Оглавление

[Всякая теория 2](#_Toc439165289)

[Принципы защиты информации 3](#_Toc439165290)

[Персональные данные 3](#_Toc439165291)

[Порядок внедрения средств защиты 4](#_Toc439165292)

[Защита ПО от несанкционированного копирования 4](#_Toc439165293)

[Моделирование угроз нарушителя 5](#_Toc439165294)

[Модель нарушителя 6](#_Toc439165295)

[Модели доступа 6](#_Toc439165296)

[Шифрование информации 7](#_Toc439165297)

[Симметричные алгоритмы шифрования 8](#_Toc439165298)

[Немного истории 8](#_Toc439165299)

[Случайные числа 10](#_Toc439165300)

[Использование в криптографии 11](#_Toc439165301)

[Алгоритмы генерации псевдослучайных чисел 11](#_Toc439165302)

[Data Encryption Standard (DES) 12](#_Toc439165303)

[Advanced Encryption Standard (AES) 13](#_Toc439165304)

[Алгоритмы ассиметричного шифрования. 14](#_Toc439165305)

[Хеш-функция 16](#_Toc439165306)

[Электронная подпись 16](#_Toc439165307)

[Квантовая криптография 18](#_Toc439165308)

[Алгоритм BB84 18](#_Toc439165309)

[Сжатие данных 19](#_Toc439165310)

[Алгоритм Шеннона-Фано 19](#_Toc439165311)

[Алгоритм Хаффмана 19](#_Toc439165312)

[Адаптивный алгоритм Хаффмана 20](#_Toc439165313)

[Алгоритм LZW 20](#_Toc439165314)

[Алгоритм арифметического кодирования 21](#_Toc439165315)

**Переходы по лекциям:**

* *Лекция №1 02.09.2015*
* *Лекция №2 16.09.2015*
* *Лекция №3 23.09.2015*
* *Лекция №4 30.09.2015*
* *Лекция №5 07.10.2015*
* *Лекция №6 14.10.2015*
* *Лекция №7 21.10.2015*
* *Лекция №8 28.10.2015*
* *Лекция №9 11.11.2015*
* *Лекция №10 02.12.2015*
* *Лекция №11 09.12.2015*

Лекция №1 02.09.2015

# Всякая теория

Федеральный закон 149-ФЗ: об информации, информационных технологиях и защите информации.

**Информация** – *сведения, сообщения, данные*, независимо от формы их представления. Более философское определение – накопленный опыт человечества.

**Жизненный цикл информации:**

1. Создание
2. Оценка (достоверность, бесполезность и т.п.)
3. Подготовка к хранению
4. Хранение
5. Выборка
6. Обработка
7. Использование

В результате использования появляются отчетные данные, которые также поддаются оценке и т.п. Оцененные данные можно уничтожить. Обновленные данные также должны проходить оценку.

**Документ** – информация, зафиксированная на *материальном* носителе с целью её дальнейшего изучения и передачи.

**Электронный документ** – документированная информация, представленная в *электронной* форме. Воспринимается субъектом с использованием ЭВМ.

**Защита информации** – комплекс мер (правовых, организационных и технических), направленных на:

* Предотвращение неправомерных действий с информацией: доступ, модификация, копирование, уничтожение, блокировка, предоставление (конкретным субъектам) и распространение (неограниченному кругу).
* Соблюдение конфиденциальности информации ограниченного доступа.
* Реализацию права на доступ к информации.

***С***тандарт ***Т***ехнического ***О***беспечения ***Б***анка ***Р***оссии по ***И***нформационной ***Б***езопасности ***Б***анковских ***С***истем; 1.0-2014.

**Актив** – всё, что имеет ценность для субъекта и находится в его распоряжении.

**Информационная сфера**:

* Информация
* Информационная инфраструктура
* Субъекты, обрабатывающие информацию
* Процедуры и правила, по которым информация обрабатывается
* Система регулирования отношений (что, где, как)

**Угроза** – опасность, предполагающая возможные потери.

**Безопасность** – состояние защищенности в условии угроз.

**Информационная безопасность** – безопасность в условиях угроз в информационной сфере.

Информационная безопасность обеспечивает:

* Доступность информации субъектам в том виде, в каком она была занесена в систему
* Целостность информации
* Конфиденциальность информации
* «Неотказуемость», защита от отказа со стороны субъекта
* Подотчётность, обязательный контроль операций над информацией
* Аутентичность (подлинность) и достоверность информации

**Идентификация** пользователя – присвоение ему уникального имени (логин).

**Аутентификация** – установление и подтверждение подлинности предъявленного идентификатора (пароль, материальный токен, биометрические данные и т.п.).

