде матрицы G размерностью k×n с единичной подматрицей (I) в первых k строках и столбцах:

G = [P|I ]

Более точно матрица G называется порождающей матрицей линейного корректирующего кода в приведенно-ступенчатой форме. Кодовые слова являются линейными комбинациями строк матрицы G (кроме слова, состоящего из нулевых символов).

Кодирование заключается в умножении вектора сообщения Хk длиной k на порождающую матрицу по правилам матричного умножения (все операции выполняются по модулю 2). Очевидно, что при этом первые k символов кодового слова равны соответствующим символам сообщения, а последние r символов образуются как линейные комбинации первых.

Для всякой порождающей матрицы G существует матрица Н размерности r×n, задающая базис нулевого пространства кода и удовлетворяющая равенству

G\*HT = 0

. Справедливо также

Xn\*HT = H\*(Xn)T = 0

В последнем выражении символ «T» означает *транспонирование*, а Xn = x1, x2, …, xn.

Матрица Н, называемая *проверочной*, равна



В коде Хемминга с минимальным кодовым расстоянием dmin = 3 проверочная матрица Н имеет классический вид и состоит из двух подматриц: P’ размером k×r и I размером r×r соответственно.

В последнем выражении I – единичная матрица порядка r (r×r).

Количество r избыточных (проверочных) символов кодового слова определяется из следующей простой логической цепи рассуждений.

Общее число всех возможных комбинаций 2^r должно удовлетворять неравенству

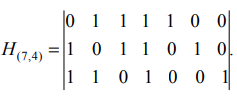


в силу того, что



Присутствие цифры «1» в приведенных выражениях соотносит ее с нулевым вектор-столбцом, который в матрице не используется.

Например, для (7, 4)-кода Хемминга проверочная матрица в упорядоченном виде выглядит так:



Результат умножения сообщения на выходе канала передачи (Yn) или (что равнозначно) сообщения, считываемого из памяти, на проверочную матрицу (Н) называется **синдромом (вектором ошибки) S:**



где Yn = y1, y2, …, yn – принятый вектор (сообщение на выходе канала), полученный после передачи либо считывания из памяти. Вектор Yn обычно представляют в следующем виде:



где Еn = е1, е2, …, еn – вектор ошибки.

Синдром – это результат проверки четности, выполняемой над сообщением Yn для определения его принадлежности заданному набору кодовых слов. При положительном результате проверки синдром S равен 0, т. е. Yn = Хn. Если Yn содержит ошибки, которые можно исправить, то синдром имеет определенное ненулевое значение, что позволяет обнаружить и исправить конкретную ошибочную комбинацию.

**Важно запомнить, что в силу этих выражений ненулевой синдром всегда равен сумме по модулю 2 тех векторстолбцов матрицы Н, номера которых соответствуют номерам ошибочных битов в слове Yn.**

## 💦18. Код простой четности. Особенности программной реализации

Какие особенности реализации???? программные

Простейший избыточный код;

Основан на контроле четности (либо нечетности) единичных символов в сообщении.

Количество избыточных символов r всегда равно 1 и не зависит от k.

Значение этого символа будет нулевым, если сумма всех символов кодового слова по модулю 2 равна нулю – при контроле четности.

Проверка четности – очень простой метод для обнаружения ошибок в передаваемом пакете данных. С помощью данного кода мы не можем восстановить данные, но можем обнаружить только лишь одиночную ошибку.

В каждом пакет данных есть один бит четности, или, так называемый, паритетный бит. Этот бит устанавливается во время записи (или отправки) данных, и затем рассчитывается и сравнивается во время чтения (получения) данных. Он равен сумме по модулю 2 всех бит данных в пакете. То есть число единиц в пакете всегда будет четно . Изменение этого бита (например с 0 на 1) сообщает о возникшей ошибке.

Пример:

Начальные данные: 1111

Данные после кодирования: 11110 ( 1 + 1 + 1 + 1 = 0 (mod 2) )

Принятые данные: 10110 (изменился второй бит)

Как мы видим, количество единиц в принятом пакете нечетно, следовательно, при передаче произошла ошибка.

Как говорилось ранее, этот метод служит только для определения одиночной ошибки. В случае изменения состояния двух битов, возможна ситуация, когда вычисление контрольного бита совпадет с записанным. В этом случае система не определит ошибку, а это не есть хорошо. К примеру:

Начальные данные: 1111

Данные после кодирования: 11110 ( 1 + 1 + 1 + 1 = 0 (mod 2) )

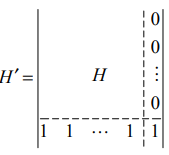
Принятые данные: 10010 (изменились 2 и 3 биты)

В принятых данных число единиц четно, и, следовательно, декодер не обнаружит ошибку.

Так как около 90% всех нерегулярных ошибок происходит именно с одиночным разрядом, проверки четности бывает достаточно для большинства ситуаций.

## 🍏19. Код Хемминга с минимальным кодовым расстоянием dmin=4.

Корректирующая способность кода Хемминга с dmin = 3 может быть увеличена введением дополнительной проверки на четность. В этом случае проверочная матрица будет иметь вид



Так, минимальное кодовое расстояние такого кода будет равно 4: dmin = 4. Такой код может исправлять все единичные ошибки с одновременным обнаружением всех двойных в анализируемом кодовом слове.

При этом нужно помнить, что вид матрицы не соответствует ее каноническому представлению, поскольку во всех столбцах единичной матрицы, кроме последнего, будет по 2 единицы.

Для придания матрице канонического вида необходимо сложить посимвольно все строки между собой и результат сложения записать в последнюю строку (под горизонтальной линией).

## 💦20. Составной код. Итеративные коды.

Укоротить? Многомерные итеративные коды?

Определенно укоротить

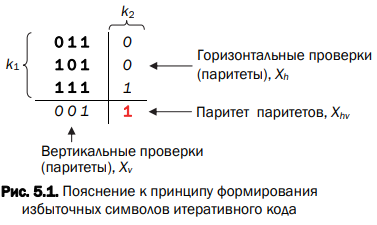
Итеративные коды относятся к классу кодов произведения. Кодом произведения двух исходных (базовых) помехоустойчивых кодов называется такой многомерный помехоустойчивый код, кодовыми последовательностями которого являются все двумерные таблицы со строками кода (k1) и столбцами кода (k2).

Итеративные коды могут строиться на основе использования дву-, трехмерных матриц (таблиц) и более высоких размерностей. Каждая из отдельных последовательностей информационных символов кодируется определенным линейным кодом (групповым или циклическим). Получаемый таким образом итеративный код также является линейным.

Простейшим из итеративных кодов является **двумерный код с проверкой на четность по строкам и столбцам**. *(см 18 - в общем случае, можно, например, кодировать каждую строку и столбец кодом Хемминга, и будет битов четности больше 1 на строку/столбец. Просто интересные факты*)

Основное достоинство рассматриваемых кодов – простота как аппаратной, так и программной реализации. Основной недостаток – сравнительно высокая избыточность.

В упомянутой двумерной матрице кодовые слова записываются в виде таблицы. Проверочные символы вычисляются исходя из того, что строки и столбцы должны содержать четное (нечетное) число единиц. Например, при кодировании информационного слова Хk = **011101111** с помощью таблицы с четностью по строкам и столбцам получим избыточные символы Хr = Xh, Xv, Xhv = 0010011, как показано на рис. 5.1 (информационные символы выделены жирным шрифтом, а проверочные – курсивом)

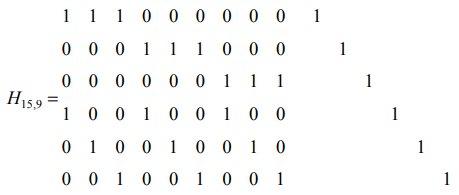


В соответствии с рис. 5.1 кодовое слово будет иметь следующий вид: Xn = **011101111**0010011. Как видно, избыточные символы (называемые также паритетами) в приведенном кодовом слове в принятом порядке (Xh, Xv, Xhv) записываются сверху вниз, справа налево. Возможен обратный или иной порядок. Важно только, чтобы при декодировании сообщения использовался аналогичный порядок следования паритетов. Символ Xhv (паритет паритетов) равен сумме по модулю 2 символов информационного слова Xk, а также проверочных символов Xv и Xh.

Поскольку двумерная матрица формируется как комбинация двух кодов простой четности (по каждому измерению), каждый из которых характеризуется минимальным кодовым расстоянием dmin = 2, то полученный итеративный код (r = k1 + k2) будет характеризоваться минимальным кодовым расстоянием, равным произведению dmin по строкам и по столбцам, т. е. 4.

Использование символа Xhv обеспечивает минимальное кодовое расстояние такого итеративного кода dmin (r = k1 + k2 + 1) на единицу больше. В этом легко обнаруживается сходство кода с кодом Хемминга при dmin = 4.

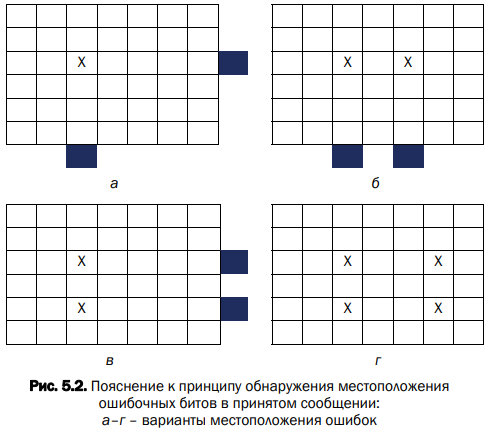
Нетрудно также представить процесс вычисления проверочных символов кодового слова для примера на рис. 5.1 с помощью проверочной матрицы Хемминга и соотношения (4.4). Для указанного примера проверочная матрица кода с dmin = 3 выглядит так:



Передачу символов кода обычно осуществляют последовательно символ за символом, от одной строки к другой, либо параллельно целыми строками.

Как показано на рис. 5.1, проверочный символ есть свертка по модулю 2 информационных символов, записанных в соответствующие строку или столбец матрицы.

Декодирование начинают сразу, не ожидая поступления всего блока информации. Проверка соответствия избыточных символов полученного слова (Yr = Yh, Yv, Yhv либо Yr = Yh, Yv) при декодировании позволяет обнаружить любое нечетное число искаженных символов, расположенных в одной строке или в одном столбце. Формально такое декодирование осуществляется сравнением принятых (Yh, Yv, Yhv) и вновь вычисленных (Y’h, Y’v, Y’hv) для полученного слова паритетов. В упрощенной форме это показано на рис. 5.2. Определение местоположения одиночной ошибки по строке указывает на наличие ошибки в этой строке матрицы, а проверка по столбцу – конкретный символ (рис. 5.2, а).



Однако этим кодом не могут быть установлены местоположения многократных ошибок, имеющих четное число искаженных символов как по строкам, так и по столбцам (рис. 5.2, б, в). Простейшая необнаруживаемая ошибка содержит четыре искаженных символа, расположенных в вершинах прямоугольника или квадрата (рис. 5.2, г). Это происходит из-за того, что четность (паритет) по строкам и по столбцам матрицы не нарушается.

## 💩21. Оценка эффективности использования корректирующего кода в ИВС

## 🍏22. Особенности использования перемежителей в ИВС

**Перемежитель** - устройство (реализовано аппаратно) или программное средство, которое определенным образом перемешивает (меняет местами) символы передаваемого сообщения (или кодового слова).

Разработка и использование перемежителей в ИС – необходимость разнесения расположенных рядом (сгруппированных) ошибок в сообщении («размазать» ошибки по сообщению) с целью упрощения и сокращения во времени процедуры исправления таких ошибок сравнительно простыми кодами.

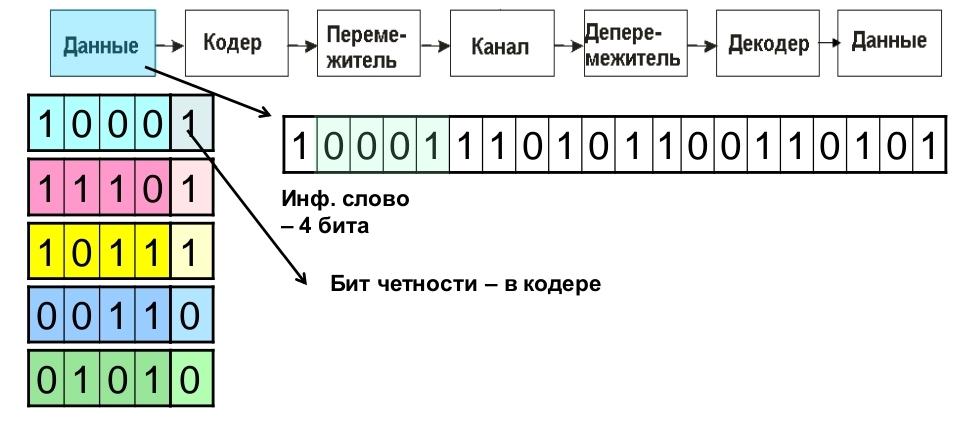
**Глубина перемежения** - максимальное расстояние – в битах, - на которое разносятся соседние символы входной последовательности

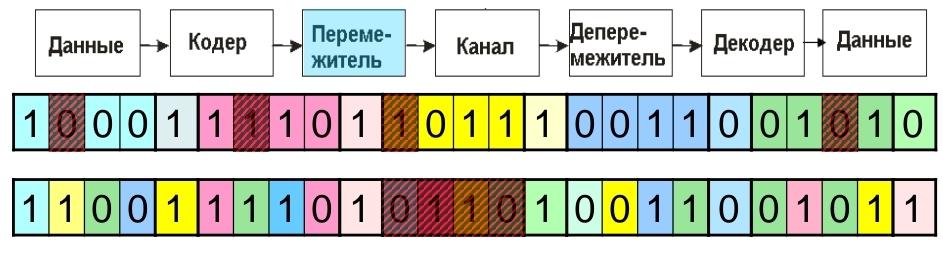
Процедура перемежения/деперемежения состоит в перестановке символов кодового слова и восстановлении исходной последовательности после передачи ее по каналу

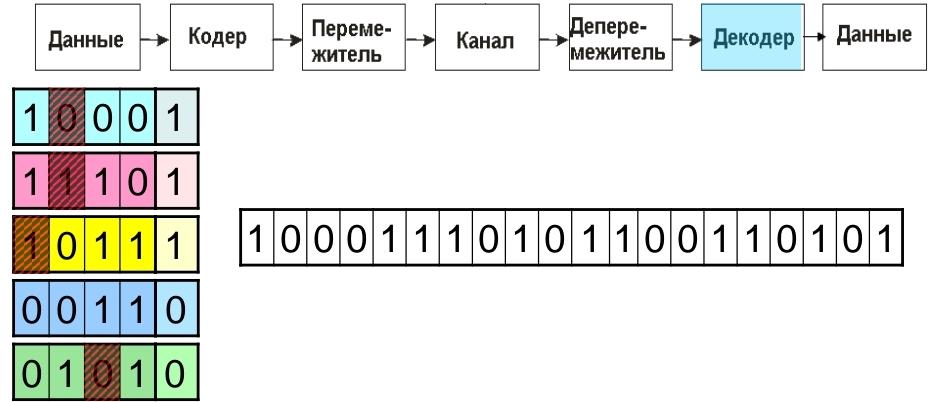
Больше глубина перемежения => больше задержка

**Длина** передаваемой последовательности **не меняется**





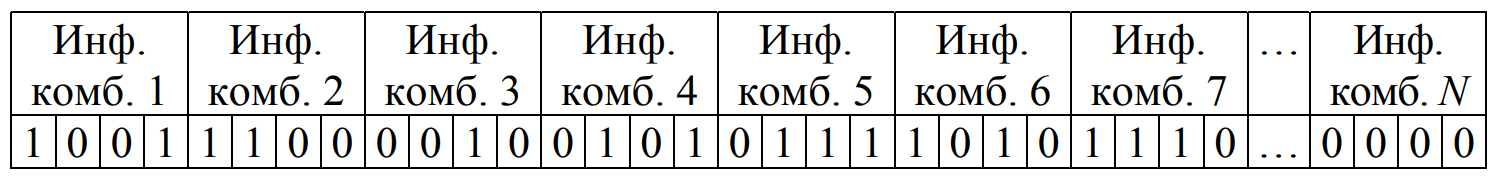






**Пример (исп-е кода Хемминга и блокового перемежителя)**

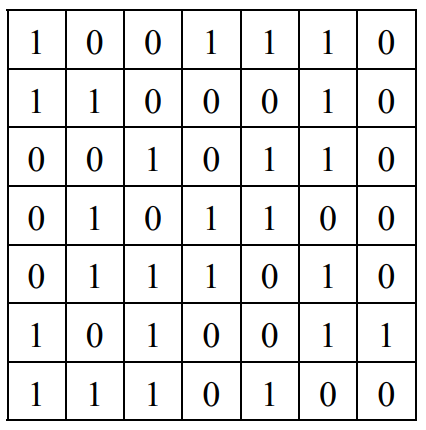
Xk = 1001 1100 0010 0101 0111 1010 1110



Для каждого 4-х битного значения необходимо вычислить избыточные символы

Строим матрицу перемежения, записывая в строки все кодовые слова Xn (присоединили уже Xr к Xk)

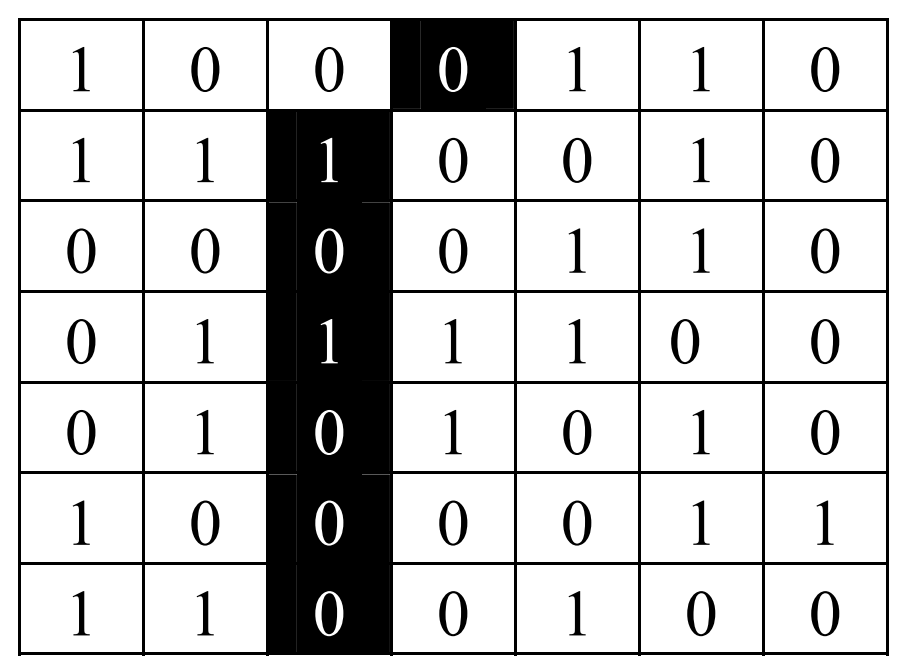
Матрица перемежения:



Символы считываются по столбцам, таким образом, Xn = 1100011 0101101 0010111 1001100 1011001 1110110 0000010

Пусть сообщение передано с ошибкой Yn = 1100011 0101101 0101000 0001100 1011001 1110110 0000010

Матрица деперемежения:



Далее сообщения считываются по строкам

Y’n = 1000110 1110010 0000110 0111100 0101010 1000011 1100100



Выделяем из кодовых слов избыточные символы и также заново высчитываем новые наборы избыточных символов, чтобы определить синдром

Благодаря синдрому определим номер столбца проверочной матрицы, в котором ошибка, а тем самым и вектор ошибки, чтобы исправить все ошибки.

## 🍏23. Совместное использование перемежителей/деперемежителей и корректирующих кодов в ИВС

Пример. Процесс передачи информации с использованием кода Хемминга и блокового перемежителя. Информационный поток на входе кодера Хемминга (7,4) имеет вид:











## 🍏24. Математические основы построения и использования циклических кодов

Относятся к классу линейных, систематических,

Сумма по модулю 2 двух разрешённых кодовых комбинаций даёт также разрешённую кодовую комбинацию

Каждый вектор (кодовое слово), получаемый из исходного кодового вектора путём циклической перестановки его символов, также является разрешённым кодовым вектором, при циклической перестановке символы кодового слова перемещаются слева направо на одну позицию:

**Пример1**. Если кодовое слово - 1101100, то разрешённой кодовой комбинацией будет и - 0110110

Принято описывать циклические коды (ЦК) при помощи порождающих полиномов G(X) степени r = n — k, где r — число проверочных символов в кодовом слове

**Пример2.** Переведём кодовое слово Хn = 101100 в полиномиальный вид:

Вi(X) = 1 \* X5 + 0 \* X4 + 1 \* X3 + 1 \* X2 + О \* X1 + О \* Х0 = X5 + X3 + X2

•Операции кодирования и декодирования ЦК сводятся к известным процедурам умножения и деления полиномов

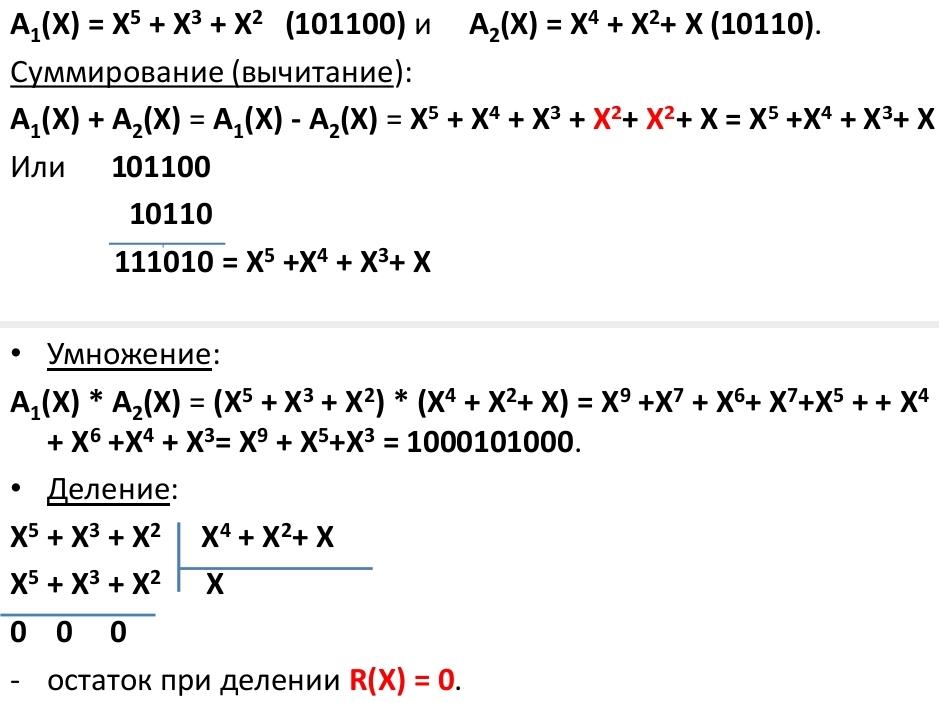
Действия с кодовыми словами в виде полиномов производятся по правилам арифметики по модулю 2 (вычитание равносильно сложению).

**Пример3.** Из равенства Хn -1 =0 получаем Хn =1. Прибавив к левой и правой частям по единице, имеем Хn + 1 = 1 + 1= 0. Таким образом, вместо двучлена Хn -1 можно ввести бином Хn +1 или 1 + Хn, из чего следует, что

Хn + Хn = Хn

(1 + 1) = 0

• Приведём далее порядок суммирования (вычитания), умножения и деления полиномов (по модулю 2). В примерах используем вышеприведённые кодовые комбинации



**При циклическом сдвиге вправо на один разряд необходимо исходную кодовую комбинацию поделить на X, а умножение на X эквивалентно сдвигу влево на один символ**

## 🍏25. Кодирование сообщения циклическим кодом

Кодовое слово состоит из двух частей: информационные Хk(Аi(Х)) и проверочные Хr(Ri(X)) символы. Число проверочных символов равно r. В младшие биты кодового слова, которые отвечают за проверочные символы, необходимо записать изначально нули, которые далее будут заменены на проверочные символы.

n - длина кодового слова

k - длина информационного слова

r - длина избыточного слова

1. На основе n, k, r необходимо выбрать порождающий полином G(X).
2. Далее необходимо привести информационное слово Xk к полиномиальному виду.
3. Выполняем вычисление проверочных символов на основе **метода деления полиномов** и их присоединяем

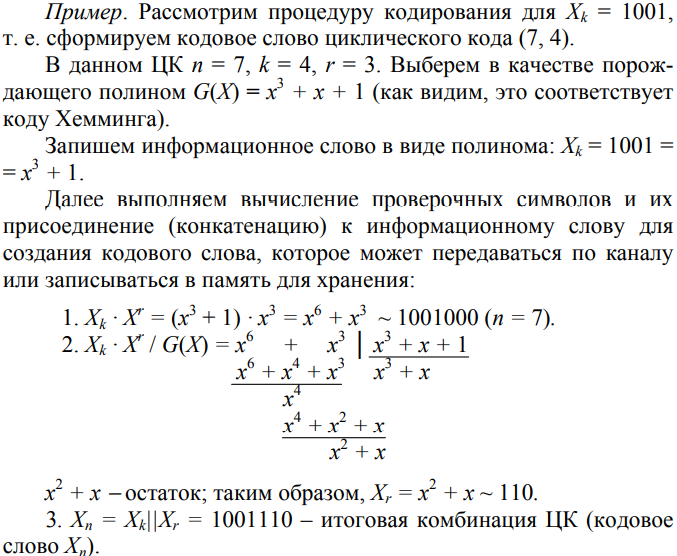
**Хk ·X^r**

1. Делим получившееся значение на порождающий полином

**Хk · X^r / G(X)**

1. Получаем итоговую комбинацию

**Xn = Хk||Хr**



## 🍏 26. Синдромный метод декодирования циклического кода

**Основная операция:** принятое кодовое слово (Yn) нужно поделить на порождающий полином, который использовался при кодировании (G(X)).

Если остаток нулевой, то Yn принадлежит коду (слово не искажено помехами).

Если ненулевой (Yn ≠ Хn) => наличие ошибки в принятой кодовой комбинации.

Для исправления ошибки нужно определить вектор (полином) ошибки Еn.

После передачи по каналу с помехами кодовое слово можно записать:

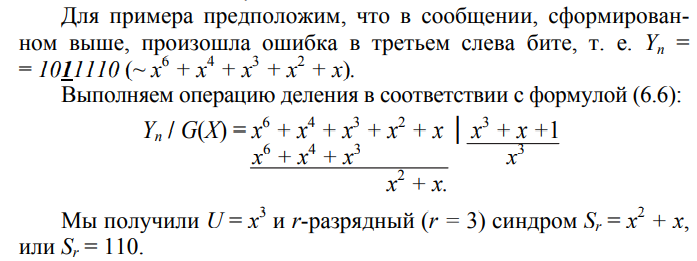
**Yn = Xn + Еn** или **Xn = Yn + Еn**

Деление принятого кодового слова на G(X)):

**Yn / G(X) = U, Sr,**

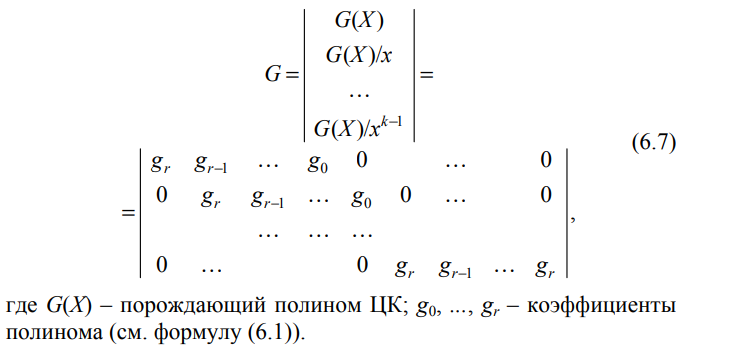
где **Sr** − остаток от деления (Yn) / (G(X)), или **синдром**.

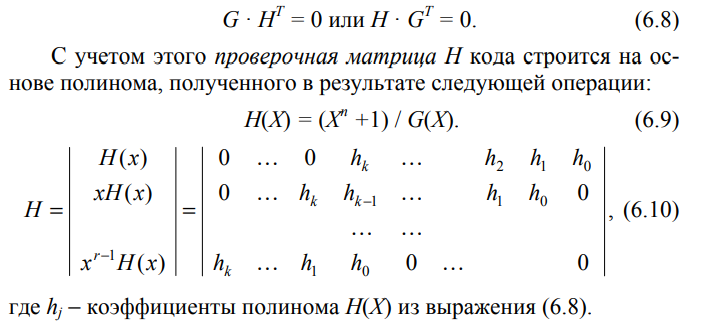
**Всякому ненулевому синдрому соответствует определённое расположение ошибок**



Цель декодирования ненулевого синдрома - определение ошибочного бита в принятом сообщении (определение вектора Е).

Порождающая матрица G циклического кода имеет в качестве строк векторы:

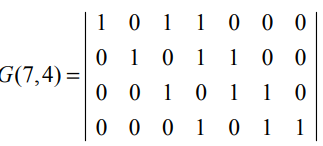




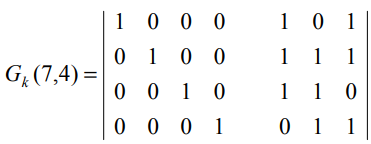
Построение порождающей матрицы:

1. первой строкой матрицы записывается порождающий полином и r нулей для резервирования места на проверочные символы
2. циклический сдвиг первой строки вправо

Обычно кол-во строк матрицы == k

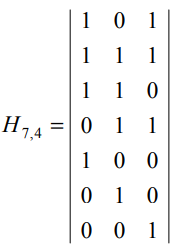


Далее необходимо преобразовать матрицу к каноническому виду.

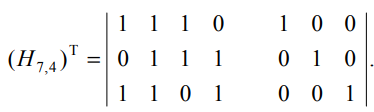


Канонический вид подразумевает, что матрица состоит из единичной подматрицы размерностью k и подматрицы Р размерностью k×r.

Проверочная матрица Н размерностью r×n может быть получена из порождающей матрицы канонического вида путем дополнения проверочной подматрицы единичной матрицей размерности r×r:



Канонический вид проверочной матрицы:



Ненулевой синдром всегда равен сумме по модулю 2 тех вектор-столбцов матрицы Н, номера которых соответствуют номерам ошибочных битов в слове Yn.

**Xn = Yn + Еn**

## 💦27. Характеристика надежности двоичного канала передачи при использовании кодов

Двоичный симметричный канал связи характеризуется тем, что вероятность:



С равной вероятностью ошибка может произойти в произвольном символе кода.

p(1|0)=p(0|1)=p — условная вероятность p(1|0)+p(0|1)=1

На практике, однако, каналы характеризуются различными вероятностями, т.е.:



**Качество канала** оценивается по вероятности полного совпадения переданного и полученного сообщения (p(x=y)=1). Возможна оценка канала по двум типам передачи:  
1 Без кодирования.  
2 C кодированием, свойства канала не изменяются.

Сравнивая вероятности без кодирования и с использованием кодирования можно оценить **эффективность использования кода**.

Пример:В реальных каналах передачи p=10^-3…10^-7. Пусть в нашем канале связи p=10^-3, k=512 bit. Оценим эффективность кода.

В соответствии с формулой для канала без использования кода p(Xk=Yk)=(1-0,001)^512.При использовании кода r=log2(512)+1=10 =>n=522. Сравнивая эти полученные рез-ты, мы получаем оценку эффективности.

Суммируя все по первому методу преобразования информации:



Всегда приводит к удлинению передаваемой последовательности n>k

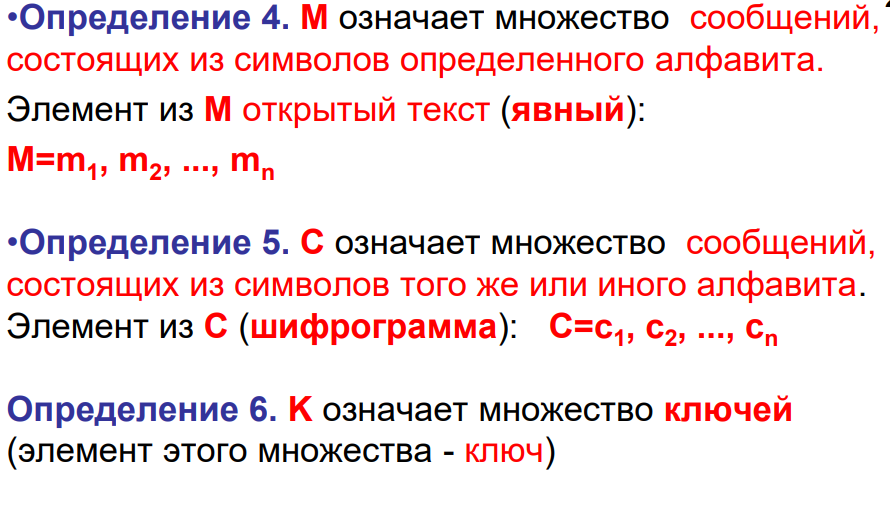
Главное назначение — повышение уровня надежности канала передачи или устройства хранения информации.

## 💦28. Базовые понятия криптографии. Основы теории больших чисел. Проблема дискретного логарифма.

**Криптография** – наука (раздел математики), о методах обеспечения конфиденциальности или безопасности информации, связанной с тайной, интегральностью данных и идентификацией.

**Криптоанализ** — наука о методах получения исходного значения зашифрованной информации без наличия секретного и не известного аналитику ключа

**Криптосистема** – это понятие, относящееся к совокупности программно-технических средств, функционирующих на основе установленных криптографических алгоритмов и осуществляющих зашифрование и расшифрование данных



**Задачи:**

1) обеспечение конфиденциальности (секретности),

2) обеспечение анонимности (неотслеживаемости),

3) обеспечение аутентификации информации и источника сообщения

**Теория чисел** – раздел математики, изучающий целые числа.

Множество ц.ч.: Z ={…, -2,-1,0,1,2,…}

Множество натуральных чисел N={1, 2, 3, ...}

| Целое число а есть **кратное** числа b, если:  a=b\*m для некоторого целого числа m; (числа ≠ 0)  b|a; пример: 9|27, т.к. 27 = 9\*3  b - делитель числа a |
| --- |

| **(алгоритм деления)** Для положительных целых чисел а и b существуют единственные целые положительные числа q и r, где 0 ≤ r < b такие что а = bq + r; а ˂ b, r – остаток, q – частное Если а ˂ b, то q=0.  а =4, b=7, тогда q =0, r=4, т.к. 4=7\*0+4; |
| --- |

| Положит целое число d называется **общим делителем** чисел а и b, если d|а и d|b.  Положит ц.ч. d называется **наибольшим общим делителем** чисел а и b, НОД(а,b), если d|а и d|b, и если из с|а и с|b следует с|d  Если НОД(а, b) = 1, то числа а, b называются **взаимно простыми** |
| --- |

| **Теорема:** Если а = bq + с, то НОД(а, b) = НОД(b, с);  *Пример:* 16=6\*2+4; а = 16, b=6, q =2, с=4. НОД(а, b) = НОД(16, 6) = 2; НОД(b, с)= НОД(6, 4)=2 |
| --- |

| **Основная теорема арифметики.**  Всякое натуральное число N, кроме 1, можно представить как произведение простых сомножителей: N = p1 \* p2 \* p3 \* ... \* pn , n > 1 .  Натуральное число p называется **простым**, если p > 1 и не имеет положительных делителей, отличных от 1 и p  **Взаимно простые числа** а, b не имеют общих множителей, кроме 1, НОД (а, b) =1  ***Проблема***. При разрядности 1024 и более бит нахождение пары взаимно простых чисел, удовлетворяющих пределенному условию, может занять сотни лет |
| --- |

|  |
| --- |

## 🍏29. Решето Эратосфена.

**Задача**: поиск простых чисел, не превышающих n.

Суть в последовательном исключении из списка целых чисел от 1 до n чисел (ил и из сокращенного диапазона, например, от m до n, 1<m≤n), кратных 2, 3, 5 и другим простым числам, уже найденным «решетом».

***Алгоритм:***

| 1. Выписать подряд все целые числа от 2 (либо от m) до n (2, 3, 4, …, n). Пусть некоторая переменная (положим s) изначально равна 2 – первому простому числу.  2. Удалить из списка числа от 2s до n, считая шагами по s (это будут числа кратные s: 2s, 3s, 4s, …).  3. Найти первое из оставшихся чисел в списке, большее чем s, и присвоить значению переменной s это число.   * Повторять шаги 2 и 3, пока возможно. |
| --- |

## 🍏30. Основная теорема арифметики. Алгоритм Евклида нахождения НОД

| **Основная теорема арифметики.**  Всякое натуральное число N, кроме 1, можно представить как произведение простых сомножителей: N = p1 \* p2 \* p3 \* ... \* pn , n > 1 .  Натуральное число p называется **простым**, если p > 1 и не имеет положительных делителей, отличных от 1 и p  **Взаимно простые числа** а, b не имеют общих множителей, кроме 1, НОД (а, b) =1  ***Проблема***. При разрядности 1024 и более бит нахождение пары взаимно простых чисел, удовлетворяющих пределенному условию, может занять сотни лет |
| --- |

**Алгоритм Евклида**

**Задача:** Поиск НОД двух чисел a и b (a > b).

|  |
| --- |

## 🍏 31. Основы модулярной арифметики. Вычеты.

***Модулярная арифметика*** - К. Ф. Гаусс. В этой арифметике мы интересуемся остатком от деления числа а на число n (n – натуральное число и n > 1).

Если таким остатком является число b, то можно записать:

| **a ≡ b (mod n),**  или  **a ≡ b mod n.**  *a сравнимо с b по модулю n* |
| --- |

## 

| **a ≡ b mod n, если a=b+kn при целом k** |
| --- |

## 

| b называют **вычетом по модулю n**;  a mod n обозначает **вычет** от а;  Множество целых чисел от 0 до n-1 образует полную **систему вычетов** по модулю n |
| --- |

## 

| * ***Коммутативна,*** * ***Ассоциативна*** * ***Дистрибутивна*** |
| --- |

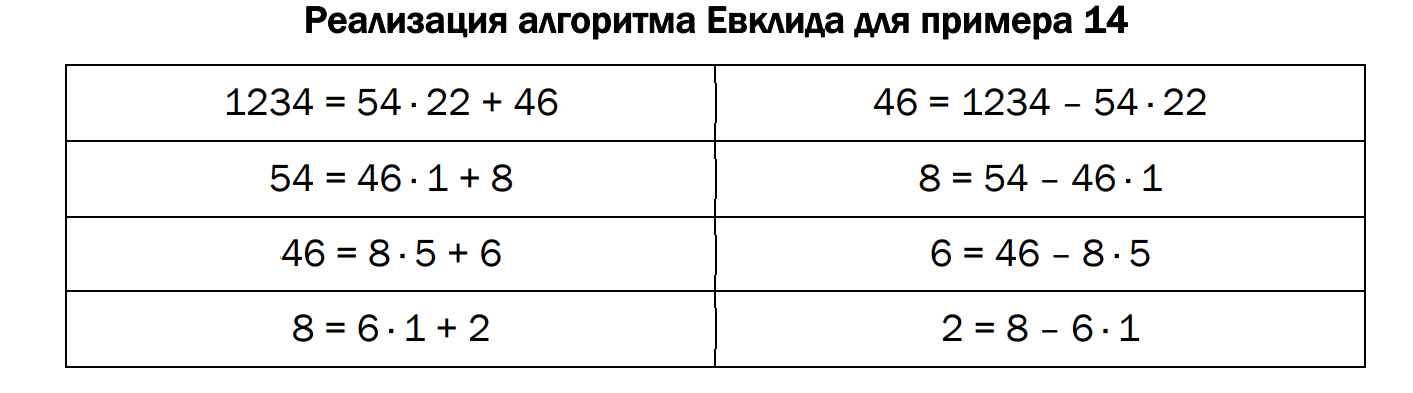
## 

## 💦32. Обратные вычисления по модулю в криптографии.

| В модулярной арифметике запись уравнения в виде  **aх ≡ 1 mod n**  Предусматривает поиск таких значений х и k, которые удовлетворяют равенству **aх = nk + 1**  Здесь x - **обратное** к а по модулю n  Может быть сформулирована следующим образом: найти такое х, что  **1 ≡ ах mod n** |
| --- |

Поиск обратного числа по модулю - **расширенный алгоритм Евклида.** Состоит из двух этапов: спуск и восхождение.

| Если НОД (a, b) = d, то справедливо следующее соотношение (**соотношение Безу**):  аu + bv = d.  Следовательно, если НОД = 1, то  аu + bv = 1 |
| --- |



| **Малая теорема Ферма**  Если n – простое число и а не кратно n, то справедливо  **a(n-1) ≡1 mod n**  или  **an ≡ a ( mod n )**  **Обощение Эйлера над Малой теоремой Ферма:**  Если НОД (а, n) =1, то  **аφ(n) mod n = 1**  (т.к. функция эйлера от простого числа n равняется n-1)  Зачем?  Можем найти обратное по модулю, если разделить обе части уравнения на a:  **а-1= аφ(n)-1 mod n**  Особенно удобно, если у нас n простое, степень простого числа или произведение простых (см 33 для этих случае) |
| --- |

## 💦33. (?)Функция Эйлера в криптографии.

***Функция Эйлера (φ(n))***- Количество натуральных чисел, не превосходящих n и взаимно простых с n

| pi - разные простые множители числа n в некоторой степени, тогда:    ***Пример:***  Определить количество натуральных чисел, не превосходящих 12 и взаимно простых с 12.  Взаимно простыми с 12 будут четыре числа 1, 5, 7, 11, т. е. φ(12) = 4 – получено методом «ручного» подсчета.  Каноническое разложение числа 12: 12 = 2 · 2 · 3 = = 22 · 3, т. е. p1 = 2, p2 = 3.  Теперь подсчитаем функцию Эйлера: φ(12) = 12 · (1 – 1/2) · (1 – 1/3) = 4. |
| --- |

| Если p – **простое** число, то:  φ(p) = p – 1,  если числа **p и q являются простыми и p ≠ q**, то:  φ(p) = (p – 1)(q – 1).  если **p простое, а n - целое число, то:**    Если p и q – очень большие простые числа и известен результат их перемножения (число n), то обратная задача – найти p и q по известному n **(задача факторизации)** даже для современных вычислительных средств представляется практически неразрешимой.  Эта особенность используется в некоторых алгоритмах асимметричной криптографии. |
| --- |

**Применение: RSA**

## 🍏34. Хеш-функция и ее свойства.

| **Первообразный корень** **по модулю n** является таким числом, что его степени дают все возможные по модулю n вычеты, которые взаимно просты с n. |
| --- |

**Однонаправленная функция** предполагает простоту ее вычисления (вычисления f(x) по известному аргументу х) и сложность обратного вычисления (вычисления х по известному f(x) )

**Хэш функция** – функция, которая принимает на входе строку символов переменной (произвольной) длины и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины, называемой **значением хеш-функции** или ее **сверткой**

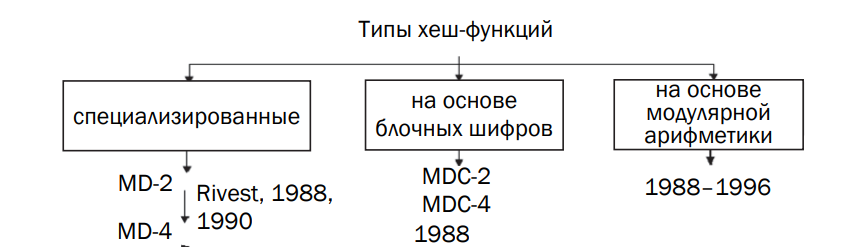
Однонаправленная ХФ – основа многих протоколов

**Задачи хэш-функций:**

* аутентификация (хранение паролей);
* проверка целостности данных;
* защита файлов;
* обнаружение зловредного ПО;
* криптовалютные технологии.

***Свойства хэш-функции***

* **Детерминированность**: h = H(M), значение для одного M всегда одинаковое.
* Зная М, легко вычислить h
* Зная h, трудно определить М - **односторонность**
* Зная М, трудно определить М’ (М≠ М’ ), для которого H(М)= H(М’) – **коллизия 1-го рода**
* Трудно найти два случайных сообщения (М и М’), для которых H(М)= H(М’) - **коллизия 2-го рода**

****

Для хеш-функций универсальным методом поиска коллизий является метод, основанный на известной статистической задаче – «парадоксе дня рождения».

**Парадокс дней рождения** - это кажущееся парадоксальным утверждение, что вероятность совпадения дней рождения (даты) хотя бы у двух членов группы из 23 и более человек, превышает 0,5.

## 💦35. Общая классификация криптографических методов защиты информации.

1. На основе процедуры шифрования
   1. подстановочные - Цезарь, Виженера
   2. перестановочные
2. На основе генерирования и использования ключа
   1. блочные - DES
   2. поточные (потоковые) - RC4
3. На основе типа ключа
   1. симметричные (с тайным ключом): e=d - DES
   2. асимметричные (с открытым или публичным ключом): e!=d - RSA, El Gamal

## 🍏36. Подстановочные шифры. Шифр Цезаря. Криптостойкость.

**Подстановочный шифр** - замена символов открытого текста соответственно символами того же или иного алфавита - подстановка.

* **моноалфавитные** (шифры однозначной замены или простые подстановочные) - Цезарь;
* **полиграммные** - замена группы символов - Плейфер;
* **омофонические** (однозвучные шифры или шифры многозначной замены);
* **полиалфавитные** (несколько моноалфавитных(?))- Виженер

Шифр Цезаря:

* Простой (сдвиг на k)
* Аффинный
  + y ≡ ax + b mod N,
  + x ≡ a–1 (y + N – b) mod N
* С лозунгом (ключевым словом)

**Криптостойкость**: уязвим перед частотным анализом(?)

## 🍏37. Шифр на основе аффинных преобразований. Криптостойкость.

Моноалфавитный подстановочный

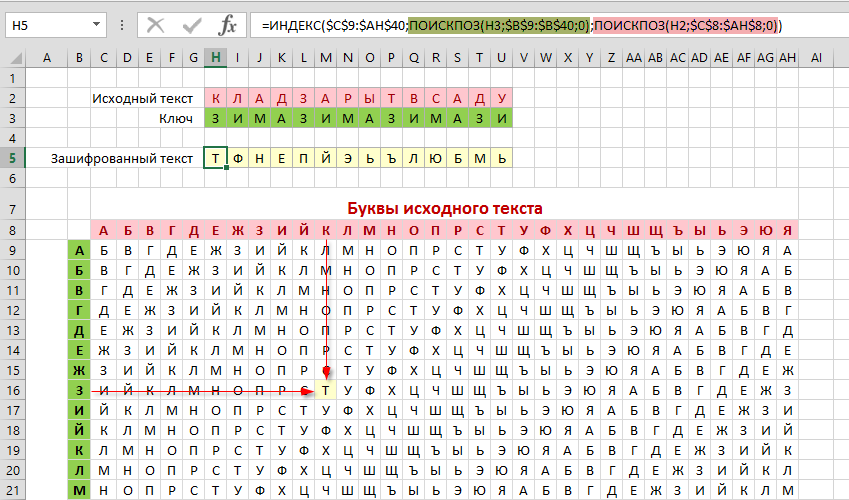
* Аффинный
  + y ≡ ax + b mod N,
  + x ≡ a–1 (y + N – b) mod N

**Криптостойкость**: уязвим перед частотным анализом и полным перебором

(**Полный перебор** - В случае шифрования сообщений на русском языке N=33, существует 297 нетривиальных аффинных шифров, не учитывая 33 тривиальных шифра Цезаря (a = 1)

## 🍏38. Шифр Виженера. Криптостойкость.

Полиалфавитный подстановочный



**Криптостойкость:**

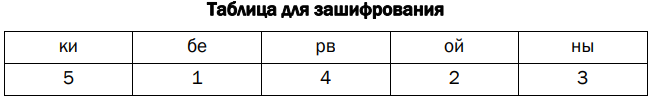
Шифр Виженера «размывает» характеристики частотностей появления символов в тексте, но некоторые особенности появления символов в тексте остаются. Главный недостаток шифра Виженера состоит в том, что его ключ повторяется. Поэтому простой криптоанализ шифра может быть построен в два этапа:

* Поиск длины ключа. Можно анализировать распределение частотностей в зашифрованном тексте с различным прореживанием. То есть брать текст, включающий каждую 2-ю букву зашифрованного текста, потом каждую 3-ю и т. д. Как только распределение частотностей букв будет сильно отличаться от равномерного (например, по энтропии), то можно говорить о найденной длине ключа.
* Криптоанализ. Совокупность l шифров Цезаря (где l — найденная длина ключа), которые по отдельности легко взламываются.

## 🍏39. Перестановочные шифры. Криптостойкость.

Сущность **перестановочного шифрования** состоит в том, что исходный текст (М) и зашифрованный текст (С) основаны на использовании **одного и того же алфавита**, а тайной или ключевой информацией является алгоритм перестановки.

* Одиночная перестановка
  + Простая - без ключа - прим: в обратном порядке
  + Блоками из нескольких символов:



(сортировкой по алфавиту)

* + Маршрутная - прим: Скитала, зигзаг, спираль и т.д.
    - Вертикальная перестановка - ключ сортируется по алфавиту, столбцы меняются местами:



* Множественная перестановка
  + Пример: Вертикальная перестановка, но ключа 2 и перестановки 2  
    

**Криптостойкость**: Уязвимы перед частотным анализом. Одной из наиболее устойчивых характеристик осмысленного текста является отсутствие запретных биграмм (пара соседних букв). Например, биграммы «Ъ + Ь», «гласная + Ь», «пробел + Ь».

## 🍏40. Шифровальная машина Энигма. Устройство, функционал, криптостойкость.

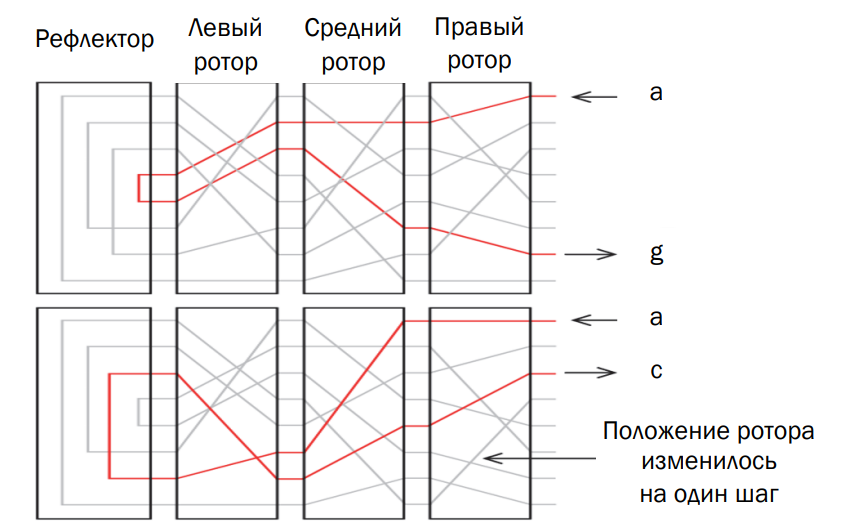
«Энигма» строится на основе подстановочных шифров, подобных шифру Цезаря, в котором, как известно, ключ сообщения, который должен знать получатель, – это просто смещение между двумя алфавитами. Принято считать, что в основе шифра «Энигмы» лежит динамический шифр Цезаря.

Машина «Энигма» – это электромеханическое устройство. Как и другие роторные машины, «Энигма» состоит из комбинации механических и электрических подсистем.

Механическая часть включает в себя клавиатуру, набор вращающихся дисков – роторов, которые расположены вдоль вала и прилегают к нему, и ступенчатого механизма, движущего один или несколько роторов при каждом нажатии на клавишу.

Электрическая часть, в свою очередь, состоит из электрической схемы, соединяющей между собой клавиатуру, коммутационную панель, лампочки и роторы (для соединения роторов использовались скользящие контакты).





Вся схема - электрический контур, который замыкается при нажатии клавиши, в результате зажигается одна из ламп.

Правый ротор сдвигается постоянно, остальные через некоторое количество операций. Таким образом один символ кодируется различным образом на каждом новом шаге.

Всего создано 8 роторов и 4 рефлектора (отличаются отображением алфавитов).

Существовали модели с 4 одновременными роторами, но они сильно утяжеляли машину, а само использование доп. ротора было избыточным.

**Ключ**: Стартовая позиция роторов.

**Шифрование В WW2:**

* Стартовая позиция роторов - код дня
* Случайный ключ сообщения (длина = количество роторов, чаще всего 3)
* Зашифровать ключ сообщения
* Незашифрованный ключ сообщения - новая начальная позиция роторов
* Ввод сообщения

*Отправляется*:

Код дня, зашифрованный ключ, зашифрованное сообщение

**Расшифрование**:

* Начальная позиция - код дня
* Расшифровать ключ
* Использовать расшифрованный ключ как начальную позицию

**Криптостойкость:**

Факторы:

* выбор и порядок роторов;
* разводку (коммутацию) роторов;
* настройку колец на каждом из роторов;
* начальное положение роторов в начале сообщения;
* отражатель;
* настройки коммутационной панели.

Но часть из этих факторов (такие менялись значительно реже и были известны заранее.

**Общее число комбинаций**:

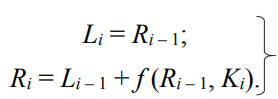
n = 1,07 \* 10^23

## 🍏41. Сеть Фейстеля. Понятие «лавинного эффекта» и его использование в криптографии

Само название конструкции Фейстеля (сети) означает её ячеистую топологию. Формально одна ячейка сети соответствует одному раунду зашифрования или расшифрования сообщения.

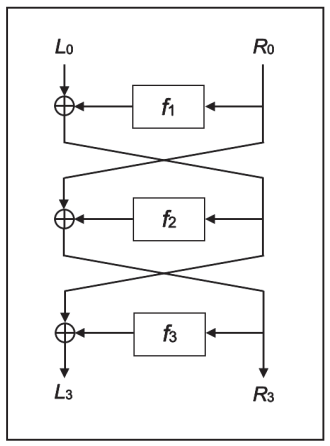
При зашифровании сообщение разбивается на блоки одинаковой длины (обычно 64 или 128 бит). Полученные блоки называются входными. В случае если длина входного блока меньше, чем выбранный размер, блок удлиняется установленным способом.

Каждый входной блок шифруемого сообщения изначально делится на два подблока одинакового размера: левый (L0) и правый (R0). Далее в каждом i-м раунде выполняется преобразования в соответствии с формальным представлением ячейки сети Фейстеля:



По какому-либо математическому правилу вычисляется раундовый ключ Ki. В приведённом примере знак “+” соответсвует поразрядному суммированию на основе “XOR”.

Графически сеть Фейстеля можно представить так:



Расшифрование происходит так же, как зашифрование, с той лишь разницей, что раундовые ключи используются в обратном порядке по отношению к зашифрованию.

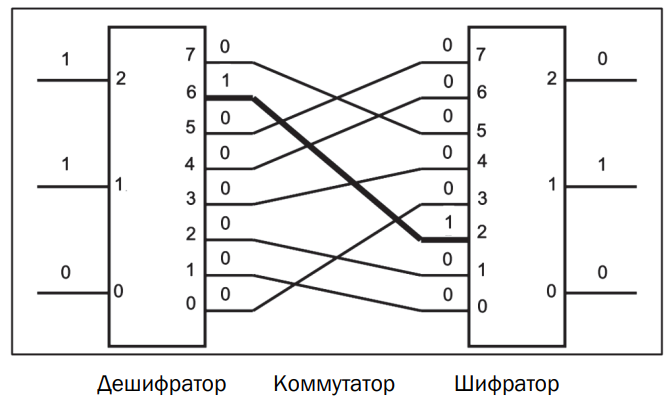
Фейстель описывает два блока преобразований с использованием функции f(Ri-1, Ki):

* блок подстановок (S-блок);
* блок перестановок (P-блок).