**Авторизация** – определение субъекта и предоставление ему прав доступа к защищаемому ресурсу.

**Ценность информации** – мера ущерба, наносимого нарушением безопасности информации. Ключевой критерий при выборе средств защиты. Виды важности информации:

* Жизненно важная (без неё функционирование системы невозможно)
* Важная (ущерб велик, хотя работоспособность системы сохраняется)
* Полезная (используется при работе)
* Несущественная (вспомогательная информация)

Система защиты информации должна удовлетворять *четырём критериям*:

1. Простота
2. Полнота
3. Изоляция от внешнего доступа
4. Ответственность (см. неотказуемость)

## Принципы защиты информации

**1. Обоснованность доступа**. Пользователь должен иметь достаточный уровень доступа для выполнения своих обязанностей. При этом для предоставления доступа необходимы основания.

**2. Достаточная глубина контроля**. Система должна контролировать все действия на всех уровнях.

**3. Разграничение потоков информации**. К примеру, нельзя перемещать секретную информацию на общедоступный носитель и наоборот.

**4. Частота повторного использования**. Необходимо надежное уничтожение информации.

**5. Персональная ответственность**.

**6. Целостность средств защиты**. Любая незначительная брешь в одном из компонентов системы подвергает компрометации всю систему целиком.

## Персональные данные

152-ФЗ фиксирует правила обработки персональных данных.

**Персональные данные** – любые данные, позволяющие *идентифицировать* конкретного человека.

* *Общедоступные* источники персональных данных: например, имя-фамилия в резюме на сайте
* *Специальные* персональные данные: паспортные данные, СНИЛС и т.д.
* *Биометрические* персональные данные: отпечатки пальцев, медицинская карта, данные о здоровье

Цикл Деминга:

планирование

реализация

проверка

усовершенствование

Лекция №2 16.09.2015

# Порядок внедрения средств защиты

1. Аналитическое обследование автоматизированных систем, которые планируется защищать
   1. выявить уровни конфиденциальности информации (что)
   2. установить уровни доступа (кто)
   3. установить правила разграничения доступа (как)
   4. создание модели нарушителя
2. Проектирование системы защиты информации
   1. выбор методов
   2. оформление документации
3. Создание системы
   1. тестирование
   2. …
4. Прием в эксплуатацию
5. Эксплуатация
6. Развитие

# Защита ПО от несанкционированного копирования

Варианты защиты можно разделить на несколько категорий:

1. Внутренняя самозащита
   * Пассивные:
     + Парольные замки
     + Ограничения по времени использования
     + Наблюдение
   * Активные:
     + Искажение программы
     + «вирусы» (в хорошем смысле)
2. Видимая самозащита (активные)
   * Различные сигналы
   * Информация о владельце
3. Идентификация
   * Корреляционный и частотный анализ
   * Различные виды избыточности и внедренных ошибок
4. Информационные средства (пассивные)
   * Водяные знаки на лицензиях, голографические наклейки
   * Реестр ПО
5. Составляющие системы
   * Изменение формата записи
   * Специальные методы разметки (дискета со специальным размером кластера или предварительно поврежденным конкретным сектором)
   * Уникальные метки
6. Специальная аппаратура
   * Использование данных процессора
   * Использование серийных номеров
   * Аппаратные средства защиты: токены и электронные ключи
7. Изменение функций
   * Перегрузка прерываний
   * Переименование дисков
8. Запрос дополнительной информации
   * Пароль
   * Персональные данные
9. Шифрование
   * Симметричное
   * С открытым ключом
10. Проверка сигнатур
    * Характеристики ПК
    * Каталоги/реестр
    * Привязка к уникальному электронному ключу
11. Защита от автоматизированного подбора
    * Ограничение числа попыток
    * Капча
    * Монитор активности

Как своеобразный способ можно упомянуть наличие\отсутствие документирования.

*К вопросам лабы.*

*Два типа средств: постоянные и переменные. Постоянные параметры: почти всегда аппаратные средства.*

*Критерии оценки при выборе параметров:*

* + *уникальность*
  + *неизменность*
  + *доступность*
  1. *Самый легкообходимый метод - использование Get\*\*\* в WinApi. GetWindowsDirection ... GetCurrentHWProfile*
  2. *Windows Management Instrumentation. Используется язык запросов WQL: Select \* From Win32\_Bios*
  3. *В случае линуксов: привязка к /proc, например /proc/cpuinfo*
  4. *В случае маков придётся использовать sysctl -a hw*

# Моделирование угроз нарушителя

Важный этап при создании системы защиты информации. **Цель** такого моделирования – заставить разработчика *мыслить* *конструктивно* (т.е. на основе формальных требований) при проектировании систем с точки зрения информационной безопасности.

Моделирование угроз включает в себя ***шесть* этапов**:

1. определение активов (что)
   * ресурсы (информация, её пользователи, инфраструктура)
   * средства контроля безопасности
2. описание архитектуры
   * границы системы
   * функции и возможности
   * технологии
3. декомпозиция системы
   * области защиты (разное защищается по-разному)
   * политики безопасности
     + проверка входных данных
     + проверка подлинности
     + авторизация
     + управление конфигурацией
     + выделение конфигурационных данных
     + шифрование
     + управление параметрами
     + управление исключениями
     + аудит
4. определение угроз (источники)
   * природные
   * техногенные
   * антропогенные
     + умышленные
     + случайные
5. документирование
   * цель
   * риск
   * методы борьбы
6. оценка серьёзности угроз
   * потенциальный ущерб
   * воспроизводимость
   * доступность атаки
   * активы (субъекты + объекты)
   * легкость обнаружения

## Модель нарушителя

Низкий уровень возможности – обычные пользователи системы, с крайне ограниченными возможностями.

Средний уровень возможности – предусматривается создание и запуск новых средств с собственными функциями.

Высокий – воздействие на функции системы

Абсолютный – имеют отношение к проектированию и реализации системы.

Хакеры:

Увлеченные (любопытство, развлечение, крутизна, получение денег)

Профессионалы (вербовка, изучение открытой информации, электронная почта, анализ "мусора" в виде распечаток и т.п., кража)

Классификация автоматизированных систем с точки зрения безопасности

группа 3: однопользовательские системы, пользователь имеет доступ ко всей информации (3A, 3B)

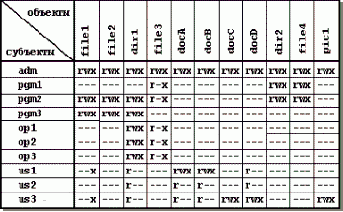
группа 2: многопользовательские системы, с одинаковым уровнем доступа (2A, 2B)

группа 1: многопользовательские системы с разным уровнем доступа (1A - 1D)

# Модели доступа

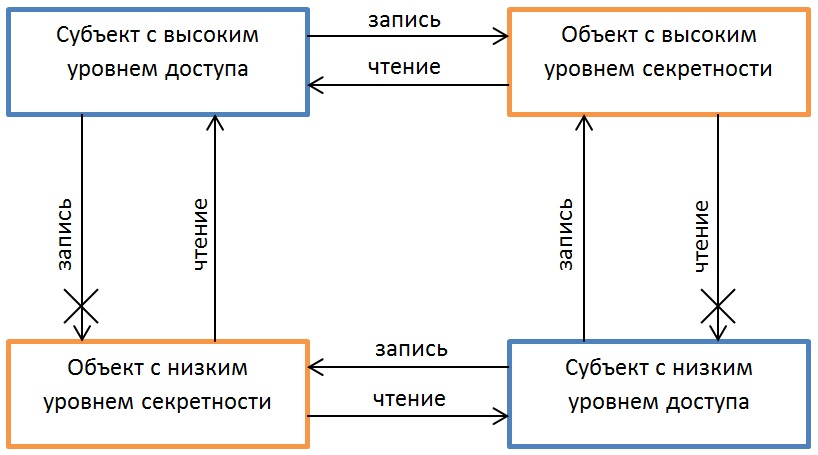
Модель доступа *устанавливает соответствие* ресурсов к субъектам, к ним обращающимся.

Самая простая модель – **матрица доступа** (Харрисона-Ризо-Ульмана). Включает в себя перечень *субъектов* и *объектов*. В различных вариациях развития могут быть виды доступа, время доступа, различные условия. Отображается в виде простой таблицы, в которой перечисляются все субъекты и объекты. В *ячейках* указывается что субъект I *имеет доступ* к объекту J на чтение\запись\редактирование\удаление.



Лекция №3 23.09.2015

**Мандатная модель** (Белла-Лападулы). Уровень доступа *субъекта* должен быть *не ниже* чем уровень *объекта*. Аналогия с документами: совершенно секретные особой важности, совершенно секретные, секретные, для служебного пользования.



**Ролевая модель**. Развитие модели матрицы доступа. Дополняется ролями: объекты соответствуют ролям (одна матрица) и роли соответствуют субъектам (вторая матрица).

При создании систем необходимо *журналирование*. Оно может давать следующие профиты:

* + выделение частых ошибок пользователей
  + выделение нагруженности элементов в системе
  + восстановление системы по журналам в случае сбоя

Криптография

Шифрование информации.