Блок подстановок состоит:

* из дешифратора, преобразующего n-разрядное двоичное число в одноразрядный сигна по основанию 2n
* внутреннего коммутаторы
* шифратора, преобразующего сигнал из одноразрядного 2n-ричного в n-разрядный двоичный

Пример реализации 3-разрядного S-блока:



Блок перестановок изменяет положение цифр, т. е. является линейным устройством. Этот блок может иметь очень большое количество входов-выходов, но в силу линейности является слабым местов преобразования с точки зрения криптостойкости.

Чаще всего значение “блок” применяется для обозначения “функций”.

В основе криптостойкости блочных шифров лежит идея К. Шеннона о представлении составного шифра таким образом, чтобы он обладал двумя важными свойствами: рассеиванием и перемешиванием. Рассеивание должно скрыть отношение между зашифрованным текстом и исходным текстом.

Рассеивание подразумевает, что каждый символ (или бит) в зашифрованном тексте зависит от одного или всех символово в исходном текссте. Другими словами, если единственный символ в исходном тексте изменён, несколько или все символы в зашифрованном тексте будут также изменены.

Идея относительно перемешивания заключается в том, что оно должно скрыть отношения между зашифрованным текстом и ключом.

## 🍏42. Методы симметричного криптопреобразования. Стандарт DES. Общая характеристика. Криптостойкость алгоритма

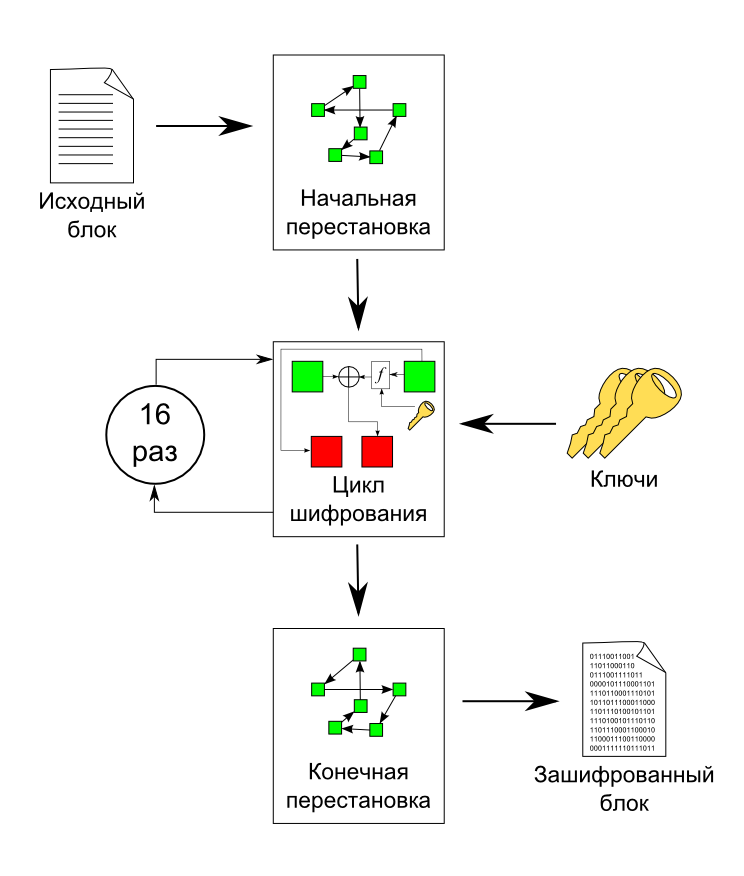
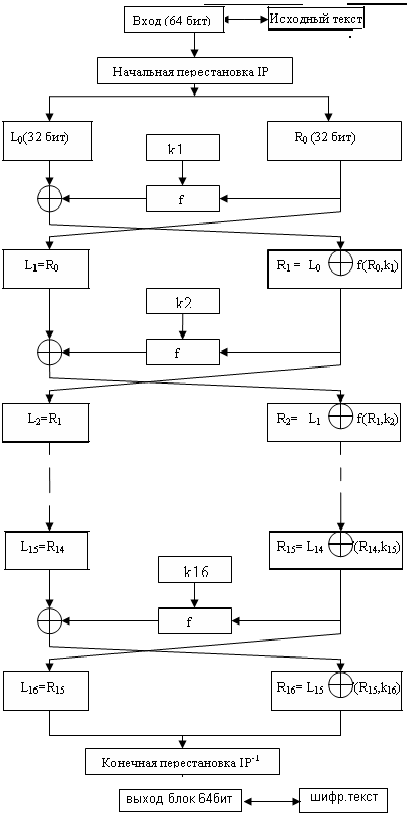


Схема шифрования алгоритма DES



Исходный текст — блок 64 бит.

Процесс шифрования состоит из начальной перестановки, 16 циклов шифрования и конечной перестановки.

Начальная перестановка

Исходный текст T (блок 64 бит) преобразуется c помощью начальной перестановки IP которая определяется таблицей 1:



По таблице первые 3 бита результирующего блока IP(T) после начальной перестановки IP являются битами 58, 50, 42 входного блока T, а его 3 последние бита являются битами 23, 15, 7 входного блока.

Циклы шифрования

Полученный после начальной перестановки 64-битовый блок IP(T) участвует в 16 циклах преобразования Фейстеля.

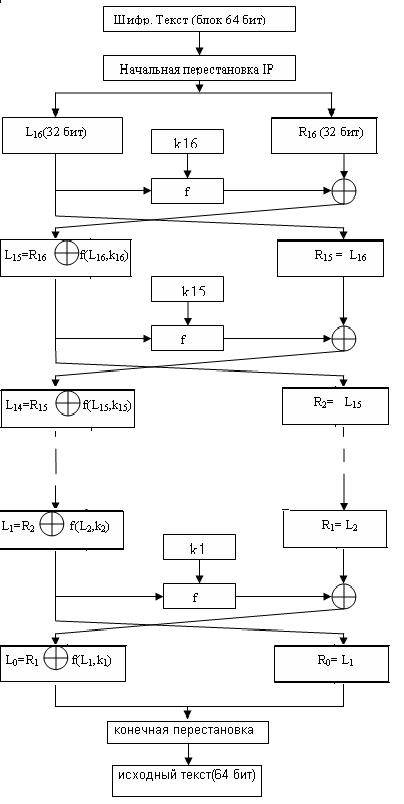
— 16 циклов преобразования Фейстеля:



Сеть Фейстеля см. вопрос выше

Генерирование ключей ki

Ключи ki получаются из начального ключа k (56 бит = 7 байтов или 7 символов в ASCII) следующим образом. Добавляются биты в позиции 8, 16, 24, 32, 40, 48, 56, 64 ключа k таким образом, чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64). Такая перестановка определена в таблице 5.





Эта перестановка определяется двумя блоками C0 и D0 по 28 бит каждый. Первые 3 бита C0 есть биты 57, 49, 41 расширенного ключа. А первые три бита D0 есть биты 63, 55, 47 расширенного ключа. Ci, Di i=1,2,3…получаются из Ci-1,Di-1 одним или двумя левыми циклическими сдвигами согласно таблице 6.



Ключ ki, i=1,…16 состоит из 48 бит, выбранных из битов вектора CiDi (56 бит) согласно таблице 7. Первый и второй биты ki есть биты 14, 17 вектора CiDi



Конечная перестановка

Конечная [перестановка](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0) IP-1 действует на T-1 (где T16-1 =R16+L16) и является обратной к первоначальной перестановке. Конечная перестановка определяется таблицей 8.



Схема расшифрования

При расшифровании данных все действия выполняются в обратном порядке. В 16 циклах расшифрования, в отличие от шифрования c помощью прямого преобразования сетью Фейстеля, здесь используется обратное преобразование сетью Фейстеля.



Ключ ki, i=16,…,1, функция f, перестановка IP и IP-1 такие же, как и в процессе шифрования. Алгоритм генерации ключей зависит только от ключа пользователя, поэтому при расшифровании они идентичны.

Криптостойкость алгоритма DES

Нелинейность преобразований в DES средствами только S-блоков и использование слабых S-блоков позволяет осуществлять контроль над шифрованной перепиской. Выбор S-блоков требует соблюдения нескольких условий:

* Каждая строка каждого блока должна быть перестановкой множества {0, 1, 2, …, 15}
* S-блоки не должны являться линейной или афинной функцией своих аргументов.
* Изменение одного бита на входе S-блока должно приводить к изменению, по крайней мере, двух битов на выходе.
* Для каждого S-блока и любого аргумента х значение S(x) и S(x ⊕ 0011002) должны различаться, по крайней мере, двумя битами.

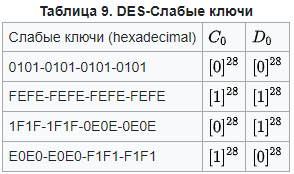
Из-за небольшого числа возможных ключей (всего 256), появляется возможность их полного перебора на быстродействующей вычислительной технике за реальное время. В [1998 году](https://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) [Electronic Frontier Foundation](https://ru.wikipedia.org/wiki/Electronic_Frontier_Foundation), используя специальный компьютер DES-Cracker, удалось взломать DES за 3 дня.

Слабые ключи

Слабыми ключами называется ключи k такие, что

DESk(DESk(x))=x, где x — 64-битный блок.

Известны 4 слабых ключа, они приведены в таблице 9. Для каждого слабого ключа существует 232 [неподвижные точки](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D0%BE%D0%B4%D0%B2%D0%B8%D0%B6%D0%BD%D0%B0%D1%8F_%D1%82%D0%BE%D1%87%D0%BA%D0%B0), то есть таких 64-битных блоков х, для которых DESk(x)=x.



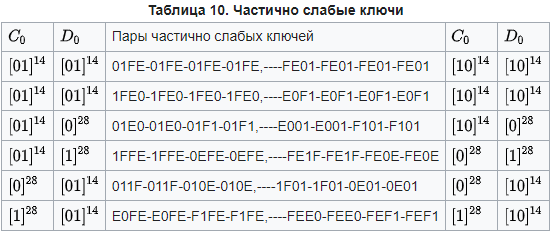
[0]28 обозначает вектор, состоящий из 28 нулевых битов.

Частично слабые

В алгоритме DES существуют слабые и частично слабые ключи. Частично слабые ключи — это такие пары ключей

(k1,k2), что DESk(DESk(x))=x.

Существуют 6 частично слабых пар ключей, они приведены в таблице 10. Для каждого из 12 частично слабых ключей существуют 232 «антинеподвижные точки», то есть такие блоки х, что DESk(x)= 



Известные атаки на DES



* Метод [полного перебора](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80) требует одну известную пару шифрованного и расшифрованного текста, незначительный объём памяти, и его выполнение требует около 255 шагов.
* [Дифференциальный криптоанализ](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) — первую такую атаку на DES заявили [Бихам](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%85%D0%B0%D0%BC,_%D0%AD%D0%BB%D0%B8) и [Шамир](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%BC%D0%B8%D1%80,_%D0%90%D0%B4%D0%B8). Эта атака требует шифрования 247 открытых текстов, выбранных нападающим, и для её выполнения нужны примерно 247 шагов. Теоретически являясь точкой разрыва, эта атака непрактична из-за чрезмерных требований к подбору данных и сложности организации атаки по выбранному открытому тексту. Сами авторы этой атаки Biham и Shamir заявили, что считают DES защищённым для такой атаки.
* [Линейный криптоанализ](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) разработан Matsui. Этот метод позволяет восстановить ключ DES с помощью анализа 243 известных открытых текстов, при этом требуется примерно 243 шагов для выполнения. Первый экспериментальный криптоанализ DES, основанный на открытии Matsui, был успешно выполнен в течение 50 дней на автоматизированных рабочих местах 12 HP 9735.

Для линейного и дифференциального криптоанализа требуется достаточно большой объём памяти для сохранения выбранных (известных) открытых текстов до начала атаки.

## 🍏43. Методы симметричного криптопреобразования. Стандарт DES. Структура одного цикла. Криптостойкость алгоритма

См. выше

## 🍏44. Методы симметричного криптопреобразования. Стандарты 3DES. Реализация и криптостойкость

Чтобы увеличивать [криптостойкость](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%BE%D1%81%D1%82%D1%8C) DES, появляются несколько вариантов: [double DES](https://ru.wikipedia.org/w/index.php?title=Double_DES&action=edit&redlink=1) ([2DES](https://ru.wikipedia.org/w/index.php?title=2DES&action=edit&redlink=1)), [triple DES](https://ru.wikipedia.org/wiki/Triple_DES) ([3DES](https://ru.wikipedia.org/wiki/3DES)), [DESX](https://ru.wikipedia.org/wiki/DESX), [G-DES](https://ru.wikipedia.org/w/index.php?title=G-DES&action=edit&redlink=1).

* Методы [2DES](https://ru.wikipedia.org/w/index.php?title=2DES&action=edit&redlink=1) и [3DES](https://ru.wikipedia.org/wiki/3DES) основаны на DES, но увеличивают длину ключей (2DES — 112 бит, 3DES — 168 бит) и поэтому увеличивается [криптостойкость](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%BE%D1%81%D1%82%D1%8C).
* Схема 3DES имеет вид DES(k3,DES(k2,DES(k1,M))), где k1,k2,k3 ключи для каждого шифра DES. Это вариант известен как в ЕЕЕ, так как три DES операции являются шифрованием. Существует 3 типа алгоритма 3DES:
* [DES-EEE3](https://ru.wikipedia.org/w/index.php?title=DES-EEE3&action=edit&redlink=1): Шифруется три раза с 3 разными ключами.
* [DES-EDE3](https://ru.wikipedia.org/w/index.php?title=DES-EDE3&action=edit&redlink=1): 3DES операции шифровка-расшифровка-шифровка с 3 разными ключами.
* [DES-EEE2](https://ru.wikipedia.org/w/index.php?title=DES-EEE2&action=edit&redlink=1) и [DES-EDE2](https://ru.wikipedia.org/w/index.php?title=DES-EDE2&action=edit&redlink=1): Как и предыдущие, за исключением того, что первая и третья операции используют одинаковый ключ.

Самый популярный тип при использовании 3DES — это DES-EDE3, для него алгоритм выглядит так:

[Зашифрование](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5):

C=Ek3(Ek2-1(Ek1(P))).

[Расшифрование](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5):

P=Ek1-1(Ek2(Ek3-1(C)))

При выполнении алгоритма 3DES ключи могут выбрать так:

* k1,k2,k3 независимы.
* k1,k2 независимы, а k1=k3
* k1=k2=k3.
* Метод [DESX](https://ru.wikipedia.org/wiki/DESX) создан [Рональдом Ривестом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D0%B2%D0%B5%D1%81%D1%82,_%D0%A0%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%B4_%D0%9B%D0%B8%D0%BD%D0%BD) и формально продемонстрирована Killian и Rogaway. Этот метод — усиленный вариант DES, поддерживаемый инструментарием [RSA](https://ru.wikipedia.org/wiki/RSA) Security. [DESX](https://ru.wikipedia.org/wiki/DESX) отличается от DES тем, что каждый бит входного открытого текста DESX логически суммируется по модулю 2 с 64 битами дополнительного ключа, а затем шифруется по алгоритму DES. Каждый бит результата также логически суммируется по модулю 2 с другими 64 битами ключа. Главной причиной использования DESX является простой в вычислительном смысле способ значительно повысить [стойкость](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%BE%D1%81%D1%82%D1%8C) DES к атакам [полного перебора](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80) ключа.
* Метод [G-DES](https://ru.wikipedia.org/w/index.php?title=G-DES&action=edit&redlink=1) разработан Schaumuller-Bichl для повышения производительности DES на основе увеличения размеров шифрованного блока. Заявлялось, что G-DES защищён так же, как и DES. Однако Biham и Shamir показали, что [G-DES](https://ru.wikipedia.org/w/index.php?title=G-DES&action=edit&redlink=1) с рекомендуемыми параметрами легко взламывается, а при любых изменениях параметров шифр становится ещё менее защищён, чем DES.
* Ещё другой вариант DES использует независимые суб-ключи. Различно от алгоритма DES, в котором на основе 56-битного секретного ключа пользователя алгоритм DES получает шестнадцать 48-битных суб-ключей для использования в каждом из 16 раундов, в этом варианте использует 768-битный ключ (разделённый на 16 48-битных подключей) вместо 16 зависимых 48-битных ключей, создаваемых по ключевому графику алгоритма DES. Хотя очевидно, что использование независимых суб-ключей значительно усложнит полный поиск ключа, но стойкость к атаке дифференциальным или линейным криптоанализом ненамного превысит стойкость обычного DES. По оценке Biham для дифференциального криптоанализа DES с независимыми подключами требуется 261 выбранных открытых текстов, в то время как для линейного криптоанализа требуется 260 известных открытых текстов.

## 💦45. Сравнительная характеристика алгоритмов Lucifer, IDEA, ГОСТ 28147-89, Blowfish



IDEA (International Data Encryption Algorithm) — алгоритм блочного симметричного шифрования, который был предложен на замену стандарта DES. Начальная версия алгоритма IDEA появилась в 1990 году. Алгоритм запатентован в США и в большинстве европейских стран. Владеет патентом Ascom Tech, но в некоммерческих целях алгоритм можно использовать бесплатно.

Размер блока в этом шифре — 64 бита, длина ключа — 128. Стоит сразу сказать, что алгоритм IDEA — самый молодой из перечисленных и его математика очень сложна.

Конфузия — шифрование должно зависеть от ключа сложным и запутанным способом. Диффузия — каждый бит незашифрованного текста должен влиять на каждый бит зашифрованного текста. Распространение одного незашифрованного бита на большое количество зашифрованных битов скрывает статистическую структуру незашифрованного текста.

В IDEA эти свойства достигаются за счет применения независимых математических операций. В отличие от DES, главной операцией которого является XOR (сложение по модулю 2), IDEA предусматривает наличие:

* XOR;
* сложения по модулю 216;
* умножения по модулю (216 + 1).

Комбинирование этих трех операций обеспечивает комплексное преобразование входных данных, затрудняя криптоанализ IDEA по сравнению с DES.

В шифре IDEA выполняется восемь раундов, и в каждом раунде блок открытого текста подвергается преобразованию посредством математических операций. Любители «зреть в корень» могут позреть на схему одного раунда шифра IDEA, приведенную ниже. Блок текста, длиной 64 бита, делится на подблоки по 16 бит. Каждый такой полученный блочок поступает на вход раунда и подвергается сложному преобразованию.



IDEA отличается высокой стойкостью благодаря своим многочисленным математическим операциям. Кроме того, к достоинствам данного алгоритма относится высокая скорость зашифрования — почти в два раза выше, чем у алгоритма DES (в зависимости от платформы, на которой выполняется шифрование). Однако скорость расшифровки снижается из-за тяжелых операций вычисления, обратных умножению по модулю (216 + 1).

Алгоритм IDEA не стал международным стандартом шифрования, как того желали его авторы. Однако его можно считать одним из наиболее распространенных в мире алгоритмов шифрования. IDEA используется во множестве приложений, в том числе в широко известном и распространенном протоколе защиты данных PGP.

**Blowfish**

Как говорит сам Шнайер, этот алгоритм был разработан для реализации на больших микропроцессорах. Поэтому он компактен (всего 5 Кбайт памяти) и прост (использует простые математические операции — сложение, XOR и выборку из таблицы). Также алгоритм позволяет настраивать длину ключа (до 448 бит).

На 32-битных процессорах Blowfish выполняет шифрование значительно быстрее, нежели DES, однако на интеллектуальных платах в силу своей упрощенности он не особо применим. В основе Blowfish сеть Фейстеля из 16 раундов,

Алгоритм реализован в некоторых программных продуктах (FolderBolt, Nautilus, PGPfone), однако сейчас он уже теряет свою актуальность.

**ГОСТ 28147–89**

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

Шифр ГОСТ 28147–89 построен по тем же принципам, что и американский DES, однако по сравнению с DES первый более удобен для программной реализации. В ГОСТ 28147–89 применяется более длинный ключ – 256 битов, здесь используются 32 раунда шифрования.

Таким образом, основные параметры алгоритма криптографического преобразования данных ГОСТ 28147–89: размер блока составляет 64 бита, размер ключа – 256 битов, количество раундов – 32.

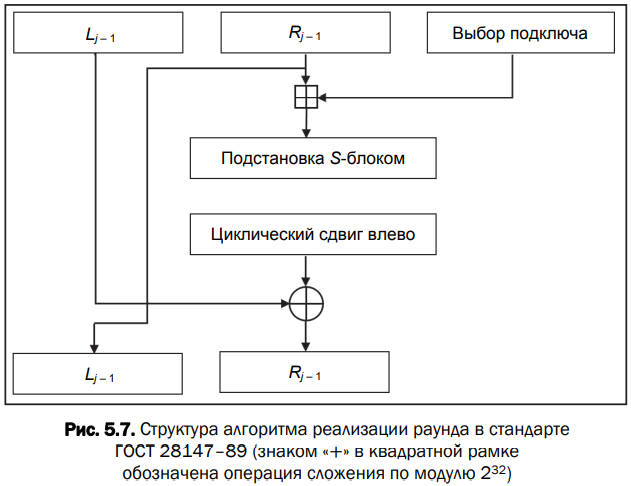
Алгоритм представляет собой классическую сеть Фейстеля. Шифруемый блок данных разбивается на две одинаковые части, правую R и левую L. Правая часть складывается по модулю 232 с подключом раунда и посредством принятого алгоритма шифрует левую часть. Перед следующим раундом левая и правая части меняются местами. Такая структура позволяет использовать один и тот же алгоритм как для зашифрования, так и для расшифрования блока (рис. 5.7). Таким образом, в алгоритме используются следующие операции:

• сложение слов по модулю 2^32: правый блок (Ri) складывается по модулю 2^32 с текущим подключом (Ki);

• циклический сдвиг слова влево на указанное число битов;

• побитовое сложение по модулю 2 (XOR);

• замена (подстановка в блоке S) по таблице.



Таким образом, в алгоритме используются следующие операции:

• сложение слов по модулю 2^32: правый блок (Ri) складывается по модулю ^232 с текущим подключом (Ki);

• циклический сдвиг слова влево на указанное число битов;

• побитовое сложение по модулю 2 (XOR);

• замена (подстановка в блоке S) по таблице.

## 💦46. Криптографические системы с открытым (публичным) ключом. Задача об укладке ранца.

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый, или публичный, ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения. Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

ВЫШЛО ДОХУЯ ЧТО СОКРАЩАТЬ НЕ ПАНИМАЮ

**Определение.** Ранцевый (рюкзачный) вектор S = (s1, ..., sz) – это упорядоченный набор из z, z ≥ 3, различных натуральных чисел si. Входом задачи о ранце (рюкзаке) называем пару (S, S), где S – рюкзачный вектор, а S – натуральное число.

Решением для входа (S, S) будет такое подмножество из S, сумма элементов которого равняется S.

**!Суть метода для шифрования состоит в том, что существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный. Трудный для укладки ранец применяется в качестве открытого ключа, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.**

В качестве закрытого ключа *d* (легкого для укладки ранца) используется сверхвозрастающая последовательность, состоящая из z элементов: d1, *d2, …, dz: d = {di}, i = 1, …, z.*

**Определение 3**. *Сверхвозрастающей* называется последовательность, в которой каждый последующий член больше суммы всех предыдущих. *{2,3,6,13,27,52,105,210}*

1. В качестве текущего выбирается число S, которое сравнивается с «весом» самого тяжелого предмета (dz); если текущий вес меньше веса данного предмета, то его в ранец не кладут (0), в противном случае его укладывают (1) в ранец и переходят к анализу очередного (в общем случае – i-го предмета).
2. Если на предыдущем (i-м шаге) предмет пополнил ранец, то текущий вес уменьшают на вес положенного предмета (S = S – di); переходят к следующему по весу предмету в последовательности: di – 1.

Шаги повторяются до тех пор, пока процесс не закончится. Если текущий вес уменьшится до нуля (S = 0), то решение найдено. В противном случае – нет

Открытый ключ e представляет собой нормальную (не сверхвозрастающую) последовательность. Он формируется на основе закрытого ключа и не позволяет легко решить задачу об укладке ранца.

Для получения открытого ключа e (e = {ei}, i = 1, …, z) все значения закрытого ключа умножаются на некоторое число a по модулю n:



Значение модуля n должно быть больше суммы всех чисел последовательности; кроме того, НОД (а, n) = 1

**Зашифрование сообщения.** Для зашифрования сообщения (М) оно сначала разбивается на блоки, по размерам равные числу (z) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (Si, i = 1, …, z): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя e.

Для расшифрования сообщения получатель (используя свой тайный ключ d: сверхвозрастающую последовательность) должен сначала определить такое обратное к а число а–1 , что



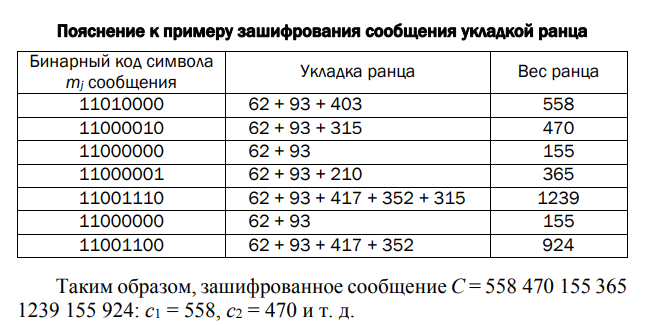
Для вычисления обратных чисел по модулю можно использовать известный нам расширенный алгоритм Евклида. После определения обратного числа каждое значение шифрограммы (ci) преобразуется в соответствии со следующим соотношением:

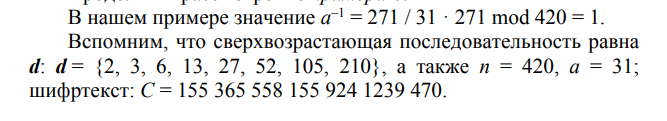


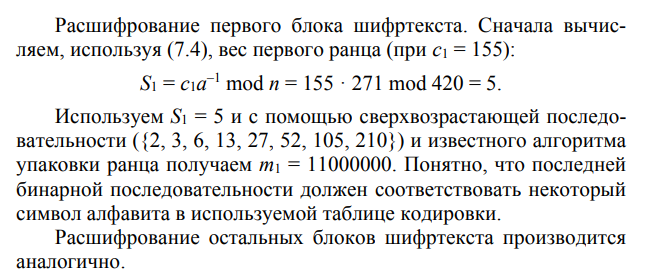
Полученное на основании последней формулы для каждого блока число далее рассматривается как заданный вес ранца, который следует упаковать по изложенному выше алгоритму, используя сверхвозрастающую последовательность (тайный ключ получателя).

**Пример 4.** Возьмем открытое сообщение M, состоящее из 7 букв (mj), которые представим в бинарном виде (1 символ текста – 1 байт).

е: {62, 93, 186, 403, 417, 352, 315, 210}.





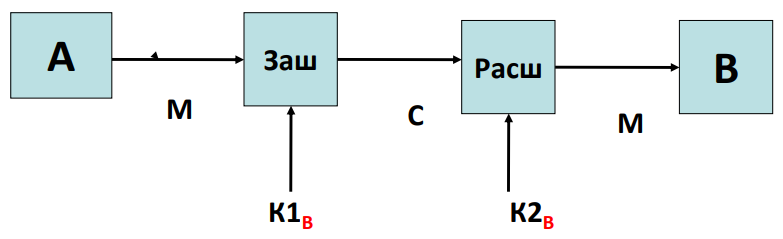


## 🍏47. Алгоритм передачи ключа по Диффи-Хеллману. Криптостойкость алгоритма

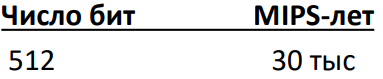
Авторы асимметричной криптографии У. Диффи и М. Хеллман - 1976 г.