**Шифрование** – преобразование открытого текста в зашифрованный, plaintext -> ciphertext, с целью сохранения конфиденциальности. Enciphering и deciphering.

**Требования** к алгоритмам шифрования:

1. *сложность расшифровки* и модификации

2. *малое* изменение исходного текста/ключа влечет *значительные* изменения шифра текста

3. область значений ключа должна исключать его *перебор*

4. стоимость *дешифровки* превышает стоимость *данных*

Требования были сформулировал голландец Керкгоффс. **Правило Керкгоффса**: стойкость алгоритма определяется только секретностью ключа.

**Рассеивание информации** - влияние одного знака открытого текста на несколько знаков шифрованного текста.

# Симметричные алгоритмы шифрования

Алгоритмы с единым секретным ключом шифрования и расшифровки. Включают в себя *блочные* и *поточные* алгоритмы.

**Блочные** – используют для шифрования блоки. Алгоритмы подстановки, перестановки. Строка разбивается на блоки, которые преобразуются по отдельности. Результирующие блоки должны быть больше оригинальных. Используются для небольших сообщений.

Шифр Фейстеля – разбиение исходного сообщения на правую и левую части:

левая складывается по модулю 2 с функцией ключа, и записывается в правую часть зашифрованного сообщения;

правая без изменений записывается в левую часть зашифрованного сообщения.

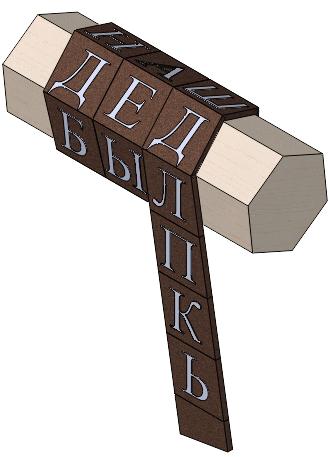


**Поточные** - работают с непрерывным потоком данных. Передача начинается с синхропосылки. Дальше идёт первый блок данных, шифруется ключом (банально - XOR) с синхро. Зашифрованное выкладывается в канал, берётся второй блок данных, шифруется ключом с результатом первого блока, ...

## Немного истории

Алгоритм "скитала".

Палочка, намотана лента, на ленте буквы. Буквы, "находящиеся" на палочке складываются в слово. Ключ – палочка такого же диаметра.



Перестановка по коду.

Задается правило (например 312): word[1] -> word[3], word[2] -> word[1], word[3] -> word[2], … . Используется для перестановки элементов в тексте. "|Тек|сты|" -> "кТеыст..."

PigPen.



Каждой букве ставится в соответствие символ в соответствие с таблицей,

An example pigpen message

Шифр Цезаря - сдвиг в алфавите.

Квадрат Полибия.

A B C D

v v v v

E F G H

.......

Великий шифр Людовика 14.

Пары символов меняются на кодовые значения. ое -> 123, ая -> 256. Вводится специальная комбинация, означающая удаление предыдущего слова.

Таблица омофонов.

Символу ставится в соответствие столько значений, сколь часто он встречается. A соответствует пять значений, Z одно. Размывает статистику, мешает частотному анализу.

Переходный вариант – вроде бы подстановка ещё на одном алфавите, но алфавит уже неоднозначный.

**Одноалфавитные** подстановки - для шифрования всех символов сообщения используется один алфавит.

**Полиалфавитные** - для шифрования каждого символа используются разные алфавиты.

Шифр Виженера.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | … | K | … | X | Y | Z |
| A | A | B | C | … | K | … | X | Y | Z |
| B | B | C | D | … | L | … | Y | Z | A |
| C | C | D | E | … | M | … | Z | A | B |
| … |  |  |  | … |  | … |  |  |  |
| K | K | L | M | … | W | … | H | I | J |
| … |  |  |  | … |  | … |  |  |  |
| X |  |  |  | … |  | … |  |  |  |
| Y |  |  |  | … |  | … |  |  |  |
| Z |  |  |  | … |  | … |  |  |  |

Все символы для каждой новой строки сдвигаются построчно. Ключом является какое-то слово. Над каждым символом сообщения записывается символ из циклически повторяющегося ключа:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | E | Y | K | E | Y | K |
| M | E | S | S | A | G | E |

Ищем в таблице замену в строке для буквы из ключа столбце буквы из сообщения. [K][M]=W, затем [E][E]=…

Алгоритм Цезаря, повторённый 26 раз. В зависимости от ключа, заменяем по-разному.

**Энигма**.

<http://habrahabr.ru/post/217331/>

<http://habrahabr.ru/post/143526/>

3 ротора и рефлектор. Если Reflect(A)=B, то Reflect(B)=A. Одна машина используется и для шифрования и для расшифровки. Ключом является положение трёх роторов.

Чтобы использовать многоалфавитность, роторы нужно поворачивать. Вначале проворачивается первый ротор, спустя 256 символов аски - поворачивается второй, и снова начинает поворачиваться первый...

Путь символа: кнопка -> правый ротор -> центральный -> левый -> рефлектор -> левый -> центральный -> правый -> экран.

*Цель лабы - продемонстрировать на трёх произвольных файлах: пустом, однобайтовом, и .ехе/архиве/пдфке*

Лекция №4 30.09.2015

## Случайные числа

**Реальные** случайные числа:

* различные шумы
* время получения сетевых пакетов сетевой картой
* счетчик Гейгера
* турбулентные потоки воздуха в вентиляторе
* действия пользователя: мышь, клавиши, движения перед камерой, микрофон

**Квазислучайные** числа – искусственно приведенные к равномерному распределению (используются в методе Монте-Карло).

**Псевдослучайные** числа – числа, которые образуют последовательность, случайную с точки зрения статистических законов (друг от друга не зависят, а распределение, как правило, нормальное). Требования к алгоритму генерации:

* длина периода должна быть больше интервала использования
* независимость последовательных значений
* **равномерность распределения**
* необратимость

Равномерность проверяют по двум критериям. Самый простой – возможность аппроксимации полиномом N-ной степени.

Вторая группа тестов оценивает равномерность по заданному количеству бит – после отсечения какой-то части (например, беря 5битные числа от 8битных), результат должен быть также распределен равномерно. N-мерность - насколько глубоко можно отсекать так значения.

### Использование в криптографии

Последовательность чисел целиком: одноразовый отрывной блокнот со случайными числами. Если кто-то крадёт блокнот, то последовательность скомпрометирована.

Потоковое сложение с К, получаем новый ключ

Как инициализирующее значение для математической функции

Как порядковый номер в большом массиве

Как шаг в последовательности

Возведение бит

### Алгоритмы генерации псевдослучайных чисел

**#1.**

**#2. алгоритм фон Неймана (метод серединных квадратов)**

Инициализирующее число: 1234

Берем среднюю часть и возводим в квадрат: 23 -> 0529; 52 -> 1704; 70 -> 4900; 90 -> 8100; 10 -> 1000; 0 -> 0. Можем очень быстро упереться в «потолок» при неправильном инициализирующем числе.

**#3. метод серединных произведений**

**#4. метод перемешивания**

**#5. на числах Фиббоначи**

Берется последняя цифра от числа: 0 1 1 2 3 5 8 3 1 ...

***#6. Простой Линейный Конгруэнтный Генератор***

Вычисляются члены рекуррентной последовательности по модулю натурального числа. . Качество очень сильно зависит от коэффициентов. Рекомендуемые значения:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | M | Период |
| 106 | 1283 | 6075 |  |
| 430 | 2531 | 11979 |  |
| 84589 | 45989 | 217728 |  |

**#7. Вихрь Мерсенна**

Состоит из двух частей: рекурсия аналогично предыдущим + закалка, убирающая зависимости. Используется матрица состояний 624стр\*32стб.

На каждом шаге берется какое-то значение в матрице, XOR с первыми 32 битами, получается новая матрица состояний - из неё удалено слово А из начала, добавлено слово А XOR Б в конец (являющееся случайным словом).

Закалка - все зависимости бит из результата убираются.

Из равномерного распределения можно получить нормальное через

Лекция №5 07.10.2015

Симметричное шифрование.

Проблемы:

* ключом необходимо обмениваться
* ключ необходимо хранить
* ключ необходимо безопасно использовать - он не должен оставаться в памяти машины
* ключ необходимо безопасно уничтожать

ГОСТ 28/47 – используется ключ на 256 бит.

В большинстве алгоритмов используется шифр Фейстеля - подстановки и перестановки.

## Data Encryption Standard (DES)

<https://ru.wikipedia.org/wiki/DES>

Длина ключа 64 бит; каждый восьмой бит контролирует четность предыдущих семи (таким образом «значащие» только каждые 7 бит вместо 8-ми). Допускается использование просто 64битного ключа. Используются блоки данных по 64 бита.

Состоит из двух частей - преобразование ключа и собственно шифрование.

Преобразование ключа: из 64бит исходного ключа необходимо получить 16 48-битных раундовых ключей.

1. начальная перестановка в исходном ключе (убирает каждый 8 бит)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 | 58 | 50 | 42 | 34 | 26 | 18 | C0 |
| 2 | 10 | 2 | 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 3 | 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 | 62 | 54 | 46 | 38 | 30 | 22 | D0 |
| 4 | 14 | 6 | 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 | 28 | 20 | 12 | 4 |

2. ключ делится на две половины по 28 бит: C0, D0.