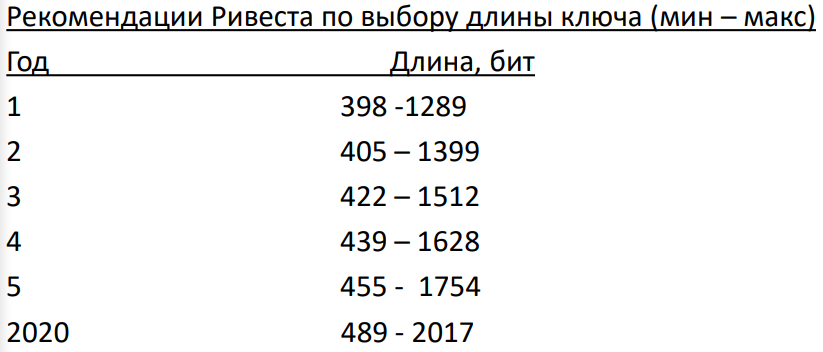
Идея – использовать ключи парами (К1: для заш и К2: для расш), которые очень трудно вычислить один из другого; ключ К1 известен и доступен для всех, ключ К2 - тайный

Упрощенная схема преобразования (Аня – А - передает зашифрованное сообщение Васе - В):

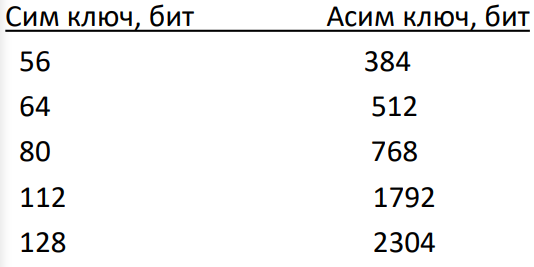


Разложение больших чисел на множители (p и q)





Длины симметр и асимметр ключей с равной устойчивостью к лобовому (путем перебора) вскрытию



А и В выбирают совместно большие простые ч: n и g

Протокол обмена:

1.А выб-т случ число х, выч-т X = g x mod n и отсылает Х В

2.В выб-т случ число у, выч-т У = gу mod n и отсылает У А

3. А выч-т k1 = Y x mod n

4. В выч-т k2 = Х у mod n

Получается: k1 = k2 = g ху mod n = k – секретный ключ

Зная n, g, X, Y, невозможно при больших значениях чисел определить

k – проблема дискр. логарифма

Протокол Диффи-Хеллмана является уязвимым для атаки, называемой "человек в середине".

Злоумышленник С может перехватить открытое значение, посылаемое от А к В, и послать вместо него свое открытое значение.

Затем он может перехватить открытое значение, посылаемое от В к А, и также послать вместо него свое открытое значение.

Тем самым С получит общие секретные ключи с А и В и сможет читать и/или модифицировать сообщения, передаваемые от одной стороны к другой.

## 🍏48. Алгоритм шифрования RSA. Реализация и криптостойкость

Безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших простых чисел. Предполагается, что восстановление открытого текста по шифртексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Для генерации двух ключей: тайного и открытого (а по сути – двух взаимосвязанных частей одного ключа, т. е. ключа, принадлежащего одному физическому лицу (или группе лиц), либо одному юридическому лицу), используются два больших случайных простых числа p и q. Для максимальной большей криптостойкости нужно выбирать p и q равной длины. Рассчитывается произведение: n = pq. Это есть один из трех компонент ключа, состоящего из чисел n, e, d.

Затем случайным образом выбирается второй компонент ключа (открытый ключ или ключ зашифрования, e, такой что e и (p – 1)(q – 1) являются взаимно простыми числами; вспомним, что (p – 1)(q – 1) = φ(n) – функция Эйлера). Б. Шнайер рекомендует число е выбирать из ряда: 3, 17, 216 + 1.

Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования d такого, что выполняется условие:



Другими словами:



Таким образом, сформирован ключ, состоящий из трех чисел, которые в свою очередь образуют две вышеупомянутые взаимосвязанные части: открытый (публичный) ключ (e, n) и тайный ключ (d, n; на самом деле, как видим, тайным здесь является лишь первое из пары чисел).

Для зашифрования/расшифрования используется ключ получателя: отправитель шифрует сообщение открытым ключом, а получатель расшифровывает шифртекст своим тайным ключом.

Зашифрование. Если шифруется сообщение М, состоящее из r блоков: m1, m2, …, mi, …, mr, то шифртекст С будет состоять из такого же числа (r) блоков, представляемых числами:



Расшифрование. Для расшифрования каждого зашифрованного блока производится вычисление вида:



Размер ключа в алгоритме RSA связан с размером модуля n. Два числа p и q, произведение которых равно n, должны иметь приблизительно одинаковую длину, поскольку в этом случае найти сомножители (факторы) сложнее, чем в случае, когда длина чисел значительно различается. Например, если предполагается использовать 768-битный модуль, то каждое число должно иметь длину приблизительно 384 бита. В 1999 г. 512-битный ключ был вскрыт за семь месяцев. Это означает, что 512-битные ключи уже не обеспечивают достаточную криптостойкость. Сейчас в критических системах применяются ключи длиной 1024 и 2048 битов.

## 🍏49. Алгоритм шифрования Эль-Гамаля. Реализация и криптостойкость

Безопасность алгоритма Эль-Гамаля, как и безопасность алгоритма Диффи – Хеллмана, основана на трудности вычисления дискретных логарифмов. Алгоритм Эль-Гамаля фактически использует схему Диффи – Хеллмана, чтобы сформировать общий секретный ключ для абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ.

И в случае шифрования, и в случае формирования цифровой подписи каждому пользователю необходимо сгенерировать пару ключей.

Рассматриваемый алгоритм отличается от алгоритма RSA несколькими параметрами и особенностями:

1) генерацией ключевой информации и числом компонент, составляющих ключ;

2) каждому блоку (символу) открытого сообщения в шифртексте на основе алгоритма Эль-Гамаля соответствуют 2 блока (в RSA – один-один);

3) в алгоритме Эль-Гамаля при зашифровании используется число (обозначим его k), которое практически никак не связано с ключевой информацией получателя и которое принимает (по определению) различные значения при зашифровании различных блоков сообщения.

Генерация ключевой информации. Выбирается простое число р. Выбирается число (g, g < p), являющееся первообразным корнем числа р – очень важный элемент с точки зрения безопасности алгоритма (см. ниже).

Далее выбирается число х (х < p) и вычисляется последний компонент ключевой информации:

y ≡ gх mod р. (8.8)

Владельцу сформированной ключевой информации, состоящей из 4 чисел, может посылаться некоторый шифртекст, созданный с использованием открытого ключа получателя: p, g, y. Расшифрование шифртекста получатель производит своим тайным ключом: p, g, х.

Как видим, на самом деле тайным является лишь одно число (как и в RSA): х.

Определение 2. Первообразный корень по модулю р является таким числом, что его степени (gi , 1 ≤ i ≤ p – 1) дают все возможные по модулю р вычеты (остатки), которые взаимно просты с p.

Понятно, что для больших значений р количество всех неповторяющихся остатков (р – 1) будет также большим. А поскольку в уравнении (8.8) мы используем модуль р большого простого числа и находим первообразным корень от р, который имеет важное свойство: при использовании разных степеней (аi = ах) решение будет равномерно распределяться от 0 до р – 1, то нахождение криптоаналитиком нужного х чрезвычайно затруднено. В этом заключается односторонность функции, задаваемой (8.8). И на этом основывается криптостойкость шифра Эль-Гамаля.

Для схемы вероятностного шифрования само сообщение и ключ не определяют шифротекст однозначно.

Зашифрование сообщения. Как ранее, предположим, что сообщение М = {mi}, где mi – i-й блок сообщения.

Зашифрование отправителем (каждого отдельного блока mi исходного сообщения) предусматривает использование, как это особо подчеркивалось выше, некоторого случайного числа k (1 < k < p – 1).

В силу использования случайной величины k шифр ЭльГамаля называют также шифром многозначной замены, а также схемой вероятностного шифрования.

Вероятностный характер шифрования является преимуществом для схемы Эль-Гамаля по сравнению, например, с алгоритмом RSA. Блок шифртекста (ci) состоит из двух чисел – аi и bi:



Здесь стал очевидным упомянутый недостаток алгоритма шифрования Эль-Гамаля: удвоение (реально – примерно в 1,5 раза) длины зашифрованного текста по сравнению с начальным текстом. Случайное число k должно сразу после вычисления уничтожаться. Расшифрование ci. Выполняется по следующей формуле:



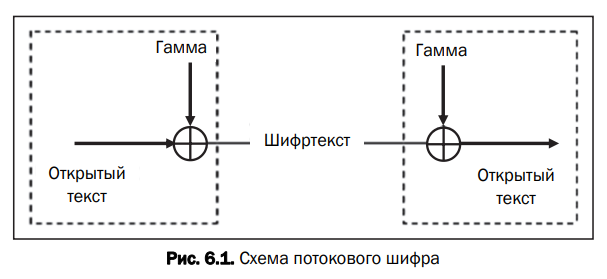
где (ax )–1 – обратное значение числа ax по модулю p. Нетрудно проверить, что (((ai)x )–1) ≡ gkх mod p. Еще раз возвратимся к криптостойкости рассмотренного алгоритма.

Если для зашифрования двух разных блоков (m1 и m2) некоторого сообщения использовать одинаковые k, то для соответствующих шифртекстов c1 = (a1, b1) и c2 = (a2, b2) выполняется соотношение b1(b2) –1 = m1(m2) –1. Из этого выражения можно легко вычислить m2, если известно m1.

При примерно одинаковой размерности ключей рассмотренные алгоритмы обеспечивают примерно одинаковый уровень криптостойкости

## 🍏50. Потоковое шифрование. Типы

Потоковый шифр (иногда говорят «поточный») – симметричный шифр, преобразующий каждый символ mi открытого текста в символ шифрованного ci, зависящий от ключа и расположения символа в тексте.



Типы поточных шифров:  
1) **Синхронные -**

* поток гаммы генерируется независимо от открытого текста и шифротекста;
* для успешного расшифрования необходимо синхрон-ть ключ с шифротекстом;

Свойства:

1. Искажение одного символа в шифротексте искажает только один символ в расшифр-м тексте (+)
2. Защита от любых вставок и удалений шифротекста, так как они приведут к потере синхронизации и будут обнаружены (+)
3. Нарушение синхр-ии (добавление или удаление символа) приводит к искажению всех символов после потери синхронизации (-)

**Синхронные потоковые шифры уязвимы к атакам на основе изменения отдельных битов шифртекста.**

2)**Самосинхронизирующиеся (асинхронные) -**

* значение ключа зависит либо от исходного текста, либо от шифротекста;
* поток ключей создается функцией ключа и фиксированного числа знаков шифртекста (N): внутреннее состояние генератора является функцией предыдущих N битов шифротекста - генератор потока ключей (при расшифровании), приняв N битов, автоматически синхронизируется с шифрующим генератором

Свойства:

1. Т.к. каждый знак открытого текста влияет на следующий шифртекст, статические свойства открытого текста распространяются на весь шифртекст (+)
2. ошибочно удаленный или добавленный символ (бит) вызывает только ограниченное кол-во ошибочных символов в дешифрованном тексте, после чего правильный текст восстанавливается (+)
3. каждому неправильному биту шифртекста соответствует N ошибок в открытом тексте (-)

**Недостаток этих потоковых шифров – распространение ошибок, так как искажение одного бита в процессе передачи шифртекста приведет к искажению нескольких битов гаммы и, соответственно, расшифрованного сообщения**.

## 🍏51. Гаммирование в потоковом шифровании. Генераторы ключа

Ключ должен быть максимально похож на **случайную** числовую последовательность.

Ключевые последовательности (случайные последовательности (СП), либо псевдослучайные последовательности (ПСП)) вырабатываются специальными блоками систем потокового шифрования – **генераторами**.

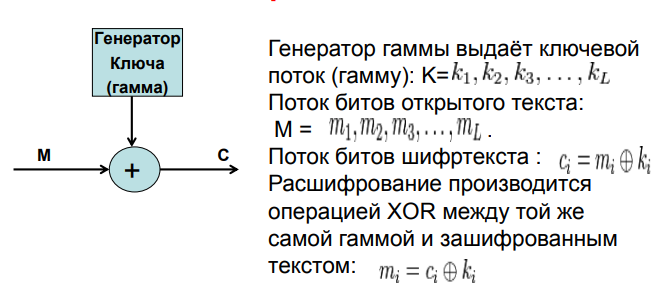
**Случайные числа** (последовательности) – последовательность элементов, каждый из которых не может быть предсказан (вычислен) только на основе знания предшествующих ему элементов данной последовательности;

**Псевдослучайные числ**а – последовательность элементов, полученная в результате выполнения некоторого алгоритма и используемая в конкретном случае вместо последовательности случайных чисел.

Ключ (гамма) должен обладать тремя критически важными свойствами:

* быть истинно случайным (последовательность, полученная с использованием любого алгоритма, является не истинно случайной, а псевдослучайной);
* совпадать по размеру с заданным открытым текстом;
* применяться только один раз.

Идея гаммирования для ПШ



Если последовательность битов гаммы не имеет периода и выбирается случайно, то взломать шифр невозможно.

**Эффективный Генератор ключа** – главная проблема ПШ: генерирование длинных ПСП

Есть несколько разновидностей генераторов псевдослучайной последовательности (ПСП):

1. **Aлгоритм на основе линейного конгруэнтного генератора**

описывается рекуррентным соотношением:

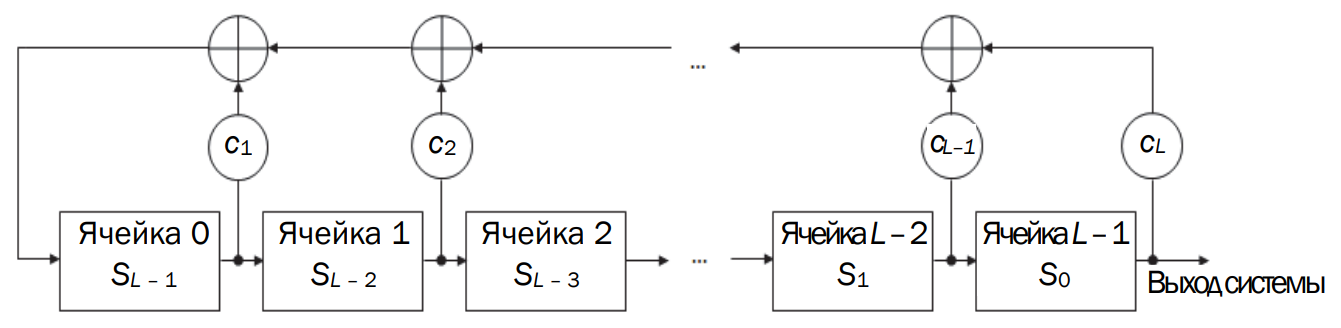
**xt+1 = (a\*xt + c) mod N,**

х0 – начальное значение ПСП, a – множитель, с – приращение, N - мощность алфавита

При с=0 – мультипликативный конгруэнтный ген-р ПСП

1. **На основе регистров сдвига**

РСЛОС строятся на основе примитивных порождающих полиномов (многочленов). Если многочлен является неприводимым, то период ПСП при ненулевом начальном условии (ненулевом состоянии) регистра будет максимально возможным: 2L – 1.

****

1. **на основе алгоритма RSA**

Последовательность генерируется с использованием соотношения

**xt ≡ (xt – 1)^е mod n**

Начальными параметрами служат n, большие простые числа p и q (причем n = pq), целое число е, взаимно простое с произведением (р – 1)(q – 1), а также некоторое случайное начальное значение x0.

Выходом генератора на t-м шаге является младший бит числа xt.

Безопасность генератора опирается на сложность взлома алгоритма RSA, т. е. на разложение числа n на простые сомножители.

1. **на основе алгоритма BBS**

Начальное значение x0 генератора вычисляется на основе соотношения

**x0 ≡ x^2 mod n**,

где n, как и в генераторе на основе RSA, является произведением простых чисел p и q, однако в нашем случае **эти простые числа должны быть сравнимы с числом 3 по модулю 4**, т. е. при делении p и q на 4 должен получаться одинаковый остаток 3; число x должно быть взаимно простым с n; число n называют числом Блюма.

Выходом генератора на t-м шаге является младший бит числа xt:

**xt ≡ (xt – 1)^2 mod n.**

С точки зрения безопасности важным является свойство рассмотренного генератора, заключающееся в том, что при известных p и q xt-й бит легко вычисляется без учета предыдущего (xt – 1) бита:

**xt ≡ (x0)^a mod n**,

где **а ≡ 2^t mod ((p – 1)(q – 1)).**

## 🍏52. Особенность шифра Вернама

One-time pad - шифр **одноразовых блокнотов**.

**Зашифрование** - открытый текст объединяется операцией “XOR” с ключом (одноразовым блокнотом или шифроблокнотом).

Ключ (гамма) должен обладать тремя критически важными свойствами:

* быть истинно случайным (последовательность, полученная с использованием любого алгоритма, является не истинно случайной, а псевдослучайной);
* совпадать по размеру с заданным открытым текстом;
* применяться только один раз.

Шифр Вернама является самой безопасной криптосистемой из всех возможных - доказал Шеннон. *(перехват шифротекста без ключа не даёт никакой информации о сообщении.)*

Шифр назван в честь телеграфиста Гильберта Вернама, который сконструировал телеграфный аппарат, автоматически кодирующий сообщения таким методом (ключ подавался на отдельной ленте).

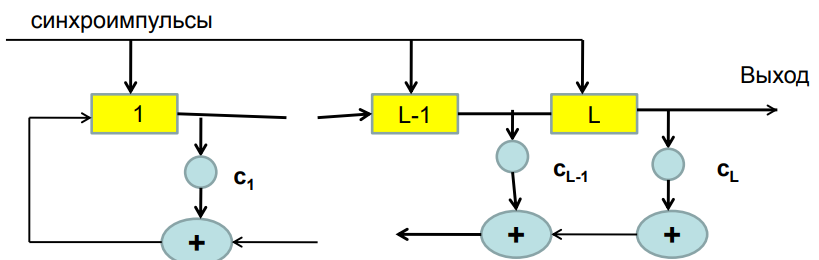
Легко заметить, что нельзя использовать один и тот же ключ несколько раз - при кодировании одинаковых сообщений с одинаковым ключом, полученные сообщения также будут одинаковыми, что позволит анализировать передаваемые сообщения.

Используется редко, в связи с большой длиной ключа, но иногда применяется для передачи **особо важных тайных сообщений** (например линия Вашингтон-Москва во время Второй Мировой Войны).

На практике можно один раз физически передать носитель информации с длинным истинно случайным ключом, а потом по мере необходимости пересылать сообщения. На этом основана идея **шифроблокнотов**: шифровальщик по дипломатической почте или при личной встрече снабжается блокнотом, каждая страница которого содержит ключи. Такой же блокнот есть и у принимающей стороны. Использованные страницы уничтожаются.

## 🍏53. Принципы построения генераторов ПСП на основе регистров сдвига в системах потокового шифрования

Достаточно распространенным является использование регистров сдвига (РС) в качестве генераторов ПСП в силу простоты реализации на основе цифровой логики. РС с линейной обратной связью (РСЛОС) состоит из двух частей: собственно РС и функции обратной связи.



Выходная послед-ть определяется начальным состоянием каждого Тг (общее число – L: от 1 до L) и видом ФОС, Чаще всего ФОС – XOR – РСЛОС (регистр сдвига с линейной обратной связью),

**Период регистра сдвига** — длина получаемой последовательности до начала еѐ повторения

**Свойства**:

1. В течение каждой единицы времени (за такт) выполняются следующие операции: содержимое ячейки L формирует часть выходной последовательности; содержимое i-й ячейки перемещается в ячейку i+1 новое содержимое ячейки 1 определяется битом обратной связи, который вычисляется сложением по модулю с определѐнными коэффициентами ci битов ячеек .

2. Так как существует 2^L - 1 разных ненулевых состояний регистра, то период последовательности, генерируемой РСЛОС при любом ненулевом начальном состоянии, не превышает 2^L - 1.

3. Свойства ПСП зависят от ассоциированного многочлена :

C(x) = 1+c1x+c2x2+ … + cLxL

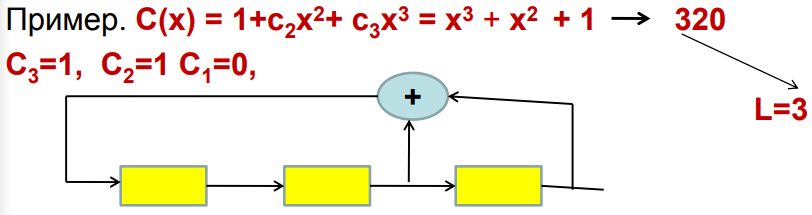
Его ненулевые коэффициенты называются отводами, (как и соответствующие ячейки регистра, поставляющие значения аргументов функции обратной связи).

Важное свойство многочлена C(x) - приводимость.

Многочлен называется приводимым, если он может быть представлен как произведение двух многочленов меньших степеней с коэффициентами из данного поля (в нашем случае с двоичными коэффициентами).

Если нет, то многочлен называется неприводимым.

Если многочлен является неприводимым, то период ПСП будет максимально возможным : 2^L - 1

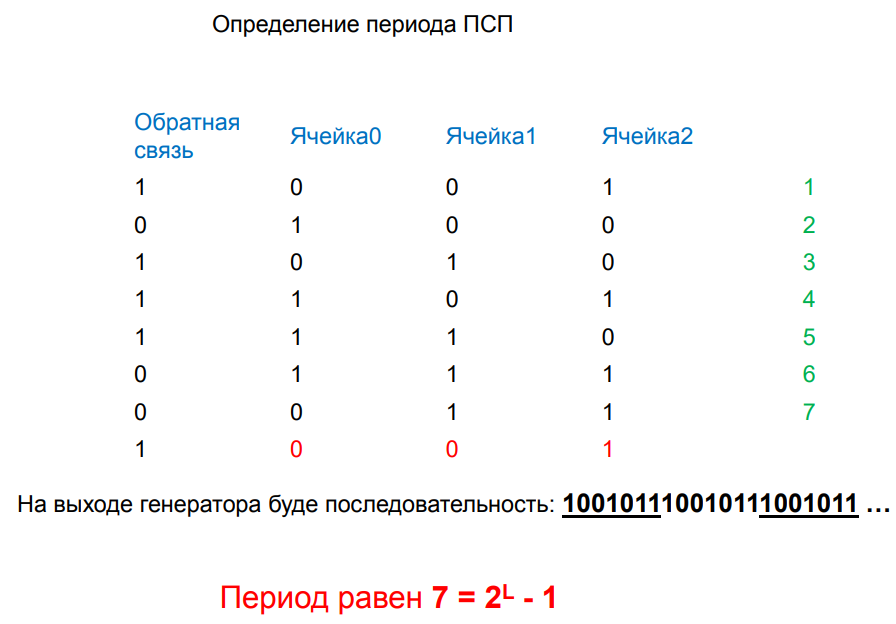


Многочлен x3 + x2 + 1 (320) является неприводимым.

Примеры других неприводимых многочленов:

210 → c2x2 + c1x + 1

310, 410, 520, ….., 84320, …



## 🍏54. Понятие эллиптичеcкой кривой. Алгебраическое и геометрическое представление

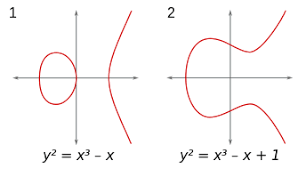
**Эллиптические кривые** – математический объект, который может быть определен над любым полем.

**Эллиптическая кривая над вещественными числами** – это множество точек, описываемых уравнением



при том константы (a и b - вещественные числа) должны удовлетворять условию:





Также частью ЭК является бесконечно удаленная точка, которая является нейтральным элементом и делает точки ЭК абелевой группой. Бесконечно удаленная точка была придумана спустя 16 веков после открытия эллиптических кривых с помощью проективного подхода к задаче.

## 💩55. Сложение и умножение точек ЭК над вещественными числами

Единичный (нейтральный) элемент – это бесконечно удаленная точка О;

Обратная величина точки R – это точка, симметричная относительно оси Х;

Сложение задается следующим правилом: сумма трех ненулевых точек P, Q и –R, лежащих на одной прямой, будет равна P + Q + (–R) = О.

В соответствии с этим можем сформулировать законы сложения точек эллиптической кривой:

• прямая, проходящая через точки R и –R, является вертикальной прямой, которая не пересекает ЭК ни в какой третьей точке; если R = (х, –у), то R + (х, у) = О. Точка (х, у) является отрицательным значением точки R и обозначается –R. Таким образом, по определению R + (–R) = О;

• P + Q = R: пусть P и Q – две различные точки ЭК (рис. 11.1), и Р не равно Q; если проведем через P и Q прямую, то она пересечет ЭК еще только в одной точке, называемой –R; точка –R отображается относительно оси Х в точку R, равную сумме точек P и Q: P + Q = R.

Геометрическая интерпретация операции сложения двух точек показана на рис. 11.1.



Что будет, если P = Q? В этом случае мы можем говорить об операции удвоения точки: P + Р = 2Р. Обобщив (к точке 2Р можно прибавить еще раз точку Р: 2Р + Р), сформулируем принцип умножения точки Р на целое положительное число n – это сумма n точек Р: nP = P + P + P + … + P.

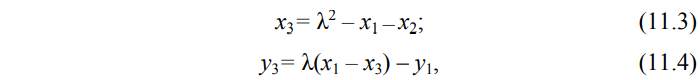
Скалярное умножение осуществляется посредством нескольких комбинаций сложения и удвоения точек эллиптической кривой.

Например, точка 25P может быть представлена как 25P = = 2(2(2(2P)) + 2(2(2P))) + P.

Понятно, что каждая точка на плоскости задается парой координат: х и у.

Числа х и у являются рациональными, а точки P, Q, R и –R (как и любые точки ЭК) – рациональными точками.

Если Р = (х1, у1) и Q = (х2, у2), то Р + Q = (х3, у3) определяется в соответствии с правилами:



где



Из этого следует, что число λ – угловой коэффициент секущей, проведенной через точки Р = (х1, у1) и Q = (х2, у2). При Р = Q секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

## 🍏56. Сложение и умножение точек ЭК над конечными полями

См. 55. Просто в каждую операцию в конце добавить mod P

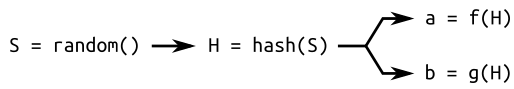
## 🍏57. Принципы построения криптосистемы на эллиптических кривых

1. Закрытый ключ — это случайное целое d, выбранное из {1, …, n-1} (где n — порядок подгруппы).
2. Открытый ключ — это точка H = dG (где G — базовая точка подгруппы).

Если мы знаем d и G (вместе с другими параметрами области определения), то найти H «просто». Но если мы знаем H и G, то поиск закрытого ключа d является «сложной» задачей, потому что требует решения задачи дискретного логарифмирования.

Существуют определённые классы эллиптических кривых, которые довольно слабы и позволяют использовать специальные алгоритмы для эффективного решения задачи дискретного логарифмирования. Например, все кривые, для которых p = hn (то есть порядок конечного поля равен порядку эллиптической кривой), уязвимы к [атаке Смарта](http://interact.sagemath.org/edu/2010/414/projects/novotney.pdf), которую можно использовать для решения дискретных логарифмов за полиномиальное время на классических компьютерах.