3. для каждой половины выполняется циклический сдвиг влево **S1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| шаг | **1** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Число сдвига | **1** | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

(Итерации 1,2,9,16 - сдвиг на 1 бит, остальные - на 2 бита)

4. сдвинутые половины объединяются и получается новый ключ на 56 бит

5. выполняется циклический сдвиг S2...

Шифрование:

0. входной блок М длины 64 бита

1. выполняется начальная перестановка **P**, каждому биту указывается место, куда его нужно переставить.

2. блок делится на две половины L0 и R0 по 32 бита

3. ; L1 = R0 (блоки переставляются в соответствие с сетью Фейстеля)

Функция Фейстеля (см. вики):

i-й раунд. На входе 32битный блок, расширяется до 48 перестановкой Е (дублирует некоторые биты). Результат XORится с i-м ключом. 48битный результат делится на 8 блоков по 6 бит. 1-й и 6-й бит дают строку, 2-5 дают столбец в таблице SBox, в ней хранятся 4битные блоки. Объединяем их, получаем 32 бита.

Повторяется с 1 по 15 раунд. На 16м процедура меняется: L16 = L15 xor F(R15,K16); R16 = R15.

4. Полученные блоки объединяются и выполняется конечная перестановка, обратная к перестановке **Р**.

При расшифровке ключи используются в обратном порядке.

3-DES - тройное шифрование. C = E(K1, D(K2, E(K3, M))). M' = D(K3, E(K2, D(K1, C))). Размер ключа увеличивается до 192

## Advanced Encryption Standard (AES)

Блок – 128бит, ключи – 1 28, 192 и 256. Алгоритм должен быть криптостойким по отношению к атакам, известным на момент создания. Ясная и обоснованная структура самого алгоритма. Отсутствие слабых и эквивалентных ключей (Е(к1) == Е(к2)). Скорость. Минимальная память. Возможность распараллеливать вычисления при зашифровке. Использование как для блочного шифрования так и для других видов.

Лекция №6 14.10.2015

<https://ru.wikipedia.org/wiki/Advanced_Encryption_Standard>

<http://habrahabr.ru/post/212235/>

Data: 128бит (4 слова по 32бит)

K\_i: 128бит, 192бит, 256бит

В 128битном ключе: ~10^36 вариантов.

Составление раундовых ключей W[]:

Первые Nk байт (4 столбца) – копии байтов из исходного ключа

Для столбцов, кратных длине ключа (i mod Nk == 0):

Для остальных –

Nk – длина ключа в 32битных словах (=4 для 128битного ключа)

Шифрование: 4 операции

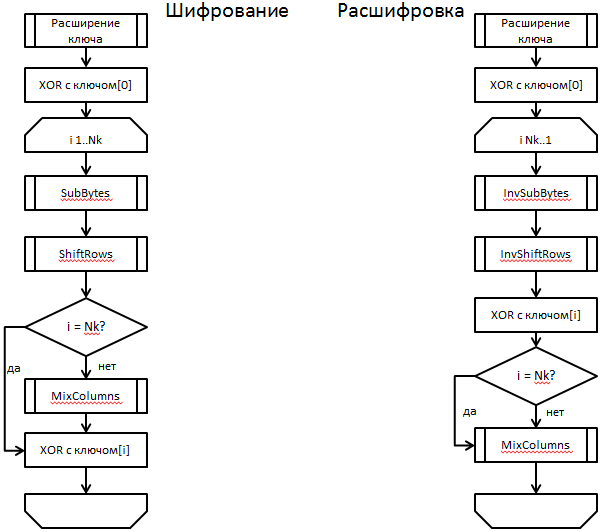
1. Побитовое сложение очередным раундовым ключом, AddRoundKey

2. Циклический левый сдвиг строк в блоке, ShiftRows (на 1 для 2-й строки, на 2 для 3-й, на 3 для 4-й)

3. Смешивание столбцов в блоке, MixColumns (каждый столбец представляется многочленом, затем перемножается в поле GF(28) по модулю x4 + 1 с многочленом 3x3 + x2 + x + 2).

4. Подстановка SubBytes.

Суть АЕС - последовательное применение этих операций. На последнем раунде не используется перемешивание столбцов.



Лекция №7 21.10.2015

(by krivolapov)

# Алгоритмы ассиметричного шифрования.

<https://ru.wikipedia.org/wiki/RSA>

<http://www.paveldvlip.ru/algorithms/rsa.html>

Алгоритмы с открытым ключом. Открытый ключ для шифрования и закрытый для расшифровки.