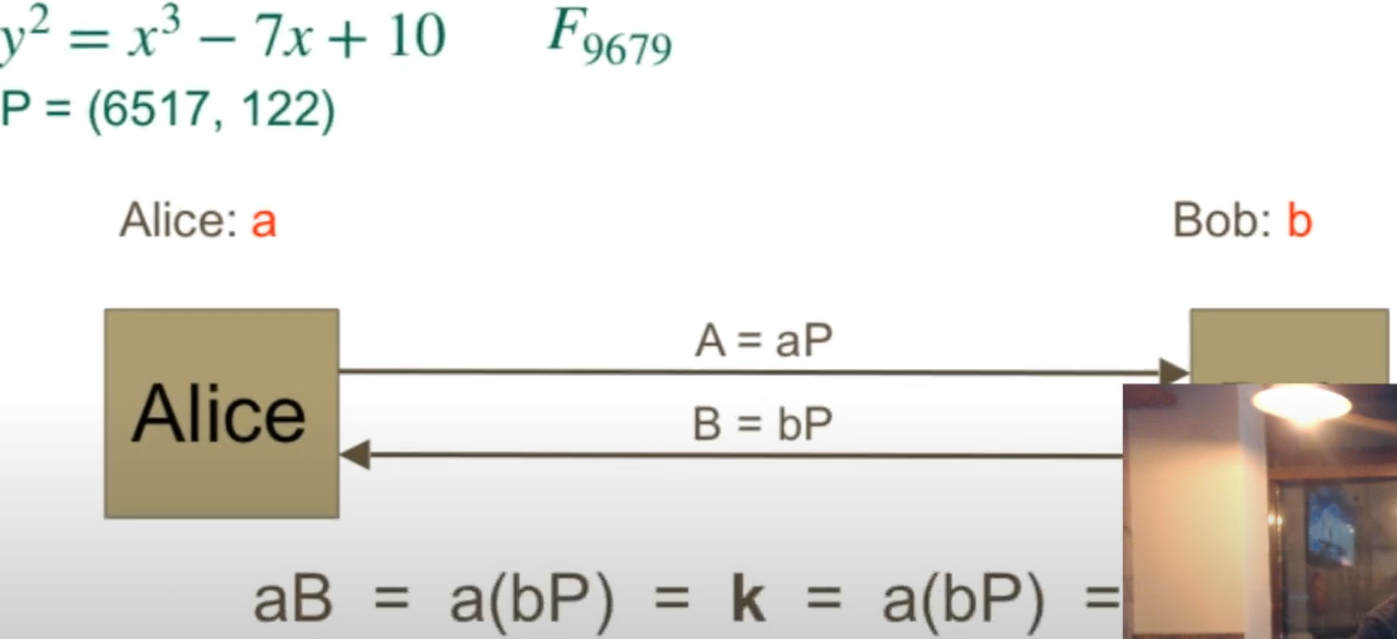
Чтобы решить эту проблему, иногда приходится использовать дополнительный параметр области определения: порождающее значение (seed) S. Это случайное число, используемое для генерирования коэффициентов a и b или базовой точки G, или того и другого. Эти параметры генерируются вычислением хеша S. Хеши, как мы знаем, «просто» вычислить, но «сложно» реверсировать.



Сгенерированная с помощью порождающего значения кривая называется проверяемо случайной. Принцип использования хешей для генерирования параметров известен как "[nothing up my sleeve](http://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number)" («в рукавах ничего нет»), и широко распространён в криптографии.

## 🍏58. Система распределения криптографических ключей на основе эллиптической кривой

Система распределения ключей на основе эллиптической кривой Диффи-Хеллмана (ECDH)



Как и в обычном алгоритме Диффи-Хеллмана,

1. Алиса и Боб генерируют свои приватные ключи a (alice) и b(bob)
2. Алиса умножает генерирующую точку на свой ключ a и получает A = aP. Боб делает то же самое и получает точку B = bP
3. Они обмениваются полученными точками
4. Алиса умножает точку Боба на свой ключ и получает k = a(bP), а Боб получает точку k = b(aP). Легко видно, что эти точки равны между собой и являются приватным ключом для дальнейшего обмена зашифрованными сообщениями.

Сложно найти конечную точку (приватный ключ). При использовании больших p у ЭК будет огромное количество точек и перебор всех будет очень длительным.

## 💦59. ЭЦП. Назначение и свойства.

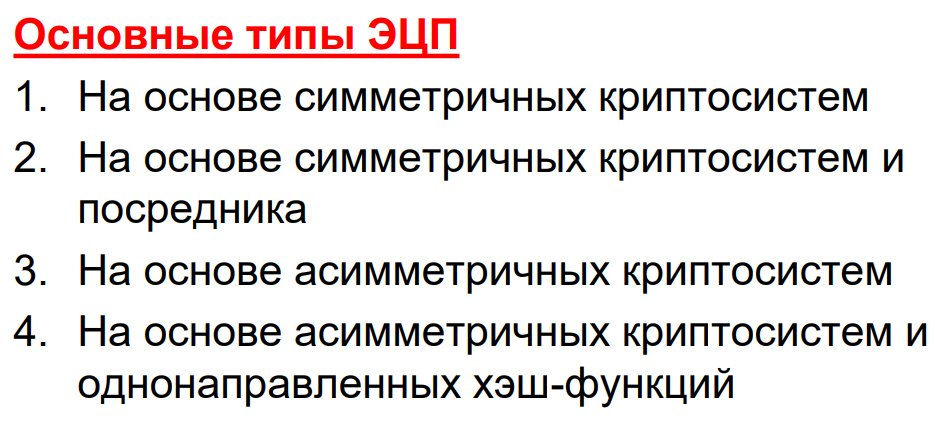
ЭЦП – бинарная последовательность, которая добавляется к подписываемому документу М (также представленному в двоичной форме) и зависящая от М и от ключа (К)

Похожа на графическую подпись по функциям, но в отличие от графической является бинарной последовательностью и **зависит от сообщения.**

**В ЭЦП в отличие от асимметричного шифрования секретным ключом сообщение подписывается, а открытым - валидируется.**

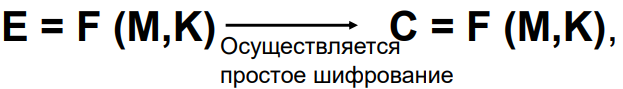
***Функции и назначение ЭЦП***

1. Подпись должна быть достоверна: подписавший документ человек сделал это осознано.
2. Подпись неподдельна. Подписавший — автор подписи.
3. Подпись невозможно использовать повторно, мошенник не должен иметь возможность переносить подпись без ведома подписавшегося.
4. Подписанный документ не может быть изменен, особенно, если сделано несколько копий.
5. От подписи нельзя отречься

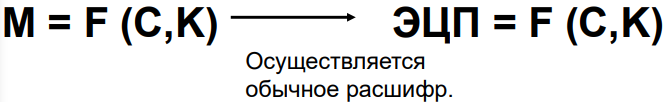


## 💩60. ЭЦП. Основные методы генерации. Атаки на ЭЦП

## 💦61. ЭЦП на основе симметричной криптографии (какая-то фигня из лекции)

Генерация ЭЦП: 

т.е. Е ≡ С; физически ЭЦП нет Проверка подлинности сообщения (верификация подписи)



Если сообщ. расшифровано тайным ключом (К), известным только А и В, то это подтверждает аутентичность документа

## 🍏62. Хеш-функция. Особенности и применение методов хеширования.

СМ. вопрос 34.

## 💩63. Хеширование сообщений на основе алгоритмов класса MD.

## 💩64. Хеширование сообщений на основе алгоритмов класса SHA

## 💦65. ЭЦП на основе алгоритма RSA

2 ситуации:

* сообщение Мo подписывается и передается в открытом (незашифрованном) виде;
* сообщение Мo подписывается и передается в зашифрованном виде.

В открытом виде:

| S - подпись,  dо и no – элементы тайного ключа отправителя  М' = Мo||S. (|| - конкатенация) - передаваемое сообщение  Валидация:    Условие валидиции:  H(Mo) = H(Mп) |
| --- |

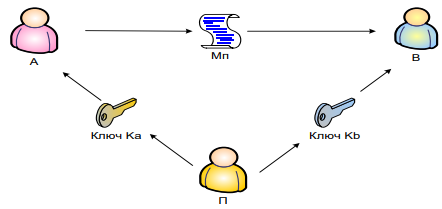
В закрытом виде:

Если подписываемое сообщение М(М') также должно передаваться в зашифрованном виде, то обычно М' шифруется на стороне отправителя стандартным образом: с помощью открытого ключа получателя (еп и nп), который перед основным процессом верификации подписи расшифровывает послание своим тайным ключом: dп и nп. Далее осуществляются вычисления и анализ, как и в первом случае.

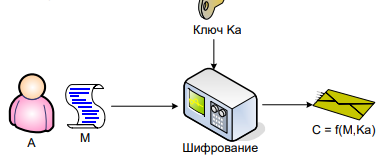
## 🍏66. ЭЦП на основе симметричной криптосистемы и посредника

ЭЦП на основе симметричных криптосистем и посредника

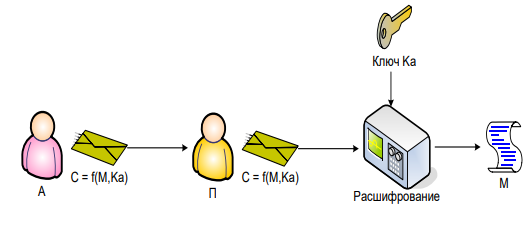
1. П (посредник, СЦ) вырабатывает для А и В разные ключи



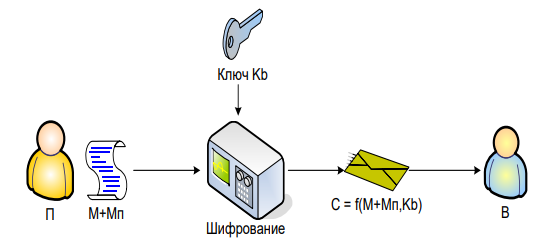
2. А шифрует М ключом Ка и отсылает его П



3. П расшир-т С ключом Ка, включает в него инф-ю, что оно получено от А (М’)



4. П шифрует М’ ключом Кв и отправляет его В



5. В расшифр-т ключом Кв, и читает М’ с гарантией П



Таким образом:

1. Подпись достоверна (П – гарант)

2. Подпись неподдельна (только А и В знают кл: П – абсолютное доверие)

3. Подпись невозможно использовать повтор.

4. Подпис-й док-т нельзя изменить

5. Подпись нельзя отрицать

## 💩67. ЭЦП DSS.

## 🍏68. ЭЦП на основе алгоритма Эль-Гамаля

Ключевая информация отправителя для ЭЦП состоит из тех же элементов, что и ключи в DSA. Основное отличие в том, что результатом зашифрования является только одна пара чисел, а не пара для каждого блока исходного сообщения. Причём рассматриваемым сообщением является хеш подписываемого документа: H(M0).

Ключевая информация отправителя: открытый ключ: y, g и p. Тайный ключ: x. Чтобы подписать сообщение М0, обладатель используемых для ЭЦП ключей должен выбрать случайное число k, взаимно простое с (p-1). Затем вычисляются числа a и b, являющиеся цифровой подписью (S = {a, b})



для вычисления b с помощью расширенного алгоритма Евклида решается уравнение:



Получателю отправляется сообщение: M’ = M0||S.

Для верификации подписи вычисляется хеш полученного сообщения H(MП) = h. Далее нужно убедиться, что выполняется равенство:



Если равенство выполняется, подпись верифицируется.

## 🍏69. ЭЦП на основе асимметричной криптографии и хеш-функций

Рассмотрим на примере алгоритма Эль-Гамаля.

Зашифрованное сообщение M или каждый зашифрованный блок (mi) этого сообщения состоят из двух чисел. Блок состоит из чисел ai и bi:



Поскольку символы a и b удобнее использовать в обозначении параметров ЭВ, то блок шифротекста зададим символами Ci1 и Ci2.

При использовании ЭК зашифрование предполагает представление сообщения в виде точки P (или представления каждого блока сообщения в виде разных точек Pi) ЭК с известной точкой G и известным Q. Соответственно шифротекст - это две точки на той же ЭК: C1 и C2 или Ci1 и Ci2.

Предположим, что шифруемое сообщение M - это точка P на ЭК.

Сторона А выбирает некоторое случайное число k и далее выполняет вычисления с использованием открытого ключа стороны В:



Получатель для расшифрования сообщения вычисляет:



Знак “-” в последней формуле означает сложение с инверсией: инверсией по отношению к точке (x, y) является точка (x, -y) на ЭК.

## 🍏70. ЭЦП на основе эллиптической кривой

Как пример генерации и верификации ЭЦП на основе алгоритма DSA и ЭК. Во внимание принимаем: порядок точки G, т. е. число q. Полагаем, что отправитель подписывает хеш H(M) сообщения M.

Генерация ЭЦП.

1. Выбрать число k (1<k<q), q - порядок точки G.
2. Вычислить точку kG = (x, y), вычислить r =x mod q; если r = 0, возвращаемся на этап 1 и меняем k.
3. Вычисляем t=k-1 mod q (например, на основе расширенного алгоритма Евклида)
4. Вычислить s = (t(H(M) + dr))mod q; если s = 0, возвращаемся на этап 1 и меняем k.

Стороне В отсылаются сообщение М и ЭЦП (числа r и s).

Верификация ЭЦП. Получатель знает алгоритм хэширования, который использовался отправителем, открытый ключ отправителя, с помощью чего выполняет следующие операции над M и полученной ЭЦП (обозначения чисел без изменений).

1. Проверить выполнение условия: 1<r, s<q; если условие не выполняется, то легитимность подписи не подтверждается, а противном случае выполняются дальнейшие шаги.
2. Вычисляются H(M) и w=s-1 mod q.
3. Вычисляются u1 = w\*H(M) mod q; u2 = w\*r mod q.
4. Вычисляются Gu1 + Qu2 = (x’, y’), v = x’ mod q.
5. Сравниваются v и r, если равенство выполняется, подтверждается легитимность подписи и целостность полученного сообщения.

## 💩71. Стандарт ЭЦП в РБ.

## 💩72. Общая характеристика криптовалютных технологий.

## 🍏73. Общая характеристика и классификация методов сжатия информации.

Большинствo типов данных хар-ся избыточностью:

Степень избыточности видеоданных > степени избыточности графических данных > степень избыточности текстовых данных

Фактором, влияющим на степень избыточности является принятая система кодирования

**Избыточность** - центральное понятие в теории сжатия информации. Любые данные с избыточной информацией можно сжать. Данные, в которых нет избыточности, сжать нельзя

| **Теорема Шеннона** об источнике шифрования показывает, что невозможно сжать данные настолько, что оценка кода (среднее число бит на символ) станет меньше, чем **энтропия Шеннона** исходных данных, без потери точности информации. |
| --- |

1. ***По степени соответствия входных и выходных данных***

* ***С потерями информации*** 
  + JPEG - для графических данных;
  + MPG - для для видеоданных;
  + MP3 - для аудиоданных.
* ***Без потерь информации*** 
  + GIF, TIFF - для графических данных;
  + AVI - для видеоданных;
  + ZIP, ARJ, RAR, CAB, LH - для произвольных типов данных

1. ***По способу кодирования (преобразования)***

* На основе выявления повторяющихся последовательностей данных и замены их более простой структурой (интервалов, RLE, Б-У) - **символориентированные**
* Сжатие по **ключевым словам**: кодирование лексических единиц группами байт фиксированной длины (ЛЗ)- **словарные**
* На основе использование частотных (вероятностных) свойств символов входного алфавита (Хаффмана, Шеннона-Фано)- **Вероятностные (статистические)**
* **Арифметические**

## 💦74. Метод сжатия данных Лемпеля-Зива.

Классический алгоритм Лемпеля − Зива – LZ77, названный так по году представления метода, формулируется следующим образом: «если в проанализированном (сжатом) ранее выходном потоке уже встречалась подобная последовательность байт, причем запись о ее длине и смещении от текущей позиции короче, чем сама эта последовательность, то в выходной файл записывается ссылка (смещение, длина), а не сама последовательность».

Суть метода LZ77 (как и последующих его модификаций) состоит в следующем: упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также скользящим словарем (англ. sliding dictionary), варьируются в разных реализациях систем сжатия.

Скользящее окно имеет длину n, т.е. в него помещается n символов, и состоит из двух частей:

* последовательности длины n1 = n - n2 уже закодированных символов (словарь);
* упреждающего буфера (Буфера предварительного просмотра, lookahead) длиной n2 - буфера кодирования

Пусть к текущему моменту времени закодировано t символов: S1, S2, …, St. Тогда словарём будут являться n1 предшествующих символов St-(n1-1), St-(n1-1)+1, …, St.

В буфере находятся ожидающие кодирования (сжатия) символы St+1, St+2,.., St+n2. Если n2>= t, то словарем будет являтся все уже обработанная часть входной последовательности.

**!Нужно найти самое длинное совпадение между строкой буфера кодирования, начинающейся с символа St+1, и всеми фразами словаря.**

Эти фразы могут начинаться с любого символа St-(n1-1), St-(n1-1)+1,...,St, выходить за пределы словаря, вторгаясь в область буфера, но должны лежать в окне. Длина совпадения не должна превышать размера буфера. Полученная в результате поиска фраза St-(p-1), St-(p-1)+(q-1) кодируется с помощью двух чисел:

1. смещения от начала буфера p;
2. длины соответствия, или совпадения q.

Cсылки (p и q - указатели) однозначно определяют фразу. Дополнительно в выходной поток записывается символ s, следующий за строкой буфера.

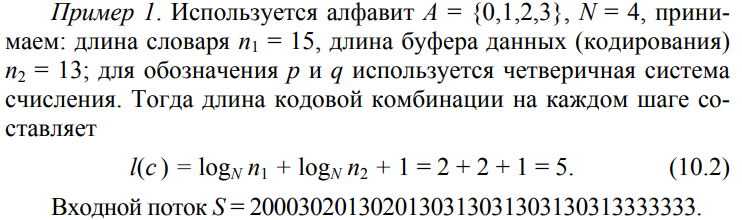
Длина кодовой комбинации (триады - p,q,s) на каждом шаге определяется соотношением:



N- мощность алфавита.

После каждого шага окно смещается на q+1 символов вправо и осуществляется переход к новому циклу кодирования. Величина сдвига объясняется тем, что мы реально закодировали именно q+1 символов: q- с помощью указателя и 1 - с помощью тривиального копирования.

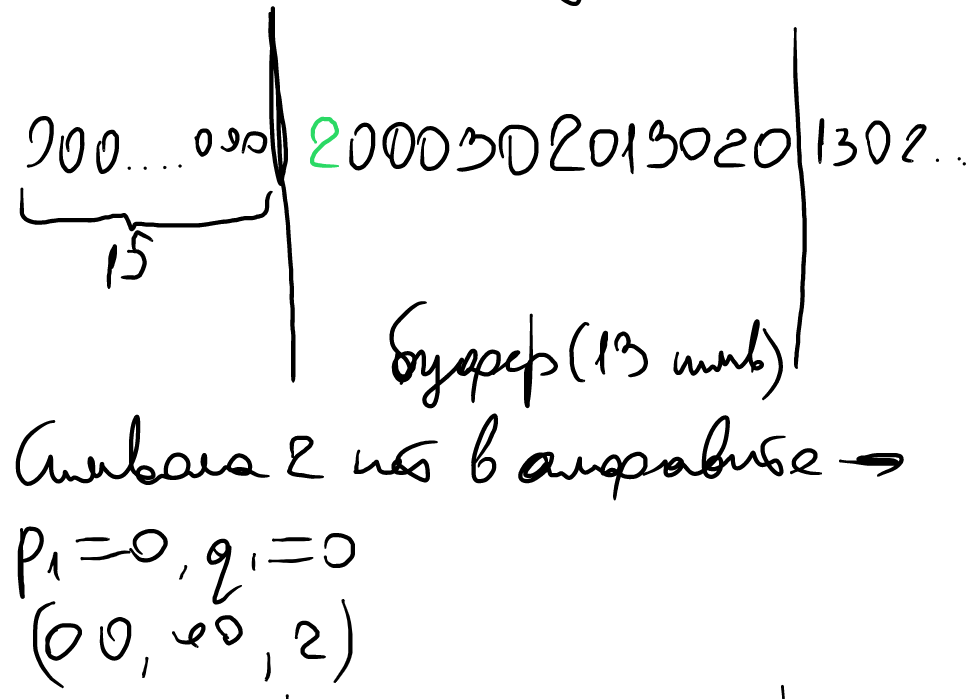
Передача одного символа в явном виде s позволяет разрешить проблема обработки еще ни разу не встречавшихся символов, но существенно увеличивается размер сжатого блока.



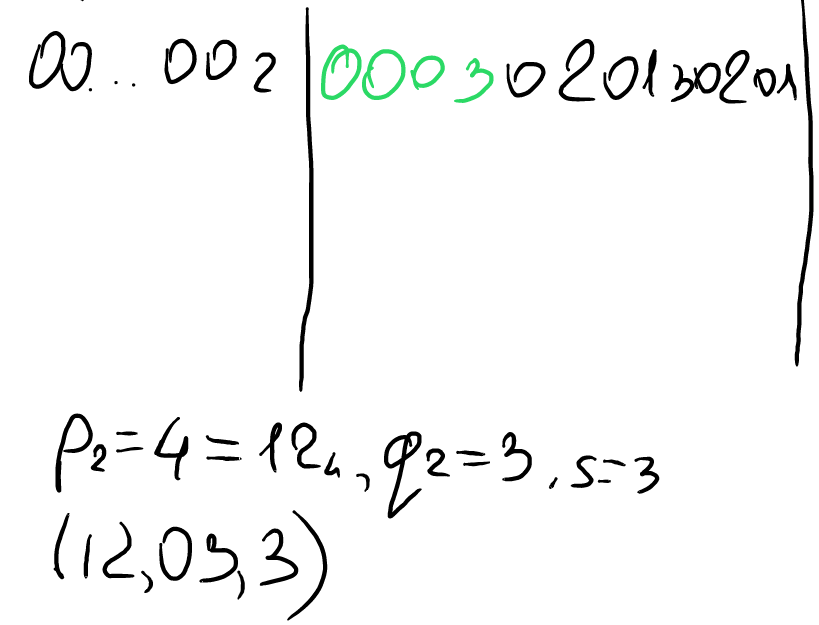
Прямое преобразование

1 шаг.

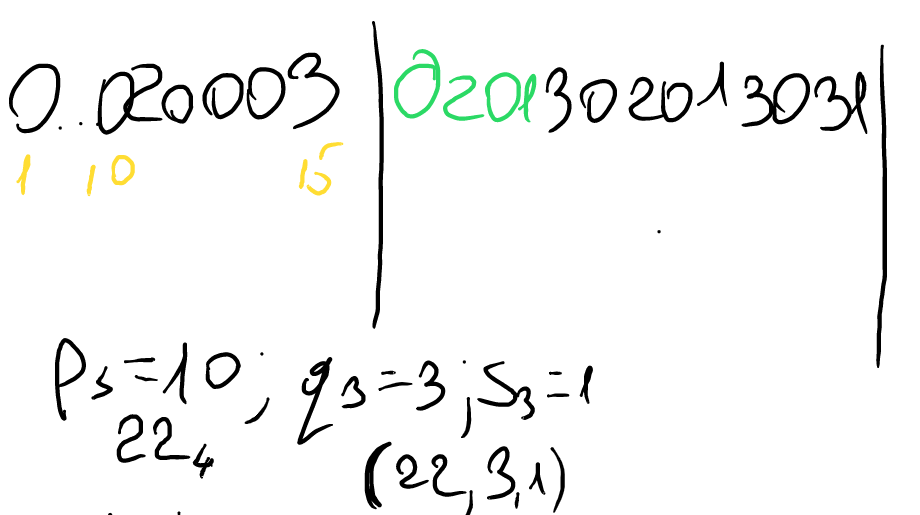
Сначала заполняем алфавит нулями или другими символами по желанию (по ситуации)



2 шаг.

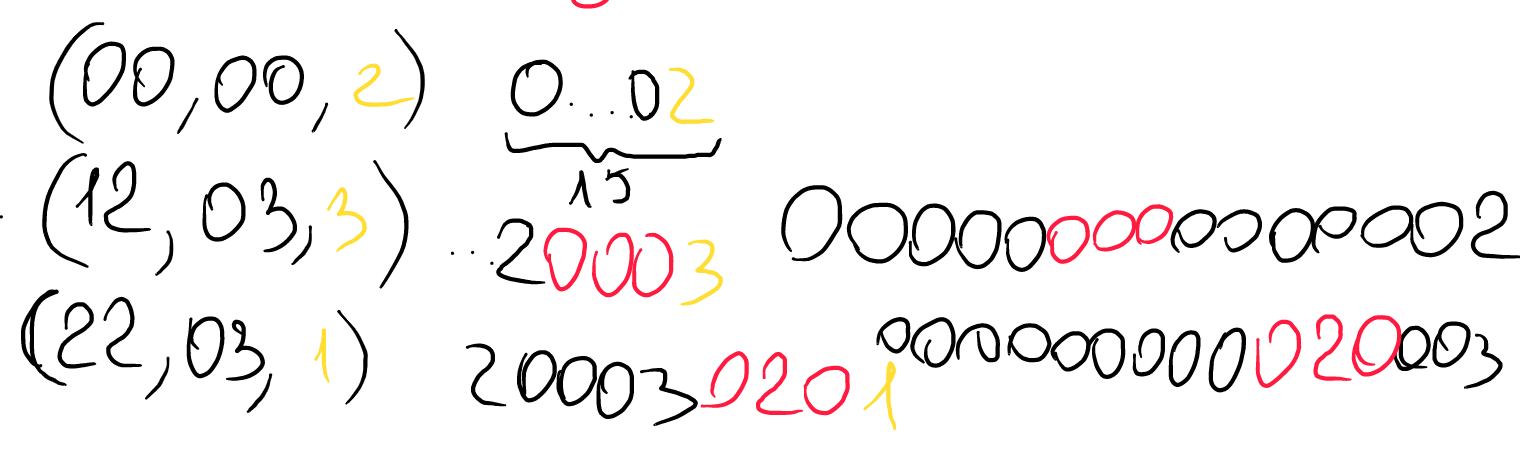


3 шаг.



И так далее

Обратное преобразование



## 🍏75. Метод сжатия данных Барроуза-Уиллера.

Сам по себе, **BWT** не является классическим алгоритмом сжатия, однако выходная последовательность гораздо ***удобнее для сжатия***, нежели исходная.

Преобразование Б-У применяется в алгоритмах сжатия качественных данных.

Является компромиссным между быстрыми словарными алгоритмами и статистическими алгоритмами. Обычно применяется **совместно** со статистическими алгоритмами.

Оперирует целым блоком данных длиной **k** символов

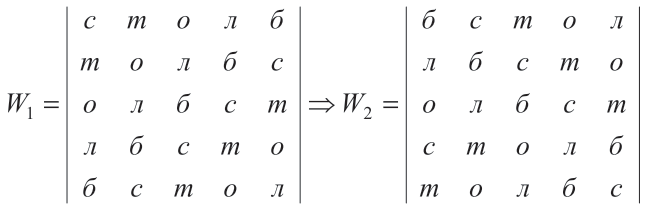
***Прямое преобразование***:

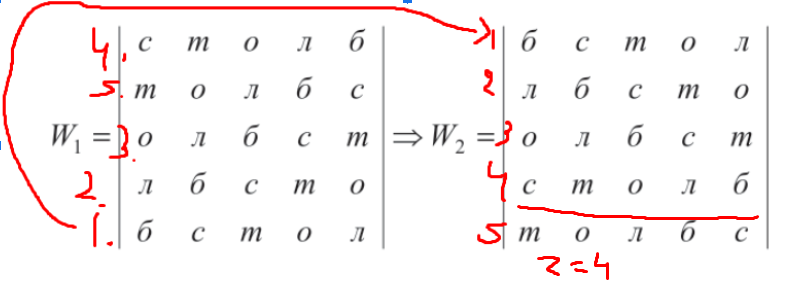
1) выделяется блок данных (строка длиной k символов некоторого алфавита мощностью N), который обозначим символом М;

2) составляется таблица W1 размером k×k всех циклических сдвигов входной строки M: W1 = (M);

3) производится лексикографическая (в алфавитном порядке) сортировка строк таблицы W1, в результате чего получается таблица W2 того же размера;

4) в качестве выходной строки (обозначим ее BWT(М), z) выбирается последний столбец (Мk) таблицы W2 преобразования и номер строки z, совпадающей с исходной строкой М. Как видим, выходная строка (сжатое сообщение) всегда по объему превышает входную.





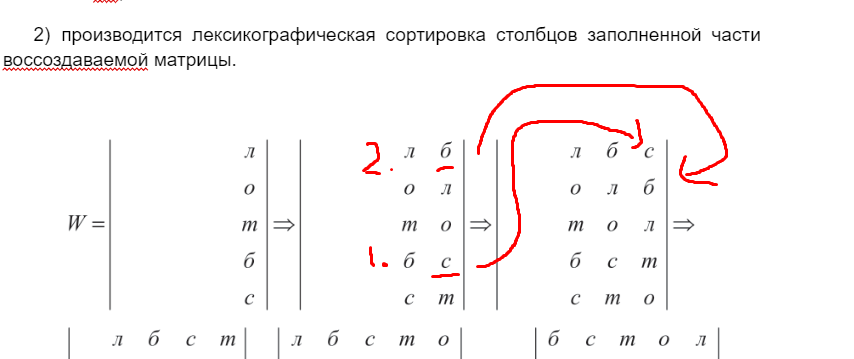
(z может соответствовать нескольким строкам (т.е несколько одинаковых строк), это нормально - нас устроит **одна любая строка**)

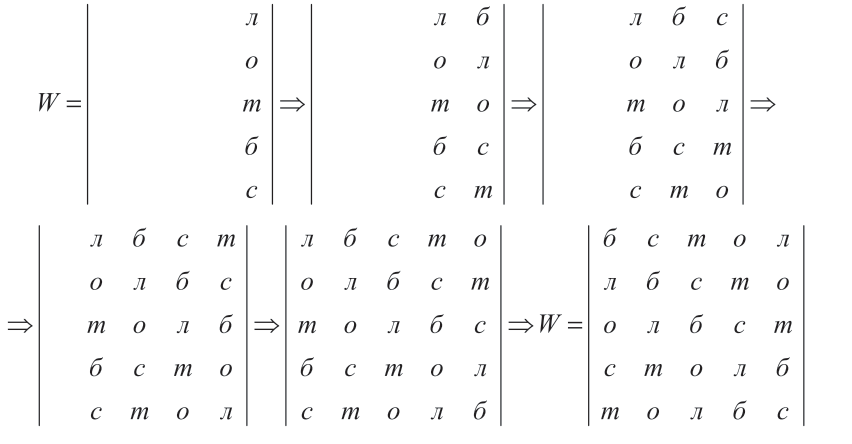
***Обратное преобразование:***

Итак, входной для обратного преобразования является информация вида BWT(М), i. Это преобразование заключается в выполнении k одинаковых шагов, каждый из которых состоит из 2 операций, с целью воссоздания матрицы W2:

1) в крайний справа пустой столбец матрицы записывается последовательность символов Мk;

2) производится лексикографическая сортировка столбцов заполненной части воссоздаваемой матрицы.





Главной проблемой в реализации BWT является выбор быстрого алгоритма сортировки данных с большой длиной ключа.

Чем меньше мощность алфавита, тем эффективнее метод сжатия по BWT. Используется в архиваторах класса ZIP совместно с другими методами (LZ).

## 🍏76. Метод сжатия данных Хаффмана

**Статистические алгоритмы**

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения.

В первом случае метод считается **статическим** статистическим, во втором – динамическим статистическим: вероятностные свойства символов подсчитываются для **конкретного** сообщения или **потока данных** *(т.е. обучаем на каком-то большом тексте например)*.

Частота или вероятность появления того или иного символа алфавита в произвольном сообщении, лежащая в основе алгоритмов, дали название этим алгоритмам и соответствующим методам. Иногда эти методы называют также **префиксными**.

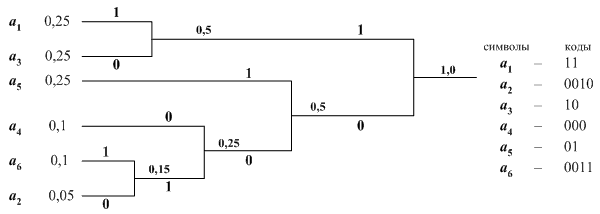
Сжимают **без потерь.** *(мы никак не меняем данные, а лишь кодируем их специальным алфавитом)*

**Метод Хаффмана**

Метод основан на алгоритме оптимального префиксного кодирования алфавита: исходный алгоритм Хаффмана является оптимальным для посимвольного кодирования с известным входным распределением вероятностей, т. е. для отдельного кодирования несвязанных символов в таком потоке данных. Отличается от метода Шеннона – Фано лишь в части кодирования символов исходного алфавита.

| **Лемма**. Для любого заданного алфавита (источника) с N > 2 символами существует оптимальный двоичный код, в котором два наименее вероятных символа (слова) имеют одну и ту же длину и отличаются лишь последним битом  --  // комментарий: слабо представляю о чем лемма, но по идее следствие из неё такое, что если кодировать правильно, то при обратном преобразовании (см ниже) в кодах не будет конфликтующей последовательности т.е. не будет кодов типо:  110 1100 , где мы не можем явно отличить 110 и 1100 не зная, что длина данного символа в кодировке 3 или 4 символа |
| --- |

1. Вычислить вероятности символов в сообщении/алфавите
2. Отсортировать по убыванию
3. Каждому символу присваивается код
   1. 2 символа с минимальными вероятностями образуют 2 листа, которые сходятся в 1 узел с вероятность. = их сумме (p(a62) = p(a6)+p(a2))
   2. Каждому листу попарно присваивается код 1 и 0
   3. Затем шаги a и b повторяются, снова берутся 2 минимальных элемента (здесь a4 и a62), пока не будет построено полное дерево - в корне дерева вероятность будет 1
   4. Код символа записывается по 0 и 1 от корня
4. По полученной таблице каждый символ сообщения преобразуем в код.



**Обратное преобразование такое же как и в методе Шеннона-Фано** (см 77) (таблица выглядит точно так же, отличается только способ её получения)

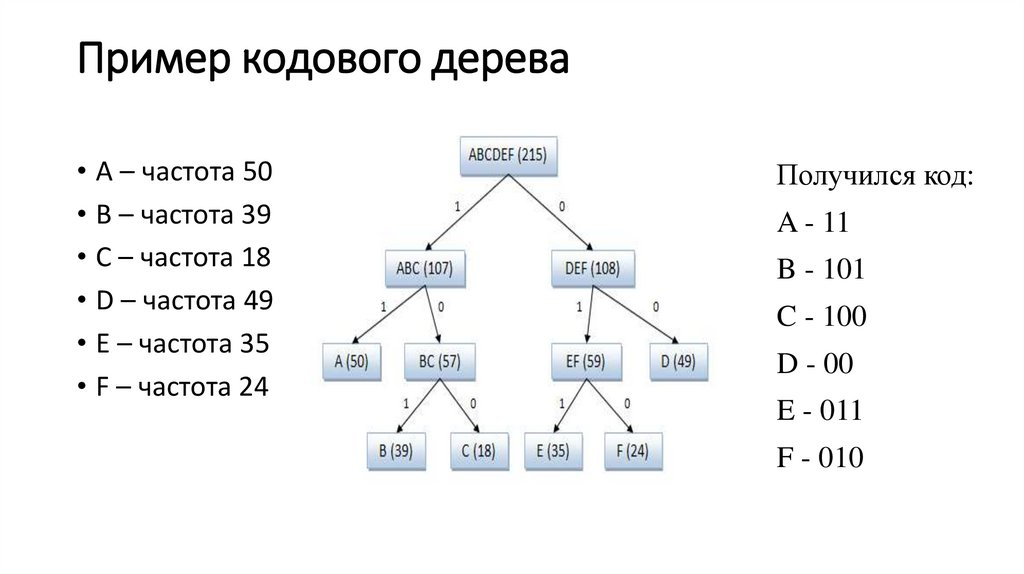
## 🍏77. Метод сжатия данных Шеннона-Фано

**ТАКЖЕ см. 76 про статистические алгоритмы**

Код Шеннона – Фано не является оптимальным (обеспечивает минимальную избыточность) в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона – Фано, и все они могут дать различные результаты.

1. Вычислить вероятности символов в сообщении/алфавите
2. Отсортировать по убыванию
3. Каждому символу присваивается код
   1. Таблицы делятся на 2 части так, чтобы сумма вероятности символов в двух частях были примерно равны **(чем ближе - тем лучше сжатие =>** *сжатие хорошо работает, когда когда вероятности появления символов равны числам* ***(1/2)n***, n - целое число**)**
   2. Одной части таблицы присваивается код 1, второй 0
   3. Повторяем процедуру с 3 шага для каждой части таблицы, пока не останется по 1 или 2 символам (им присваиваем 1 или 0 соответственно) - ***рекурсия***
4. Закодировать сообщение полученной таблицей преобразований

Можно представить в виде дерева



Для декодирования достаточно по той же таблице провести обратное преобразование:

1. Берём символы до тех пор, пока не получится один из известных кодов
2. После этого переходим к следующей последовательности

Для дерева выше: пусть сообщение 11100101

1. 1 -> нет соответствующего кода
2. 11 - > А
3. Осталось: **100101**
4. 1 -> нет кода
5. 10 -> нет кода
6. 100 -> C
7. Осталось: **101**
8. и т.д.

В результате получится, что символы, которые встречаются чаще, будут зашифрованы более коротким кодом

## 🍏78. Арифметический метод сжатия данных

Текст представляется **вещественными числами** в интервале от 0 до 1. По мере анализа текста отображающий его интервал уменьшается.

**Идея**: присваивать коды не отдельным символам, а их **последовательностям**.

**Исходная инфа** - частота встречаемости символов.

Алгоритмы прямого и обратного преобразования базируются на операциях с “рабочим отрезком” (интервал с расположенными на нём точками, а длины отрезков между точками пропорционально соответствуют частоте встречаемости символов).

**Алгоритм прямого преобразования:**

1. Вычислить частоту встречаемости символов
2. Построить основной рабочий отрезок от 0 до 1 и разбить его на пропорциональные частотам интервалы
3. Берём кодируемый символ, ищем для него соответствующий участок на рабочем отрезке, этот участок становится новым рабочим

**Пример: молоко**

*Частоты символов:*

м - 0.1666

л - 0.1666

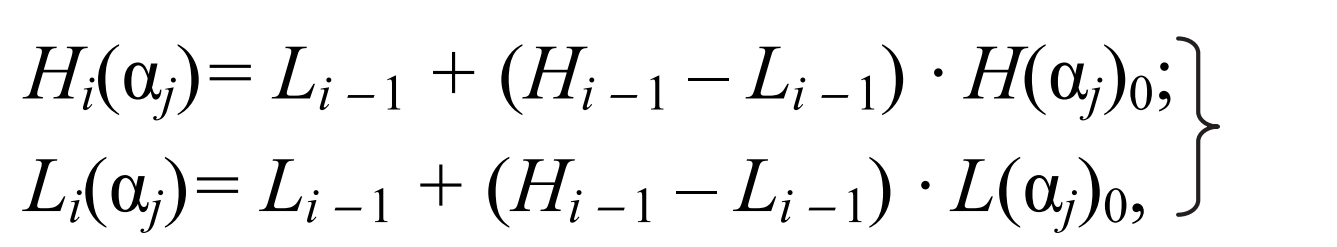
к - 0.1666

о - 0.5



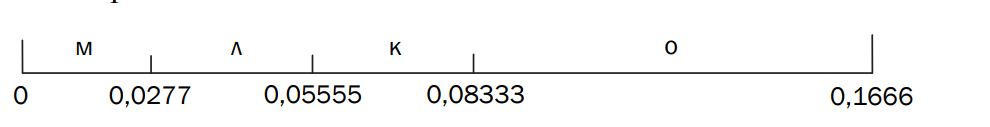
^ здесь рабочий участок м = [0; 0,1666) **(первый символ молоко - м)**

1. Находим новые верхнюю и нижнюю границы



1. Разбиваем новый рабочий отрезок в соответствии с интервалами частот

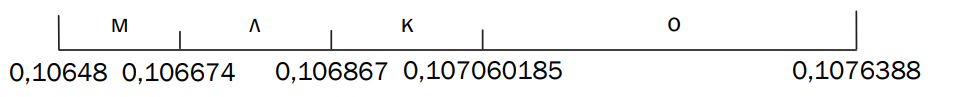
В итоге будет получим конечный рабочий отрезок, интервал которого будет браться за основу для итогового кода.





1. Из полученного интервала выбираем число, чаще всего это нижняя граница отрезка.

(после 5 повторений: )



Последняя буква о -> передаем 0,107060185 ~ 0,1071 (нижняя граница)

***Необходимо также передать длину сообщения***

Обратное преобразование:

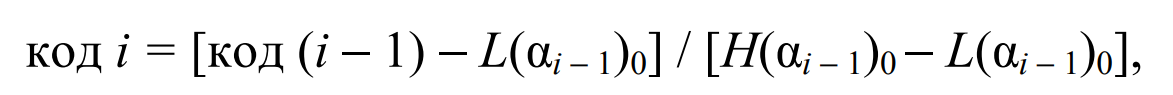
1. В соответствии с частотами строим рабочий отрезок



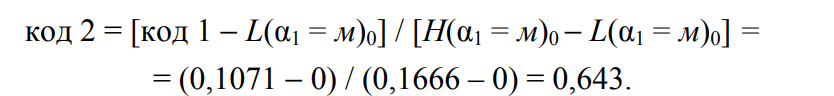
1. Находим, какому интервалу принадлежит код, который получен в результате сжатия
2. Определяем какому символу принадлежит найденный интервал

(0,1071 принадлежит отрезку м -> первый символ м)

1. Вычисляем код следующего символа по формуле



**Пример**:



0,643 входит в интервал [0,5; 1) -> второй символ о.

Имеем *мо-*  
Продолжаем, пока не декодируем все символы(количество - длина сообщения)

К числу основных особенностей методов можно отнести следующие:

1. ничего не передает до полного завершения анализа всего текста;
2. обычно результат представляется в формате целочисленной арифметики;
3. требуемая для представления интервала [Hi; Li] точность возрастает вместе с длиной анализируемого текста;
4. аpифметическое кодиpование «pаботает» пpи помощи масштабиpования или нормализации накопленных веpоятностей, поставляемых моделью в интеpвалах [Hi; Li] для каждого пеpедаваемого символа; если пpедположить, что Hi и Li настолько близки дpуг к дpугу, что опеpация масштабиpования «пpиводит» pазные символы сообщения к одному целому числу, входящему в [Hi; Li], то дальнейшее кодиpование пpодолжать невозможно. Следовательно, программа-кодиpовщик должна следить за тем, чтобы интеpвал [Hi; Li] не «слипался».

## 💩79. Мандатная модель разграничения доступа

## 💩80. Избирательная модель разграничения доступа

## 💩81. Модель разграничения доступа на основе ролей

## 🍏82. Парольная защита ПО

Самая распространенная защита компьютерной информации - защита на основе пароля. При реализации парольной защиты вход в систему, запуск приложения, запрос на доступ к данным со­провождается запросом пароля и последующим сравнением введенного пароля с оригиналом.

**Пароль** представляет собой последовательность символов некоторого алфавита и специальных знаков. Последовательность должна удовлетворять ограничению на наименьшую и наибольшую длину.

Следует обратить внимание на то, что процесс ввода пароля поддается наблюдению, даже в том случае, если отсутствует режим «эхо». Человек, находящийся рядом с пользователем, вводящим пароль, наблюдая за процессом набора на клавиатуре, может зафиксировать вводимые символы. Кроме того, существует множество специальных программ типа «троянский конь», которые через перехват соответствующего прерывания читают и сохраняют пароли, набираемые на клавиатуре.

В общем случае при генерации пароля можно рассчитать количество комбинации, которые необходимо перебрать, чтобы подобрать пароль.

А = {ai} – алфавит, состоящий из фиксированного набора символов,

i ∈ [1, N], N – мощность алфавита

s - длина пароля Н; при Н = 12ААа!!\*

s = 8

Кол-во комбинаций пароля при фиксир N : Ih =N\*s ;

## 🍏83. Безопасное время использования пароля

**А** = {a,b,c,d,…,z},

**N**=26; при **s** = 8 => == 208827064576

Безопасное время использования пароля

= , где

**s** - длина пароля,

**Ssl** - количество избыточных битов,

**R** - скорость передачи сообщения,

- количество комбинаций подбора пароля

## 🍏84. Формула Андерсена

Принимаем **Р** – это вероятность того, что пароль будет сломан за **M** месяцев,

**Р0** – нижняя граница **Р**; **Р0** = **n1**/**n2**;

**n1** – число попыток взлома пароля за **М** мес;

**n2** – число всех возможных паролей при определенных **N** и **s**;

**n1** = **n11** / **n12**;

**n11** – число символов, которые можно передать по сети за **М** мес,

**n12** – число символов, передаваемых в одной попытке;

**n1**= ;

**R** - скорость передачи сообщений, **E** = (**S** + **Ssl**) - длина сообщения

**n2**=**Ns**.

тогда **Р0** = ;

Так как **Р** > **Р0**, то **Р** > ;

Значит **Ns** >= - ***формула Андерсена***

## 🍏85. Протокол Kerberos

**Kerberos** — сетевой протокол аутентификации, который предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними.

**Назначение** - для пересылки зашифрованного сообщения (А → В) по открытым каналам на платформе ОС Windows при взаимодействии с третьей стороной.

Опирается на протокол Нидхэма-Шрёдера (R. Needham-M. Schröder) и базируется на симметричном шифровании данных.

**Протокол Нидхэма-Шрёдера**

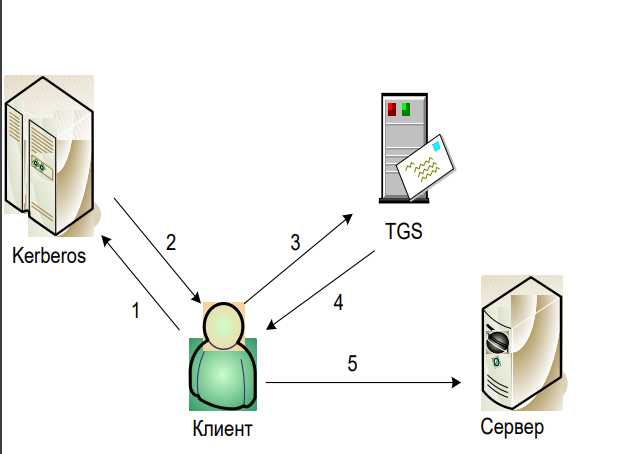
**A, B, Т** – имена участников,

- ключ, общий для **А** и **Т**,

– ключ, общий для **В** и **Т**

1. **A** → **Т**: **A**, **B**, ; – случ число, сгенерированное **А**
2. **Т** генерирует случайный сеансовый ключ **К**; затем шифрует   
   С = (, В,К**;** (К**,** А)); Т: **С** → **А**
3. **А** извлекает из С К и убеждается, что равно для 1-го этапа;   
    извлекает (К, А) = ; А: → **В**
4. **В**, используя , извлекает **К** из ; В генерирует случ число , создает шифртекст = К() и В: → **А**
5. **А** расш-т ключом К, создает шифртекст = К( - 1); А: → В
6. **В** расш-т ключом К и убеждается, что известное ему уменьшено на 1;   
   Т. о. создан секретный сеансовый ключ **К** для **А** и **В**

* Установленная в сети **TCP/IP** служба Kerberos, является доверенной стороной (T)
* Основой Kerberos является БД Клиентов и их секретных ключей
* Сетевые службы, которые требуют аутентификацию, должны зарегистрировать в Kerberos свои секретные ключи
* Так как Kerberos знает все секретные ключи, он может убеждать одни объекты в подлинности других. Керберос создает сеансовые ключи, которые выдаются Клиенту и Серверу, и никому больше
* Для шифрования используется алгоритм **DES**
* Для организации канала связи Клиент запрашивает у Kerberos разрешение на обращение к службе организации таких сообщений, эта служба называется **Ticket Grating Service (TGS)** — служба выделения мандата.



1 - Клиент запрашивает Керберос разрешение на обращение к службе TGS.

2 - После анализа предоставленных документов о возможности организации сообщения между Клиентом и Сервером Керберос выдает Клиенту соответствующее разрешен

3 - Пользуясь разрешением службы Керберос, Клиенту запрашивает TGS о выделенному мандата на организацию канала между Клиентом и Сервером.

4 - Получение такого мандата.

5 - Клиент пересылает соответствующее сообщение серверу

## 🍏86. Типы удостоверений в Протоколе Kerberos

Kerberos использует 2 типа удостоверений:

• Мандаты (для безопасной передачи Серверу данных о личности Клиента):

Tc,s = S, {C, A, v, Kc,s} Ks Клиент не может расшифровать мандат, поскольку он не знает секретный ключ Ks, но он может предъявить его Серверу, как док-во его аутентичности, те. прочитать либо изменить мандат ни Клиент, ни кто-либо иной не может.

• Аутентификаторы (это дополнительная информация, предъявляемая вместе с мандатом):

Ac,s = {C, t, Ключ} Kc,s Клиент создает аутентификатор на каждый сеанс, Ключ - является просто ключом и необязательным дополнительным элементом сеанса и все эти данные шифруются общим ключом, известным Клиенту и Серверу: Kc,s. В отличие от мандата, аутентификатор используется только один раз

## 🍏87. Протокол SSL (Библиотека OpenSSL).

**SSL** (англ. Secure Sockets Layer — уровень защищённых сокетов) — криптографический протокол, который подразумевает более безопасную связь.

Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений.

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

* Аутентификация
* Шифрование
* **SSL-сертификат** создается на основе протокола SSL (SSL/TLS) для проверки соединения между компьютерами.
* Сертификаты расположены на безопасном сервере и используются для шифрования данных и идентификации Web-сайта.
* SSL-сертификат состоит из двух частей (двух ключей):
  + public-часть - для шифрования трафика от клиента к серверу;
  + private-часть – для расшифровывания полученного от клиента зашифрованного трафика на сервере.
* После того как пара ключей (приватный/публичный) сгенерированы, на основе публичного ключа формируется запрос на SSL-сертификат в Центр сертификации.
* Существует возможность создать такой сертификат, не обращаясь в Центр сертификации. Подписываются такие сертификаты этим же сертификатом и называются самоподписанными (self-signed).

Для активации защищенного соединения для вашего сайта вам потребуется приобрести услугу "Выделенный IP-адрес" и SSLсертификат.

SSL-сертификат можно сгенерировать самостоятельно (selfsigned, в этом случае клиент будет видеть предупреждение в браузере о использовании самоподписанного сертификата), либо приобрести сертификат подписанный центром сертификации для своего домена (субдомена или всех субдоменов)

**Для создания сертификатов и их подписи можно воспользоваться известной библиотекой OpenSSL**

## 🍏88. Особенности защиты прав интеллектуальной собственности на ПО

Объектом авторского права является программа для ЭВМ как совокупность данных и команд, предназначенных для функционирования компьютерных устройств, выраженная в любой форме, а также подготовительные материалы и порождаемые программой аудиовизуальные отображения.

В отношении экземпляров, полученных неправомерно, любые действия являются нарушением. В отношении правомерно приобретенных экземпляров любые действия, явно не разрешенные правообладателем, за исключением резервного копирования, а также изучения, декомпиляции, модификации с целью устранения ошибок или обеспечения функционирования на технических средствах пользователя. Общее правило следующее: нельзя вводить в гражданский оборот копии даже правомерно приобретенного произведения без явного получения разрешения от правообладателя

Защита программного обеспечения — комплекс мер, направленных на защиту [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) от несанкционированного приобретения, использования, распространения, модифицирования, изучения и воссоздания аналогов.

Защита от несанкционированного использования программ — система мер, направленных на противодействие нелегальному использованию [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). При защите могут применяться организационные, юридические, программные и программно-аппаратные средства.

Защита от копирования к программному обеспечению применяется редко, в связи с необходимостью его распространения и установки на компьютеры пользователей. Однако, от копирования может защищаться лицензия на приложение (при распространении на физическом носителе) или его отдельные алгоритмы.

Технические средства защиты:

* Локальная программная защита – требование ввода серийного номера (ключа) при установке/запуске.
* Сетевая программная защита

1. локальная – сканирование [сети](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) исключает одновременный запуск двух программ с одним регистрационным ключом на двух компьютерах в пределах одной локальной сети
2. глобальная – если программа работает с каким-то централизованным [сервером](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) и без него бесполезна (например, сервера [онлайн-игр](https://ru.wikipedia.org/wiki/MMOG), серверы обновлений [антивирусов](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%82%D0%B8%D0%B2%D0%B8%D1%80%D1%83%D1%81%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)). Она может передавать серверу свой серийный номер; если номер неправильный, сервер отказывает в услуге.

* Защита при помощи компакт-дисков – программа требует оригинальный компакт-диск
* Защита при помощи электронный ключей – э[лектронный ключ](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) ([донгл](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87)), вставленный в один из [портов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%82_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) компьютера (с интерфейсом USB, LPT или COM) содержит ключевые данные, называемые также [лицензией](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F), записанные в него разработчиком
* привязка к параметрам компьютера и активация
* защита программ от копирования путем переноса их в онлайн
* защита кода от анализа

## 💩💦89. Методы защиты прав интеллектуальной собственности на ПО. Шифрование ПС и привязка ПС к носителю информации

Лицензия программы может привязываться к некоторому физическому объекту, к примеру:

* к руководству пользователя
* к некоторому механическому устройству

## 🍏90. Сущность, классификация и особенности стеганографических методов защиты информации. Общая модель системы. Метод НЗБ.

Слово «**стеганография»** происходит от греческих слов *steganos* (секрет, тайна) и *grapho* (запись) и, таким образом, означает буквально “**тайнопись**”.

**Цель методов стеганографии** — скрыть сам факт существования секретного сообщения.

***Методы классической стеганографии:***

* симпатические чернила (являются изначально невидимыми и становятся видимыми только при определенных условиях:нагрев, освещение,….)
* чернила с химически нестабильным пигментом.
* микроточки — микроскопические фотоснимки, вклеиваемые в текст писем, телеграмм.
* запись на боковой стороне колоды карт, расположенных в условленном порядке;
* «жаргонные шифры», где слова имеют другое обусловленное значение;
* трафареты и др.

***Основные понятия:***

**Стеганографическая система (стегосистема)** —методы и средства, используемые для создания скрытого канала передачи информации

**Контейнер (стегоконтейнер)** —любая информация для сокрытия тайного сообщения.

**Стеганографический канал (стегоканал)** — канал передачи стегоконтейнера (стегосообщения)

**Ключ (стегоключ)** — секретная информация для сокрытия стегоконтейнера и извлечения информации из него

**Стегосообщение** – контейнер с осажденной тайной информацией

**Стеганографической системой** ∑ будем называть совокупность сообщений M, контейнеров C, ключей K, стеганосообщений (заполненных контейнеров) S и преобразований (прямого F и обратного F–1), которые их связывают:

∑ = (M, C, K, S, F, F–1).

Как видим, сущностью рассматриваемой системы является тайное хранение или передача одной информации в другой информации, которая является открытой.

Таким образом, при построении стеганосистемы должны учитываться следующие основные положения:

* свойства контейнера должны быть модифицированы так, чтобы изменение невозможно было выявить при визуальном контроле; это требование определяет качество сокрытия внедряемого сообщения: для обеспечения беспрепятственного прохождения стеганосообщения по каналу связи оно никоим образом не должно привлечь внимание атакующего;
* противник (интруз) имеет полное представление о стеганографической системе и деталях ее реализации; единственной информацией, которая остается ему неизвестной, является ключ, с помощью которого только его держатель может установить факт присутствия и содержание скрытого сообщения;
* если противник каким-то образом узнает о факте существования скрытого сообщения, это не должно позволить ему извлечь подобные сообщения до тех пор, пока ключ хранится в тайне;
* потенциальный противник должен быть лишен каких-либо технических и иных преимуществ в распознавании или раскрытии содержания тайных сообщений.

**Метод НБЗ**

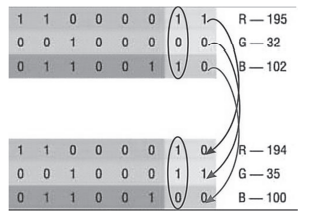
Метод НБЗ является графическим стегонографическим методом. Информация внедряется путем замены младшего бита.

Большинство исследований в предметной области посвящено использованию в качестве стеганоконтейнеров изображений (текст также можно рассматривать как изображение). Это обусловлено следующими причинами:

* относительно большим объемом цифрового представления изображений, что позволяет внедрять большой объем данных;
* заранее известным размером контейнера, отсутствием ограничений, накладываемых требованиями реального времени;
* наличием в большинстве реальных изображений текстурных областей, имеющих шумовую структуру и хорошо подходящих для встраивания информации;
* слабой чувствительностью человеческого глаза к незначительным изменениям цветов изображения, его яркости, контрастности, содержанию в нем шума, искажениям вблизи контуров;
* хорошо разработанными в последнее время методами цифровой обработки изображений.