Ключи создаются на стороне получателя.

В 1977 был опубликован RSA(Rivest,Snamir,Adleman)

Создание ключей

1. P,Q – простые числа
2. Найти E такое, чтобы
3. Найти D такое, чтобы:

Проблемы

1. Получение простого числа
2. Поиск НОД – алгоритм Евклида
3. Расширенный алгоритм Евклида

Способы определения простоты:

1. Перебор(2..sqrt(N)) с проверкой делимости
2. Решето Эратосфена
3. Теорема Рабина. Берётся m>2 и для него вычисляются Берётся a от 1 до м.  
   Если выполняется или , то а – свидетель простоты
4. Теорема Милера-Рабина

Расширенный алгоритм Евклида

, где x,y – неизвестные множители, d – наибольший общий делитель. В нашем случае:

Зададим матрицу

//

Если r !=0:

Иначе – правый столбец матрицы E является ответом, .

Возведение в степень

BigInt fastmodpow(const BigInt &a,BigInt pow,const BigInt &mod)

{

BigInt t(a), res(1);

BigInt zero(0);

while (pow>zero){

if (pow.isOdd())

res = (res \* t) % mod;

t = (t\*t) % mod;

pow >>= 1;

}

return res;

}

Пример вычисления ключей

D=107

Лекция №8 28.10.2015

# Хеш-функция

Преобразование, используемое для аутентификации сообщений, определяемое следующими свойствами:

* применима к блоку данных любой длины
* создание выхода фиксированной длины
* необратимость
* легкость вычисления

Применение:

Аутентификация (прислался тот же файл что и отправлялся).

Защита данных (хранить в базе не сам пароль, а его хеш).

Примеры функций:

* MD5, 1991. Входные данные - 128бит, выход - 128бит. Основная уязвимость - "тесты на дни рождения"
* Secure Hash Algorithm
* SHA-0, 1993. 512бит => 160бит. Прожил всего полгода.
* SHA-1. 512 => 160, устранены все недостатки ша0. На долгое время стал стандартом.
* SHA-2. Собирательное обозначение для целого семейства алгоритмов: ша224, ша256, 384, 512; различный размер выходных данных. Вход 512/1024бит.

Чем больше захватывается блок в алгоритме – тем больше используется информации для создания хэша, чем меньше на выходе – тем выше вероятность нахождения двух разных блоков, дающих одинаковый выход

Смысл алгоритмов хеш-функций – входной блок делится на части, над которыми выполняются XOR, SHIFT(R,L), AND, OR, etc. Получаем перемешивание данных внутри каждого блока. Для каждого блока заводятся "константы", которые складываются\умножаются с блоками данных. Каждый блок перемешивается N раундов; "константы" заменяются на биты преобразованного предыдущего блока.

# Электронная подпись

Подпись (простая, ручкой):

* жёстко связана с документом
* выражает добровольное согласие
* удостоверяет, что её поставил автор (аутентичность)
* подтверждает неизменность
* неотказуемость

Алгоритм создания электронной подписи включает два этапа:

1. вычисление хеша

2. создание собственно подписи

На обоих шагах можно использовать целое семейство алгоритмов.

Исходное сообщение М. Вычисляется хеш-преобразование, над которым выполняется шифрование (например, RSA, потребуется ключ). Зашифрованный хеш и будет подписью C.

Расшифровка: подпись C и сообщение M'. Вычисляется хеш от M'. Расшифровывается подпись C (с помощью ключа), получается исходный хеш. Вычисленный хеш сравнивается с расшифрованным, если не равны - исходное сообщение != переданному.

В случае РСА, закрытый ключ используется для расшифровки, открытый – для шифрования. Цель РСА в чистом виде – спрятать информацию, здесь же нужно проверить целостность сообщений – ключи меняются местами, *создаваемая подпись шифруется* ***закрытым*** *и расшифровывается* ***открытым***.

Раньше всё это дело звалось «электронно-цифровая подпись», а в 2011 году – наступил 63-ФЗ, и стало «электронная подпись».

Может быть три вида подписи – простая, усиленная (неквалифицированная, квалифицированная). Квалифицированная создается с использованием сертификата, выпущенного сертифицированным удостоверяющим центром.

Процедуры создания ЭП – ***С***редства ***К***риптографической ***З***ащиты ***И***нформации. Как правило - часть ОС (CryptoAPI, создаёт CryptoServiceProvider) и доступны всем программам. Фактически, CSP - внедрённая в ОС библиотека, также подписываемая; ОС - контролирует подпись этой библиотеки.