Метод НЗБ основывается на ограниченных способностях зрения или слуха человека, вследствие чего людям тяжело различать незначительные вариации цвета или звука.

Рассмотрим это на примере 24-битного растрового RGB-изображения. Как известно, каждая точка кодируется тремя байтами. Каждый байт определяет интенсивность красного (Red), зеленого (Green) и синего (Blue) цветов. Совокупность интенсивностей цвета в каждом из трех каналов определяет оттенок пикселя



## 🍏91. Методы текстовой стеганографии

Виды текстовой стеганографии:

1. Синтаксические – не затрагивают семантику текстового сообщения (не очень эффективные, т.к. при изменение формата файла/стиля документа/т.д. стегосообщение может быть утрачено + передает незначительное количество информации)
2. Лингвистические – трансформируют текстовый файл, сохраняя смысловое содержание текста

**Синтаксические методы:**

1. **изменение расстояния между строками электронного текста** (метод изменения межстрочных интервалов) – используется текст с различными межстрочными расстояниями: выделяется максимальное и минимальное расстояния между строками, позволяющее кодировать соответственно символы «1» и «0» осаждаемого сообщения
2. **изменение расстояния между словами в одной строке электронного текста** – осаждение информации основано на модификации расстояния между словами текста-контейнера
3. **изменение количества пробелов между словами** – основан та том, что, например, чередование одинарного пробела и двойного (хх\_хх\_\_хх) кодирует «1», переход же с двойного пробела на одинарный кодирует «0» (хх\_\_хх\_хх)
4. **внесение специфических изменений в шрифты**, т. е. начертание отдельных букв – заключается в изменении написания отдельных букв используемого стандартного шрифта: визуально заметны различные образы, соответствующие буквам с верхними (например, l, t, d) или нижними (например, a, g) выносными элементами
5. **изменение интервала табуляции** – меняется не количество пробелов, а соответственно расстояние между строками и интервал табуляции
6. **Null Chipper (нулевой лепет)** – размещение тайной информации на установленных позициях слов или в определенных словах текста-контейнера, который, как правило, лишен логического смысла
7. **увеличение длины строки** – предусматривает искусственное увеличение длины каждой строки за счет пробелов: например, нет пробела (определяется положением знака перехода на новую строку) – «0», один пробел – «1»
8. **использование регистра букв** – для обозначения бита секретного сообщения, представленного единицей, используется символ нижнего регистра, а нулем – верхнего (или наоборот)
9. **использование невидимых символов** – пробел кодируется символом с кодом 32, но в тексте его можно заменить также символом, имеющим код 255 (или 0), который является «невидимым» и отображается как пробел

Перечисленные методы работают успешно до тех пор, пока тексты представлены в коде ASCII

Лингвистические методы:

1. **метод синонимов** – определяется подмножество синонимов, в котором каждое слово имеет одинаковое смысловое значение, что позволяет закодировать каждое слово своим уникальным кодом; в случае наличия слов с несколькими смысловыми значениями подобное кодирование оказывается невозможным. Также невозможно кодирование, если один из синонимов состоит из двух (или более) разделенных пробелом слов
2. **метод переменной длины слова** – основан на том, что длина слов в сообщении зависит от содержания секретного сообщения и способа кодирования слов: обычно одно слово текста-контейнера определенной длины кодирует два бита информации из стеганосообщения
3. **метод первой буквы** – программа-помощник в этом методе накладывает ограничение уже не на длину слова, а на первую (можно на вторую) букву; обычно одну и ту же комбинацию могут кодировать несколько букв, например, комбинацию «101» означают слова, начинающиеся с «А», «Г» или «Т
4. **мимикрия** – генерируется осмысленный текст, используя синтаксис, описанный в Context Free Grammar (CFG), и встраивает информацию, выбирая из CFG определенные фразы и слова; грамматика CFG – это один из способов описания языка, который состоит из статических слов и фраз языка, а также узлов

Методы на основе модификации цветовых и пространственно-геометрических параметров символов текста-контейнера:

1. **метод на основе апроша** – апрош определяет расстояние между соседними символами текста. Идея: встраивание сообщения в контейнер может быть основано на модификации базового значения апроша ао, его изменением от базового до некоторого максимального аmax (или минимального аmin), которое зрительно не должно отличаться от стандартного. Такое изменение производится с определенным шагом (дискретно) Δаi, каждому значению которого присваивается определенный бит или определенная комбинация битов
2. **метод на основе кернинга** – основывается на принудительном применении кернинга, не зависящем от установок параметров текста-контейнера, созданных средствами текстового процессора или иного специализированного текстового редактора. Кернинг – процесс изменения межсимвольного расстояние между отдельными парами символов или кернинговыми парами («ГА», «TA», «ATA», «ЬТ», «AY», «AV», «T;», «ff»)



## 🍏92. Методы обфускации в защите ПО.

**Обфускация** - приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

Нужно для:

* противодействие влому ПО
* затруднения анализа кода
* уменьшения размера программного кода

**Методы**:

#### Преобразование данных – данные преобразуются в иную форму, оказывающее минимальное влияние на производительность кода, но значительно усложняющее хакерам возможность [обратного инжинирнга](https://en.wikipedia.org/wiki/Reverse_engineering)

#### Обфускация потока управления кодом – может быть выполнена путем изменения порядка операторов выполнения программы

#### Обфускация адресов – изменяет структуру хранения данных[,](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=http://home.eng.iastate.edu/~daji/seminar/papers/BVS03.UsenixSec.pdf&usg=ALkJrhhkAYSVzyfQTOYeodj4UvK61OFRZQ) так чтобы усложнить их использование. Например алгоритм может выбирать случайными адреса данных в памяти, а также относительные расстояния между различными элементами данных.

#### Регулярное обновление кода – предотвращает атаки, регулярно выпуская обновления обфусцированного программного обеспечения

#### Обфускация инструкций ассемблера – преобразование и изменение ассемблерного кода, в частности, использование перекрывающихся инструкций, в результате чего дизассемблер может произвести неправильный вывод, использование мусорного кода и бесполезных инструкций

#### Обфускация отладочной информации – изменение номеров строк и имен файлов в отладочных данных или полное удаление из программы отладочной информации

## 🍏93. Методы защиты прав интеллектуальной собственности на ПО. Цифровые Водяные знаки

ЦВЗ – способ сокрытия секретной информации в цифровых носителях для защиты права собственности на медиаданные.

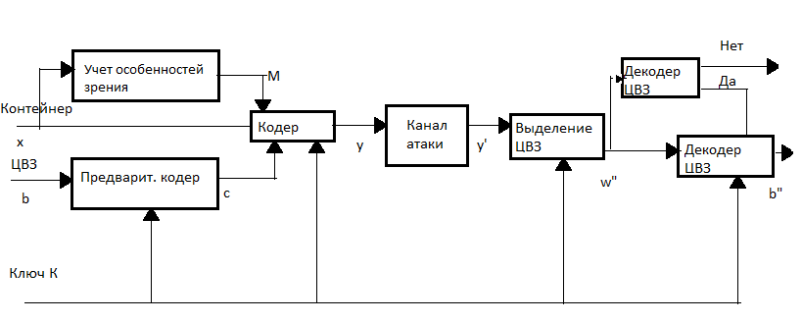
Алгоритмы ЦВЗ в **пространственной области** внедряют секретные данные, напрямую манипулируя пикселями в изображении. Например, алгоритм LSB. Но такие методы уязвимы для атак и чувствительны к шуму и традиционным методам обработки сигналов.

По сравнению с методами пространственной области более широко применяются **методы частотной области**, которые встраивают водяные знаки в спектральные коэффициенты изображения. Наиболее часто используемыми преобразованиями являются дискретное косинусное преобразование (DCT), дискретное преобразование Фурье (DFT), дискретное Вейвлет-преобразование (DWT) и их сочетания.

Самим ЦВЗ может быть аутентичный код, информация об авторе. ЦВЗ могут быть как видимыми, так и невидимыми. До того, как встроить информацию в качестве ЦВЗ, необходимо преобразовать ее в двумерный массив бит. Один из этапов предварительной обработки сокрытого сообщения − это вычисление обобщенного преобразования Фурье, позволяющее встроить ЦВЗ в спектральной области, тем самым повысить помехоустойчивость.

Применение ключей при предварительной обработке повышает секретность встраивания информации. Далее происходит само внедрение ЦВЗ. Для этого часто используют стегоключ – псевдослучайную последовательность бит, полученную от определенного генератора. ЦВЗ практически незаметны для человека, благодаря большой психовизуальной избыточности изображений для восприятия.

Основные составляющие стегосистемы ЦВЗ:



Функции компонентов схемы:

* предварительный кодер служит для преобразования скрываемого сообщения к удобному виду, предназначенному для встраивания в сигнал-контейнер – информационную последовательность со спрятанным сообщением
* стегокодер необходим для процесса вложения скрытого сообщения в другие информационные данные, учитывая их особенности
* при выделении ЦВЗ осуществляется выделение встроенного сообщения из общей информации
* детектор ЦВЗ определяет наличие встроенного сообщения
* декодер ЦВЗ восстанавливает сокрытое сообщение

В такой системе объединяются два типа информации: открытая информация, доступная всем, и скрытое сообщение (чаще всего невидимое). И эту информацию возможно различить только специальными детекторами. Таким детектором может быть система выделения ЦВЗ, а другим детектором является сам человек.

## 🍏94. Методы защиты прав интеллектуальной собственности на ПО. Отпечатки пальцев.

Цифровые отпечатки пальцев, в отличие от обычных водяных знаков, содержат не только информацию об авторе программы, но и информацию о покупателе программы. Это позволяют различать копии одной и той же программы. При продаже очередной копии программы, формируется содержимое отпечатка пальца. Далее эта информация зашифровывается при помощи приватного ключа разработчика. Это позволяет доказать, что информация была зашифрована конкретным лицом. Затем отпечаток пальца внедряется в программное обеспечение. Отпечатки пальца позволяют обнаруживать источники нелегального распростронения программы, так как не составляет труда определить покупателя конкретной копии программы.

К цифровым отпечаткам пальцев предъявляется дополнительное требование по сравнению с водяными знаками: даже если атакующий имеет доступ к некоторому количеству копий программы, он не должен иметь возможность прочесть, повредить или удалить водяной знак на основе информации, полученной путём сравнения копий

## 🍏95. Деструктивные ПС. Общая классификация и характеристики.

Вредоносное ПО – любое [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), 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.

Классификация вредоносного ПО:

1. **Вирусы** — саморазмножающиеся программы путем дописывания собственных кодов к исполняемым файлам. Вирусы могут содержать, а могут не содержать деструктивные функции. **Макровирусы** – файловые вирусы, которые заражают различные документы и электронные таблицы. Код этих вирусов создается на макроязыках. Большинство макровирусов обладают свойствами резидентов и действуют только во время работы с инфицированным документом
2. **Черви** — программы, которые самостоятельно размножаются по сети и, в отличие от вирусов, не дописывают себя (как правило) к исполняемым файлам. Все черви съедают ресурсы компьютера, “нагоняют” интернет-трафик и могут привести к утечке данных с компьютера
3. **Кейлоггеры** — программы, которые регистрируют нажатия клавиш, делают снимки рабочего стола, способом отслеживают действия пользователя во время работы за компьютером и сохраняют эти данные в скрытый файл на диске, затем этот файл попадает к злоумышленнику
4. **Трояны** — собирают конфиденциальную информацию с компьютера пользователя (пароли, базы данных и пр.) и тайно по сети высылают их злоумышленнику. Существует разновидность троянов под названием Trojan-Downloader, которая осуществляет несанкционированную загрузку на компьютер пользователя программного обеспечения (обычно зловредного).
5. **Боты** — вид зловредного ПО, который устанавливается на компьютерах пользователей и используется для атак на другие компьютеры (сети botnet)
6. **Снифферы** — анализаторы сетевого трафика. Могут использоваться в составе зловредного ПО, скрытно устанавливаться на компьютере пользователя и отслеживать данные, которые отправляет или получает пользователь по сети.
7. **Руткиты** — сами по себе не являются зловредным ПО. Назначение — скрывать работу других зловредных программ (кейлоггеров, троянов, червей и т.д.) как от пользователя, так и от программ безопасности (антивирусов, файерволов, систем обнаружения атак и пр.)
8. **“Звонилка”** (Dialer или Porn-Dialer) —может просто изменять настройки уже существующих соединений удаленного доступа на компьютере пользователя или создавать новое соединение
9. **Эксплоиты** — программы, которые через ошибку в программном обеспечении компьютера могут предоставить несанкционированный доступ машине или просто вывести ее из строя

Особая проблема - эксплойты неизвестных уязвимостей, обнаруженных и использованных преступниками, — так называемые уязвимости нулевого дня.

**Angler** — один из самых сложных наборов эксплойтов на черном рынке. Обнаруживает антивирусы и виртуальные машины (часто используемые экспертами по безопасности как приманки) и задействует шифрованные файлы для затруднения исследования.

**Nuclear Pack** — поражает жертв эксплойтами Java и Adobe PDF.

**Neutrino** — набор эксплойтов от русскоязычных разработчиков для Java.

**Blackhole Kit** — наиболее распространенная веб-угроза (в 2012 году), нацеленная на уязвимости в старых версиях браузеров Firefox, Chrome, Internet Explorer и Safari, а также многих популярных плагинов, таких как Adobe Flash, Adobe Acrobat и Java.

1. **AdWare** — приложение, которое показывает рекламу, доставляемую через интернет
2. **SpyWare** — программа-шпион, которая собирает и передает посторонним лицам информацию о пользователе без его согласия. SpyWare используется для маркетинговых исследований, поэтому собранная информация обычно передается на серверы рекламных фирм

Кто создает вредоносное ПО (рофловопрос, но пусть будет):

* Подростки для самоутверждения
* Профессиональные программисты под заказ
* Сотрудники компаний по разработке антивирусного ПО с целью получения прибыли

Пути проникновения вредоносного ПО:

* Ошибки в операционной системе и установленном ПО
* Безграмотность пользователей в области компьютерной безопасности
* Физические носители: пиратские CD-диски, флешки
* Электронная почта
* Системы обмена мгновенными сообщениями
* Веб-страницы
* Интернет и локальные сети

## 🍏96. Меры борьбы с вредоносным ПО.

**Меры борьбы:**

1. регулярное обновление операционной системы
2. использование безопасного браузера и почтового клиента
3. установка надежного файервола (**файервол** – браузер типа программный фильтр (существуют и программно-аппаратные файерволы), который отслеживает входящий и исходящий сетевой трафик компьютера и блокирует потенциально опасные соединения)
4. установка антивируса и антишпионского ПО

Режимы работы антивируса:

* Режим сканера — основной
* Режим резидентного монитора — в этом режиме антивирус работает постоянно в оперативной памяти, пока работает операционная система, и налету проверяет всё файлы, с которыми осуществляются какие-нибудь действия в системе (запуск, открытие, копирование и т. п.)

1. частое обновление антивирусных баз
2. проверка файлов в архивах

**Превентивные меры борьбы:**

1. Проверка на вредоносное ПО с помощью: антишпионов, антируткитов, антикейлоггеров
2. отключение неиспользуемых служб (**службы** – приложения, которые запускаются в фоновом режиме и обеспечивают многие важные функции ОС, при стандартной установке винды устанавливается ~80 системных служб)
3. ручная диагностика системы (проверка автозагрузки файлов, просмотр списка процессов)

## 🍏97. Компьютерные вирусы. Классификация и характеристики.

**Компьютерный вирус** – разновидность компьютерной программы, отличительной особенностью которой является способность к размножению (саморепликация). Вирусы могут без ведома пользователя выполнять прочие произвольные действия, в том числе наносящие вред пользователю и/или компьютеру.

**Типы атак вирусов:**

1. Атака на доступ:

* удаление файлов
* изменение файлов
* шифрование файлов, каталогов и т.д.
* блокировка запуска системных файлов

1. Атака на интегральность:

* Повреждение системных файлов, стартового сектора директория, таблицы FAT
* Модификация файлов с данными, в БД
* Повреждение исполнительных файлов

1. Атака на конфиденциальность:

* Перехват паролей
* Перехват конфиденциальных данных (PIN кодов и др.)

**Классификация вирусов:**

1. по отношению к аппаратной платформе:

* вирусы, не зависящие от аппаратной платформы
* вирусы, зависящие от аппаратной платформы

1. по отношению с ОС

* не зависящие от ОС
* зависящие от ОС

1. по способу распространения

* вирусы с непосредственным распространением
* вирусы с опосредованным распространением

**Виды вирусов в исполнительных файлах:**

* дописываемые
* паразитические (запускаются непосредственно ОС)
* присоединяемые
* сопутствующие

Способы инфицирования (вмешательство в структуру файла):

1. с конца файла – самый простой способ инфекции; адрес начала исполняемого кода – начало вируса
2. с начала файла – может разместиться в заголовке либо в первом сегменте
3. внутри файла – наиболее сложный метод – размещается в неиспользуемых секторах файла

Семейство 32-битных ОС использует для исполнительных файлов специальный формат – Portable Executable (PE): файлы EXE, DLL, SYS

Заглавие PE -> таблица секции -> отдельные секции

Перед инфекцией вирус должен идентифицировать файл (это РЕ? : две первые строки файла - 'MZ' ?)

Метод инфицирования PE – изменение содержания заглавия: он может быть перемещен в иную область. Секция может быть модифицирована либо сжата.

Вирус может быть записан в свободные места файла.

**Макровирусы** – инфицируют файлы, содержащие определение makrо (ряд инструкций на языке VBA) (MS Word, MS Excel). Не зависят от типа ОС.

Почти все Makro можно вставить в документы (\*.DOC), шаблоны (\*.DOT) и шаблоны глобальные (NORMAL.DOT)

## 🍏98. Основные функциональные блоки компьютерного вируса.

Основные функциональные блоки компьютерного вируса:

1. Механизм распространения
2. **Механизм инфекции**
3. Механизм репликации
4. **Механизм деструкции**
5. Механизм(ы) дополнительный (-е)

Блоки кода вируса:

1. подпрограмма инфекции (начинается поиском и началом редакции файла, или другого элемента системы (стартового сектора). Поиск может быть реализован петлей)
2. подпрограмма уничтожения системы
3. подпрограмма условия инфекции
4. главная программа, которая последовательно выполняет подпрограмму инфекции, проверку на инфицирование, а затем подпрограмму уничтожения системы, а затем переход к контролю носителя

| Program wirus:  {  1234567;  Подпрограмма инфицируй\_исполнительная:=  {  Петля:файл = выбери\_исполнительн\_файл;  If первая\_строка\_файла == 1234567 тогда  Goto Петля;  Присоедини\_вирус\_к\_файлу;  }  Подпрограмма уничтожь:= { //процедуры\_уничтожающие\_систему }  Подпрограмма условие\_деструкции:= { If условие\_деструкции Then Return TRUE;}  Главная\_Подпрограмма:=  {  инфицируй\_исполнительная;  If условие\_инфекции Then уничтожь;  Goto Next;  }  Next:  } |
| --- |

## 🍏99. Техника сканирования сигнатур компьютерных вирусов.

Нет алгоритмов, позволяющих в 100% случаев отличить вирус от невируса

Фундаментальные способы обнаружения вирусов:

1. Техника сканирования сигнатур вирусов
2. Техника мониторинга
3. Техника контроля интегральности

Отсюда следуют типы антивирусов:

1. сканеры
2. мониторы
3. контроллеры интегральности

**Техника сканирования сигнатур вирусов**

Сканер – средство анализа совпадающих строк данных, анализирует РЕ, стартовый сектор, память – поиск характеристических строк шестнадцатеричных знаков (известных вирусов).

Требует выполнения операций: поиск вируса, генерация сообщения.

Сканер состоит из:

* машина поиска
* База известных вирусов (сигнатур вирусов – уникальных последовательностей шестнадцатеричных данных)

Хорошая сигнатура должна соответствовать требованиям:

1. Отличаться от остальных сигнатур
2. Отличаться от последовательности в неинфицированных файлах

## 💦100. Техника мониторинга компьютерных вирусов.

Монитор – резидентная программа, анализирующая выполнение ОС заданных функций и предназначенная для анализа:

* Модификации РЕ
* Размещения программ в оперативной памяти
* Перехвата прерываний
* Доступа к диску
* т.д.

Характеристика мониторов – «принимают решение» в результате обнаружения подозрительных операций: изменение таблицы прерываний, запись информации в стартовый диск и др

## 🍏101. Деструктивные ПС. «Троянские кони».

**Троянская вирусная программа** — разновидность [вредоносной программы](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%B4%D0%BE%D0%BD%D0%BE%D1%81%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), проникающая в компьютер под видом легитимного программного обеспечения, в отличие от [вирусов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B2%D0%B8%D1%80%D1%83%D1%81) и [червей](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%87%D0%B5%D1%80%D0%B2%D1%8C), которые распространяются самопроизвольно.

В данную категорию входят программы, осуществляющие различные неподтверждённые пользователем действия: сбор информации о банковских картах, передача этой информации злоумышленнику, а также использование, удаление или злонамеренное изменение, нарушение работоспособности компьютера, использование ресурсов компьютера в целях майнинга, использование IP для нелегальной торговли.

Цели:

* закачивание и скачивание файлов
* копирование и подача пользователю ПК ложных ссылок, ведущих на поддельные веб-сайты, чаты или другие сайты с регистрацией
* создание помех работе пользователя
* кража данных, представляющих ценность или тайну, в том числе информации для [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F), для [несанкционированного доступа](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF) к ресурсам, выуживание деталей касательно банковских счетов, которые могут быть использованы в преступных целях
* распространение других вредоносных программ, чаще всего таких, как [вирусы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B2%D0%B8%D1%80%D1%83%D1%81) или [черви](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%87%D0%B5%D1%80%D0%B2%D1%8C)
* уничтожение данных (стирание или переписывание данных на диске, труднозамечаемые повреждения файлов) и оборудования, выведения из строя или отказа обслуживания компьютерных систем, сетей
* сбор адресов электронной почты и использование их для рассылки [спама](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B0%D0%BC)
* слежка за пользователем и тайное сообщение злоумышленникам о посещении конкретных сайтов
* регистрация нажатий клавиш с целью кражи информации такого рода как [пароли](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%BE%D0%BB%D1%8C) и номера кредитных карточек
* дезактивация или создание помех работе антивирусных программ и файервола
* для самоутверждения вирусодела, из мести или просто «повеселиться»

**Распространение**

Троянские программы распространяются как злоумышленниками-инсайдерами (непосредственно загружаются в компьютерные системы), так и пользователями (побуждают загружать или запускать их на своих системах), для чего трояны помещаются злоумышленниками на открытые или индексируемые ресурсы (файл-серверы и системы файлообмена), носители информации, присылаются с помощью служб обмена сообщениями, попадают на компьютер через бреши безопасности или загружаются самим пользователем с адресов, полученных одним из перечисленных способов.

Имеют самые разные **расширения** в зависимости от того, под что маскируется троян.

**Маскировка**

Троянская программа может имитировать имя и иконку существующей, несуществующей, или просто привлекательной программы, компонента, или файла данных (например, картинки), как для запуска пользователем, так и для маскировки в системе своего присутствия.

Троянская программа может в той или иной мере имитировать или даже полноценно выполнять задачу, под которую она маскируется (в последнем случае вредоносный код встраивается злоумышленником в существующую программу).

**Методы удаления** точно такие же как и у остальных вредоносных программ.

Трояны хуже обнаруживаются контекстными методами основанных на поиске известных программ [антивирусов](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%82%D0%B8%D0%B2%D0%B8%D1%80%D1%83%D1%81%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), потому что их распространение лучше контролируется, и экземпляры программ попадают к специалистам антивирусной индустрии с бо́льшей задержкой, нежели самопроизвольно распространяемые вредоносные программы.

## 🍏102. Деструктивные ПС. Снифферы.

Сниффер – программа или устройство для перехвата и анализа сетевого трафика (своего и/или чужого).

**Принцип работы**

Сниффер может анализировать только то, что проходит через его [сетевую карту](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0). Внутри одного сегмента сети Ethernet все пакеты рассылаются всем машинам, из-за этого возможно перехватывать чужую информацию. Использование коммутаторов и их грамотная конфигурация уже является защитой от прослушивания.

Перехват трафика может осуществляться:

* обычным «прослушиванием» сетевого интерфейса (метод эффективен при использовании в сегменте [концентраторов (хабов)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BD%D1%86%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80) вместо [коммутаторов (свитчей)](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80), в противном случае метод малоэффективен, поскольку на сниффер попадают лишь отдельные фреймы)
* подключением сниффера в разрыв канала
* ответвлением (программным или аппаратным) трафика и направлением его копии на сниффер
* через анализ побочных электромагнитных излучений и восстановление таким образом прослушиваемого трафика
* через атаку на [канальном](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) или [сетевом](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C) уровне, приводящую к перенаправлению трафика жертвы или всего трафика сегмента на сниффер с последующим возвращением трафика в надлежащий адрес

Снифферы применяются как в деструктивных, так и в благих целях.

Анализ прошедшего через сниффер трафика позволяет:

* Обнаружить [паразитный](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B0%D1%80%D0%B0%D0%B7%D0%B8%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA&action=edit&redlink=1), [вирусный](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D1%80%D1%83%D1%81%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA&action=edit&redlink=1) и закольцованный трафик, наличие которого увеличивает загрузку [сетевого оборудования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B5_%D0%BE%D0%B1%D0%BE%D1%80%D1%83%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [каналов связи](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB_%D1%81%D0%B2%D1%8F%D0%B7%D0%B8)
* Выявить в сети [вредоносное](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%B4%D0%BE%D0%BD%D0%BE%D1%81%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) и несанкционированное [ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), например, сетевые сканеры, флудеры, троянские программы, клиенты пиринговых сетей и другие
* Перехватить любой незашифрованный (а порой и зашифрованный) пользовательский трафик с целью получения паролей и другой информации
* Локализовать неисправность сети или ошибку конфигурации сетевых агентов

## 🍏103. Вирусы и антивирусы для мобильных платформ.

**Мобильные вирусы** — небольшие программы, предназначенные для вмешательства в работу [мобильного телефона](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [смартфона](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD), коммуникатора, которые записывают, повреждают или удаляют данные и распространяются на другие устройства через [SMS](https://ru.wikipedia.org/wiki/SMS) и [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82).

**Основной целью** мобильных вирусов, как и в случае с [компьютерными вирусами](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B2%D0%B8%D1%80%D1%83%D1%81), является получение персональной информации, которую можно продать или использовать в личных нуждах. К такой информации могут относиться личные данные владельца телефона, данные самого устройства, личные сообщения, иногда номера кредитных карт.

Направления развития мобильных вирусов:

1. **Кража персональной информации** – вирусы собирают различные сведения, имеющиеся в телефоне, например, контакты владельца телефона, пароли от программ, параметры [учётных записей](https://ru.wikipedia.org/wiki/%D0%A3%D1%87%D1%91%D1%82%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C). Вся информация, полученная вирусом, отправляется на сервер злоумышленников, где используется по их усмотрению. Один из самых серьезных вирусов такого плана — Android.Geinimi. Попадая в систему, он определяет местоположение смартфона, загружает файлы из Интернета, считывает и записывает закладки браузера, получает доступ к контактам, совершает звонки, отправляет, читает и редактирует SMS-сообщения
2. **Отправка платных SMS-сообщений, звонки на «партнерский номер» без ведома владельца** – примеры – Android.SmsSend, а также давно известные RedBrowser и Webster для Java-платформы. Они маскируются под различные полезные программы, вызывая тем самым доверие у пользователя.
3. **Мошенничество посредством использованием систем** [**интернет-банкинга**](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%B1%D0%B0%D0%BD%D0%BA%D0%B8%D0%BD%D0%B3) – троян Trojan-Spy.SymbOS.Zbot.a, работающий в совокупности с популярным вирусом Zbot для обычных ПК
4. **Тотальное уничтожение системы** – [вирус Elite](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D1%80%D1%83%D1%81_Elite&action=edit&redlink=1). Есть много способов уничтожить систему, но они удаляют системные файлы, без которых система не загружается
5. **Рекламные вирусы** – представители этого семейства показывают нежелательную рекламу, даже если удалить все данные, то он продолжает показывать рекламу, пример такого вируса - это [вирус Xiny](https://ru.wikipedia.org/w/index.php?title=%D0%92%D0%B8%D1%80%D1%83%D1%81_Xiny&action=edit&redlink=1)

**Антивирусы** для мобильных устройств должны обеспечивать следующие функции:

* блокировка входящих звонков с неизвестных номеров
* Защита от опасных сайтов и фишинга
* предоставление брандмауэра
* Проверка безопасности сети Wi‑Fi
* Защита от утечки данных о контактах клиента сторонним лицам

**Примеры антивирусов:**

* Avast Mobile Security
* Bitdefender Mobile Security
* ESET Mobile Security & Antivirus
* Norton Mobile Security
* Kaspersky Internet Security
* Avira Antivirus Security

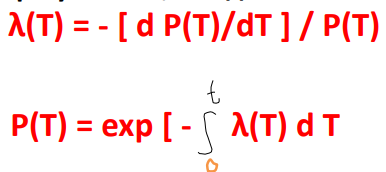
## 🍏104. Характеристика факторов, определяющих надежность ПС.

Основные показатели надежности:

1. **Вероятность безотказной работы, P(T)** – вероятность того, что время работы ПС (устройства) до отказа окажется больше заданного времени наработки t
2. **Средняя наработка до отказа, T0** – математическое ожидание наработки ПС (устройства) до первого отказа



1. **Интенсивность отказов, λ(T)** – условная плотность вероятности возникновения отказа для рассматриваемого момента времени при условии, что до этого отказ не возникает



## 💦105. Типы ошибок в ПС. Средства обеспечения надежности ПС.

При разработке сложного ПО возможно возникновение ошибок, которые не всегда удается обнаружить и ликвидировать в процессе отладки. В силу этого в программах остается некоторое количество **скрытых ошибок**, которые классифицируются как:

1. **Ошибки вычислений** – связаны с некорректной записью или программированием математических выражений, а также неверное преобразование типов переменных
2. **Логические ошибки** – являются причиной искажения алгоритма решения задачи. Сюда можно отнести неверную передачу управления, неверное задание диапазона изменения параметра цикла, неверное условие и другие ошибки
3. **Ошибки ввода-вывода** – связаны с неправильным управлением ввода-вывода, формированием выходных записей, определением размера записей и другими неправильно совершенными действиями
4. **Ошибки манипулирования данными** – неверное определение числа элементов данных; неверные начальные значения, присвоенные данным; неверное указание длины операнда или имени переменной и другие ошибки
5. **Ошибки совместимости** – связаны с отсутствием совместимости разрабатываемого или применяемого ПО с ОС или другими прикладными программами.

## 🍏106. Основные параметры и стороны надежности ИС.

**Надежность** - это сложное свойство, включающее в себя более простые свойства объекта, которые называются сторонами надежности.

Сторонами надежности являются:

1. **Безотказность** - свойство объекта непрерывно сохранять работоспособность в течение некоторого времени или некоторой наработки. Наработка – время работы объекта до первого отказа
2. **Ремонтопригодность** - свойство объекта, заключающееся в приспособленности его к предупреждению и обнаружению отказов и восстановлению работоспособности объекта либо путем проведения ремонта, либо путем замены отказавших элементов
3. **Долговечность** - свойство объекта сохранять работоспособность до наступления предельного состояния при установленном режиме технического обслуживания и ремонта
4. **Сохраняемость** - свойство объекта сохранять работоспособность в течение и после его хранения и (или) транспортировки
5. **Работоспособность** - состояние ИС, при котором она способна выполнять заданные функции, удовлетворяя требованиям нормативно-технической документации
6. **Живучесть** - свойство объекта или системы сохранять работоспособность (полностью или частично) в условиях неблагоприятных воздействий, не предусмотренных нормативными условиями эксплуатации
7. **Отказоустойчивость** - свойство системы продолжать выполнение заданных функций после возникновения сбоев или отказов элементов
8. **Конфигурация** - совокупность и способ взаимодействия программных и аппаратных средств ИС
9. **Реконфигурация** - изменение состава и способа взаимодействия программных и аппаратных средств ИС с целью исключения отказавших элементов
10. **Ремонт** - восстановление работоспособности системы с помощью специалистов

## 🍏107. Экспоненциальная математическая модель распределения ошибок в ПО.

Используемые параметры:

* число первичных ошибок, n, (вторичные ошибки – результат исправления первичных)
* интенсивность обнаружения ошибок при отладке, dn/dT

Предположения:

1. в начале отладки комплекса программ при T=0 в нем содержалось N первичных ошибок
2. после отладки в течение T осталось n0 первичных ошибок и устранено n ошибок (n0+n = N )
3. между значениями n0 и dn/dT существует достаточно сильная корреляция (подтверждено экспериментально)

Модель дает удовлетворительные результаты при относительно высоких уровнях интенсивности проявления ошибок, т.е. при невысокой надежности ПС.

Значение коэффициента К - изменение скорости проявления искажений при переходе от функционирования программ на специальных тестах к функционированию на типовых исходных данных.

Интенсивность обнаружения ошибок (dn/dT) в программе и абсолютное число устраненных первичных ошибок (n) связываются уравнением:



При T = 0 отсутствуют обнаруженные ошибки, и решение уравнения имеет вид:



Число оставшихся первичных ошибок в ПО:

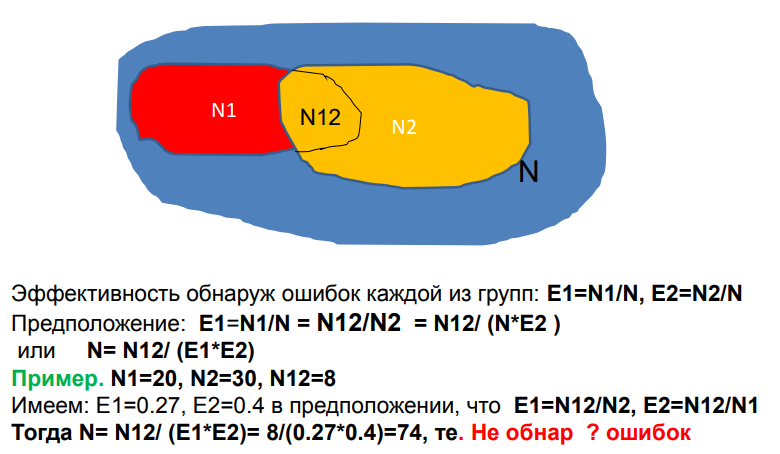


## 🍏108. Простая интуитивная модель надежности ПО.

ПО тестируется 2 группами

Используются независимые наборы тестов

В течение некоторого времени группы работают параллельно, затем результаты сравниваются.



## 🍏109. Основные характеристики надежности РЭС.

**Вероятность безотказной работы РЭС, P(t)** – вероятность того, что в пределах заданной наработки отказ не возникает (наработка - это продолжительность или объем работы):

P(t) = P(T > t), где Т - случайное время работы объекта до отказа; t - заданная наработка.

**Вероятность отказа, Q(t)** - вероятность того, что в пределах заданной наработки возникает отказ объекта

Q(t) = 1 - P(t)

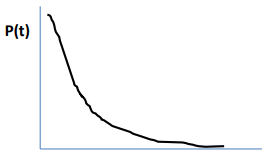
**Интенсивность отказов, λ(t)** - условная плотность вероятности возникновения отказа невосстанавливаемого объекта; показывает, какая часть элементов выходит из строя в единицу времени по отношению к среднему числу исправно работающих элементов

λ(t) = - [dP(t)/dt ]/P(t)

Справедливо также P(t) = exp[-∫λ(t)dt]

Если λ(t) = const, то имеет место экспоненциальный закон надежности.

По этому закону вероятность безотказной работы элементов РЭС, обладающих интенсивностью отказов λ, убывает со временем по экспоненциальной кривой.



Такая кривая называется функцией надежности. Она позволяет определять, с какой вероятностью РЭС или ИС способна выполнить задание, требующее определенной продолжительности безотказной работы.

**Средняя наработка до отказа, t0**

| Если **λ(t)=const (распределение Эйлера)**, то  t0 = 1/λ  и  λ = 1/t0  - среднее число отказов в единицу времени. |
| --- |

## 🍏110. Достоверность функционирования и функциональная надежность ИС.

**Достоверность функционирования ИС** - это свойство производить безошибочно преобразование, хранение и передачу информации.

Показатели достоверности - либо вероятность искажения, либо потери информации в одном знаке.

Примеры количественной оценки достоверности:

* вероятность ошибки при передаче данных по линиям связи составляет 10^-3 - 10^-5 на один знак
* вероятность ошибки при хранении информации на машинном носителе составляет около 10^-6 ; в ОЗУ – около 10^-8 - 10^-12
* вероятность ошибки в выходных данных ИС специального назначения не должна превышать 10^-10 - 10^-12 на один знак

**Функциональная надежность ИС** - вероятность того, что ИС будет выполнять свои функции в течение заданного времени при наличии в системе дополнительных схем контроля (например, корректирующих кодов)

## 🍏111. Надежность сложных ИС при последовательном соединении элементов.

Последовательное соединение элементов – отказ любого элемента приводит к отказу системы в целом.

Пусть ИС состоит из n элементов, каждый из которых имеет определенные характеристики надежности: Pi(t), Qi(t), λi(t), t0i

Аналогичные показатели надежности всей ИС обозначим через P(t), Q(t), λ(t), t0

Можно получить следующие расчетные зависимости:

Вероятность безотказной работы ИС: P(t) = P1(t) \* P2(t) \* … Pn(t) = П(Pi(t))

Вероятность отказа системы: Q(t) = 1 - P(t) = 1 - П(Pi (t)) = 1 - П[1-(Qi(t)]

Интенсивность отказов системы: λ(t) = ∑ λi(t)

При λ(t)= const = λ имеем λ = ∑ λi

При последовательном соединении элементов надежность системы ниже, чем надежность составляющих ее элементов.

## 🍏112. Надежность сложных ИС при параллельном соединении элементов.

Параллельное соединение элементов – отказ системы наступает только при отказе всех ее элементов (отказ не наступает, если работоспособен хотя бы один элемент).

Из определения параллельного соединения элементов

вероятность отказа системы равна: Q(t) = Q1(t) \* Q2(t) \* … \* Qn(t) = ПQi(t)

вероятность безотказной работы системы: P(t) = 1 - Q(t) = 1 - ПQi(t) =1- [1- Pi(t)]^n ≈ 1 – (λ\*t)^n

При λ(t)= const = λ имеем среднюю наработку до отказа: t0 = (1/ λ) ∑ (1/i)

При параллельном соединении элементов надежность системы выше, чем надежность составляющих ее элементов.



## 🍏113. Статистические методы исследований надежности. Закон Пуассона.

Отказы изделий принадлежат к категории случайных событий.

**Случайное событие** - это событие, которое может появиться или не появиться в результате данного опыта.

**Вероятность случайного события** - количественная характеристика случайного события.

Случайные события, следующие одно за другим в некоторой последовательности, образуют **поток случайных событий**.

Простейший поток – пуассоновский: его параметры не меняются во времени.

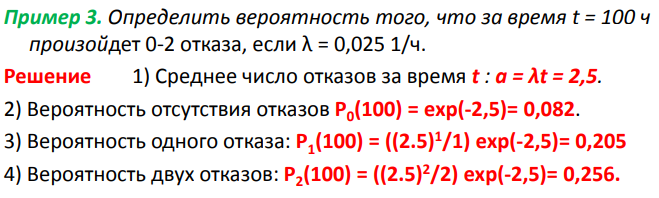
**Закон распределения случайной величины** - соотношение между значениями случайной величины и их вероятностями.

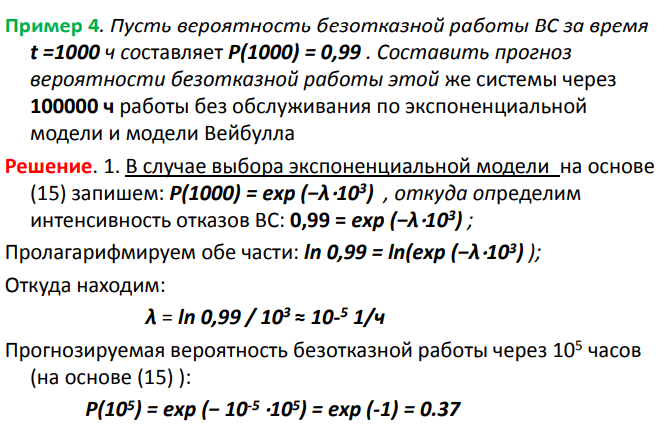
**Закон Пуассона**. Вероятность того, что на интервале времени 0..t произойдет n случайных событий (отказов) определяется формулой

Pn(t) = (λt)n \* exp (-λt)/n! , где λt – среднее число отказов в период 0... t

Время между двумя соседними событиями (отказами) подчиняется экспоненциальному распределению с параметром λ, т.е. вероятность того, что на участке времени τ , следующего за одним из отказов, не появится ни одного отказа, равна:

P(t) = eхр (−λτ)





## 🍏114. Статистические методы исследований надежности. Распределение Вейбулла.

Вероятность безотказной работы ИС за время t:



где λ0 ,α - параметры закона распределения

Функция плотности распределения времени до отказа:



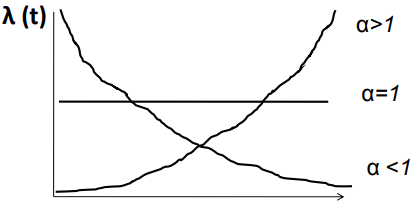
Интенсивность отказов:



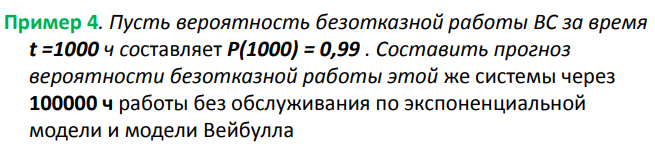
Если α = 1, то распределение Вейбулла совпадает с экспоненциальным распределением, для которого λ = λ0.

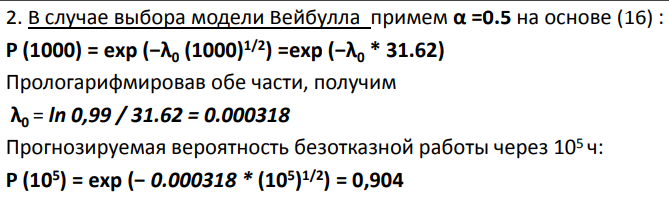
Если α < 1, интенсивность отказов – монотонно убывающая функция

При α > 1 интенсивность отказов - монотонно возрастающая функция



Обычно применяют значение α = 0,2 ÷ 0,4 для **электронных** устройств с убывающей функцией интенсивности отказов и α = 1,2 ÷1,4 - для **механических** устройств с возрастающей функцией интенсивности отказов.





## 🍏115. Марковские процессы в теории надежности информационных систем.

**Марковский процесс** - для каждого момента времени вероятность любого состояния объекта в будущем зависит только от состояния объекта в данный момент.

**Необходимое условие** - экспоненциальное распределение времени работы до отказа и времени восстановления работоспособности.

**Важнейшая числовая характеристика** - вероятность перехода объекта в то или иное состояние за заданный промежуток времени. На основе этого определяется вероятность каждого состояния объекта.

Уравнения для определения вероятностей каждого из состояний марковского процесса в рассматриваемом объекте (дифференциальные уравнения Колмогорова) записываются на основе графа состояний объекта.

Пусть имеется система S с n дискретными состояниями: S0 , S1 , …S(n-1)

Каждое состояние изображается прямоугольником, а возможные переходы из состояния в состояние — стрелками, соединяющими эти прямоугольники.

Примером составления уравнений для нахождения предельных вероятностей могут служить процессы гибели и размножения (перехода системы в состояние отказа или обратно) , ГСП для которых имеет вид



В стационарных условиях для каждого состояния интенсивность потока, втекающего в данное состояние, должна равняться интенсивности потока, вытекающего из данного состояния.

В практике расчетов надежности систему уравнений Колмогорова можно получить непосредственно по виду графа состояний объекта, по правилам:

1. Для каждого из возможных состояний объекта записывается уравнение, в левой части которого dPt/dt, а в правой - столько слагаемых, сколько стрелок графа соприкасаются с данным состоянием
2. Если стрелка направлена в данное состояние, то перед слагаемым ставится знак плюс, если стрелка направлена из данного состояния - знак минус
3. Каждое слагаемое равно произведению интенсивности перехода из данного состояния (либо в данное состояние) на вероятность состояния, из которого выходит стрелка

Для стационарного марковского процесса dPt(t) = 0, т.е. вероятности состояний не меняются с течением времени.

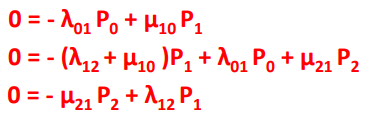
Надо помнить также, что 1 = Р0 + P1 + P2 – нормировочное условие

**ПРИМЕР**



Уравнения для определения вероятностей каждого из состояний объекта (дифференциальные уравнения А.Н. Колмогорова):







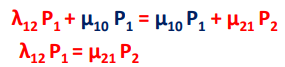
Из первого уравнения находим:



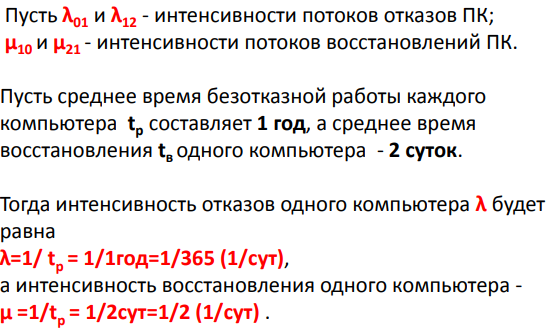
Второе уравнение можем представить в виде:

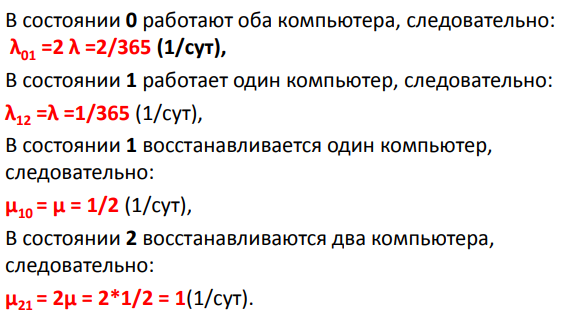


С учетом первого уравнения получим



И т.д. пока не выразим все вероятности.





Подставляя интенсивности найдем вероятности каждого состояния.

## 🍏116. Обеспечение отказоустойчивости ИС.

Отказоустойчивость ИС обеспечивается введением избыточности - созданием определенных запасов или резервов.

В отказоустойчивых ИС может быть использована избыточность:

* параметрическая
* временная
* алгоритмическая
* структурная

**Параметрическая** - облегчение режимов работы элементов и узлов аппаратуры. Малоэффективна в хорошо спроектированных ИС.

**Временная** - заключается в наличии дополнительного времени для решения задачи (в т.ч. для повторной обработки данных). Создает предпосылки для реализации реконфигурации, повторения вычислений

**Алгоритмическая** - заключается в применении алгоритмов, которые обеспечивают удовлетворительные результаты в случае наличия или возникновения ошибок в процессе обработки информации. Предполагает наличие временной избыточности и является средством ее реализации.

Свойствами избыточных алгоритмов обладают итерационные алгоритмы (обеспечивают сходимость при больших случайных отклонениях промежуточных результатов).

**Структурная** - выражается в наличии дополнительных элементов, узлов, устройств в структуре ИС, предназначенных для автоматической замены отказавших компонентов. Является наиболее эффективным видом избыточности.

Последовательность состояний отказоустойчивой ИС:

1. Работоспособное состояние
2. Возникновение ошибки
3. Выявление ошибки
4. Локализация ошибки
5. Реконфигурация системы
6. Восстановление потерянной информации
7. Восстановление вычислительного процесса
8. Работоспособное состояние системы

## 🍏117. Способы и средства нейтрализации ошибок и отказов в ИС.

Простейший способ - **повторение вычислений**. Позволяет устранить только ошибки, вызванные сбоями, и требует значительных затрат машинного времени

На практике используют:

* маскирование ошибочных действий
* реконфигурацию системы

**Маскирование ошибочных действий** - избыточная информация скрывает действие ошибочной информации за счет особенностей схемных решений и организации процесса обработки данных.

Средства маскирования делятся по принципу действия на:

1. корректирующие коды (CRC, Хэмминга, итеративные коды и др.)
2. логика с переплетениями
3. схемы с голосованием

**Реконфигурация системы** - изменение состава средств обработки информации или способа их взаимодействия. Производится после выявления отказа.

Включает:

* статическую реконфигурацию – осуществляется путем отключения отказавших компонентов. Система делится на две части: активную, участвующую в работе, и пассивную, охватывающую неработоспособные компоненты системы и отключенные в ходе реконфигурации.
* динамическую реконфигурацию – делится на:

1. реконфигурация замещением (поддержка запасом)
2. реконфигурация дублированием

**На практике**: динамическая реконфигурация - способность изменять аппаратные ресурсы сервера без необходимости его закрытия, она важна в любой среде, где приоритетом является период безотказной работы прикладной программы.

Поддержка ДР со стороны аппаратных средств и операционной системы недостаточна. Пока приложения не будут иметь информацию и возможность реагировать на изменения основных системных ресурсов, события ДР могут быть не в состоянии завершиться успешно.

После реконфигурации для продолжения нормальной работы ИС необходимо ее восстановить.

Восстановление системы происходит на двух уровнях:

1. Аппаратный уровень

* Автоматическое восстановление – путем дополнительной реконфигурации (в системе имеется ряд запасных блоков)
* Ремонт (восстановление вручную)

1. Программный уровень

* повторение операции на различных уровнях (команд или микрокоманд) – может дать правильный результат, если связанная с ними ошибка является случайной или временной
* возвращение к контрольной точке. Контрольная точка – некоторый этап процесса обработки информации, для которого зафиксированы (в ЗУ) промежуточные результаты и информация о состоянии системы, позволяющая возобновить обработку данных. При обнаружении ошибки система возвращается к КТ, предшествующей моменту возникновения отказа, и продолжает работу, используя данную точку в качестве исходной
* повторное выполнение программы – все незавершенные (до возникновения отказа) программы выполняются с самого начала. Применяется в случаях:

1. если последствия отказа успели отразиться на большей части системы
2. если возможно восстановление только части вычислительных процессов
3. если продолжение работы системы при использовании других способов восстановления сопряжено с трудностями и большими затратами времени

## 🍏118. Цели и виды сертификационных испытаний ПС.

Для удостоверения качества, надежности и безопасности применения ПС следует подвергать обязательной сертификации аттестованными, проблемноориентированными испытательными лабораториями.

Если все испытания проходят успешно, то на версию ПС оформляется специальный документ - **сертификат соответствия**.

Цели:

* защита интересов пользователей, государственных и ведомственных интересов на основе контроля качества ПО
* подготовка и принятие решения о целесообразности выдачи серти­фиката соответствия

Виды:

* Обязательная сертификация - для ПС, выполняющих особо ответственные функции, в которых ошибки или отказы могут нанести большой ущерб или опасны для жизни и здоровья людей
* Добровольная сертификация - для удостоверения качества ПС с целью повышения их конкурентоспособности, расширения сферы использования и получения дополнительных экономических преимуществ

Решение о выдаче сертификата на ПС основывается на оценке степени его соответствия специально разработанным документам:

* международным и национальным стандартам на тестирование, испытания, аттестацию программ и баз данных
* международным и государственным стандартам на технологию создания компонент ПС, языки программирования, их синтаксическим, семантическим и лексическим требованиям
* стандартам на сопровождающую ПС документацию
* нормативным документам - техническим условиям, техническим описаниям, эксплуатационным документам на ПС по выбору заказчика, разработчика и испытателя

Заявитель для получения сертификата соответствия направляет в орган по сертификации заявку на проведение испытаний с указанием схемы проведения сертификации.

Основным выходным документом является Протокол испытаний.

Протокол испытаний должен содержать информацию:

* служебную - регистрационный номер, дату, реквизиты утверждения, сведения о ПС
* исходную - краткие сведения об испытанном объекте и заявителе, основание для проведения испытаний
* выходную - краткие сведения о проведенных испытаниях, (в т.ч – рез-ты испытаний структурных компонент ПС с указанием реквизитов протоколов испытаний компонент
* сведения о нормативных документах, на соответствие которым проверялось ПС
* итоговую - краткие сведения о результатах испытаний, выводы и предложения

## 🍏119. Стратегия разработки политики безопасности и защиты информации в организациях.

(см5 (?))

## 🍏120. Безопасность транзакций в системах электронной коммерции

Электронная коммерция (e-commerce) - это любая форма бизнес-процесса, в котором взаимодействие между субъектами происходит электронным образом

Компоненты модели электронной торговли:

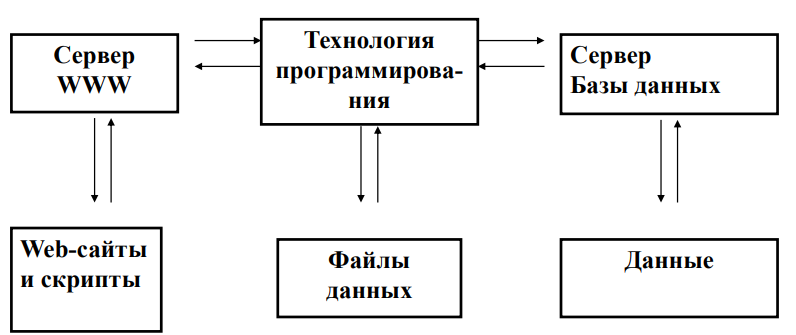
* электронный магазин
* клиент (покупатель)
* платежная система
* банк (кредитная организация)
* поставщик

Минимум компонентов программного обеспечения, необходимых для функционирования интернет-магазина:

* Web-сервер (Apache, Microsoft-IIS, Netscape-Enterprise)
* Сервер приложений
* СУБД (MySQL, PostgreSQL, ORACLE, MS SQL)

Основой для ведения электронной коммерции является Web-сайт

Во время сканирования web-сайтов наступает обмен заданиями, генерируемыми браузером, а также ответами на эти задания, генерируемыми сетевым сервером. Такой сервер следует рассматривать как часть Интернет-магазина:



Каждая транзакция между приложением WWW и Пользователем начинается с обращения, сгенерированного поисковой системой в адрес сервера сети Web. Если искомым объектом будет скрипт, то сервер передает его обработку соответствующему механизму, который обслуживает скрипты.

Информационные уязвимости интернет-проектов электронной торговли:

* возможность доступа к информационным ресурсам извне
* нападения хакеров
* вредоносные программы — вирусы и троянские кони
* частое использование электронной почты может помочь злоумышленникам скомпрометировать имена пользователей торгующей организации
* специальные программы могут быть использованы для поиска слабых мест в системах хранения пользовательских данных
* «отказы в обслуживании» (DoS)

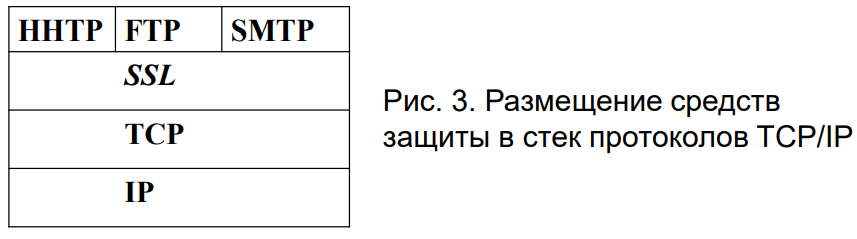
Основные методы обеспечения безопасности систем e-commerce:

* коммуникационные протоколы
* средства криптографии
* механизмы авторизации и аутентификации

**Коммуникационные протоколы**

Универсальным решением является размещение средств обеспечения безопасности над протоколом TCP

Примером подхода является стандарт SSL (Secure Socket Layer — протокол защищенных сокетов)



Протокол SSL призван обеспечить возможность надежной защиты сквозной передачи данных с использованием протокола TCP, его архитектура состоит из двух уровней протоколов: протокол записи SSL и три протокола более высокого уровня (протокол квитирования + протокол изменения параметров шифрования + протокол извещения )