**Децентрализованная** схема – субъекты хранят закрытые ключи и обмениваются открытыми, чтобы проверять целостность передаваемых данных. Алгоритм PrettyGoodPrivacy. Проблема - нет арбитров, удостоверяющих правильность подписей, есть поле для подлога.

**Централизованная** схема – есть арбитр, который сохраняет открытые ключи. PublicKeyInfrastructure. В ответ на присланный открытый ключ выдаётся сертификат открытого ключа, соответствующего закрытому.

Центр сертификации - Удостоверяющий центр, Certificate Authority. Стандарт X.509 задаёт формат сертификата и обмена УЦ с субъектом-владельцем. Формат предусматривает хранение:

* информации о версии и алгоритме хеширования и шифрования
* информацию о подписанте (ФИО, должность, организация, etc.)
* срок действия сертификата (~2 года)
* информацию про удостоверяющий центр и полномочия, которыми он наделил получателя сертификата (подпись, шифрование).

Лекция №9 11.11.2015

Удостоверяющий центр выпускает сертификат на открытый ключ. Закрытый ключ - святая святых и во внешние миры ходить не должен.

Ключевой параметр сертификата - срок действия. Сертификат можно отозвать до его "протухания". Списки отозванных сертификатов (C. revocation list) - список сертификатов, которые прекратили действие досрочно. Эти файлы также публикуются УЦ.

Лекция №10 02.12.2015

# Квантовая криптография

Метод передачи информации, основанный на принципах квантовой физики. "Обычная" передача - электроны (частицы) и разность потенциалов (поток). Квантовая передача – использование единичных квантовых частиц.

Принцип неопределенности Гейзенберга – импульс и координаты – сопряженные величины (также напряженность электрических и магнитных полей; ток и напряжение). Произведение среднеквадратичных отклонений измерений каждой из величин не может превышать константу. Чем точнее измеряется одно, тем меньше точность измерения другого.

Можно построить криптографическую систему: информация не прячется\преобразовывается, но можно увидеть, что получен несанкционированный доступ. Перехваченный сигнал будет передан дальше в искаженном виде.

## Алгоритм BB84

Отправитель Алиса, получатель Боб – "легальные" пользователи; Ева – криптоаналитик. Для кодирования информации используются фотоны - могут быть поляризованы ортогонально либо диагонально. Из ортогонального (+) – поляризация 0/90 градусов (0 и 1), из диагонального (х) – 45/135 градусов (тоже 0 и 1).

На передаваемый сигнал требуется накладывать правильный базис, иначе могут появляться ошибки.

**I.** первичный этап

1. выбор поляризаторов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| А | 1 | 0 | 1 | 1 | 0 |
|  | **+** | **х** | **х** | **+** | **+** |
|  | → | ↗ | ↘ | → | ↑ |

2. направляется в квантовый канал

3. прием фотонов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Б | → | ↗ | ↘ | → | ↑ |
|  | **x** | **+** | **x** | **+** | **x** |
|  | ? | ? | ↘ | → | ? |
|  | ? | ? | 1 | 1 | ? |

4. по открытому каналу Б сообщает А, какие поляризаторы он выбрал

-''- А сообщает Б угаданные поляризаторы

5. выбирают угаданные биты

**II.** сверка данных

6. сверка контрольной суммы (передается также по открытому каналу)

+ исключается прослушивание (чуть что - меняем конфигурацию и начинаем заново)

- информация передается не целиком

- защищается канал а не информация

=> используется в алгоритмах квантового распределения ключей

Главная проблема - для передачи каждого бита информации требуется ~1 фотон. Проблемы с передачей информации.

Увеличиваем число фотонов на один бит - получаем выше надежность доставки, но снижаем безопасность.

Атом излучает фотоны с противоположной (горизонтальная\вертикальная) поляризацией в обе сторону. В одну сторону ведётся передача, в другую – запоминание.

Используются высокочувствительные фотодиоды, ловящие фотоны. Физический недостаток - если их "засветить", то управляя интенсивностью сигнала можно управлять считываемой информацией.

Лекция №11 09.12.2015

# Сжатие данных

Процесс сокращения числа бит, необходимых для хранения одного и того же числа информации. Сокращение избыточности. Выделяют три метода оного сокращения:

1. устранение повторений
2. изменение модели хранения данных (короткие последовательности для часто встречающихся элементов, длинные для редких)
3. сжатие с потерями

ДОС6.22, алгоритм DriveSpace сжатия на лету.

Самый простой алгоритм сжатия: добавляем бит к повторяющейся последовательности, 0 - данные, 1 - число повторов.

Алгоритм Шеннона-Фано, 1948

1. Строится таблица частот встречающихся символов.

2. Выбираются два подмножества символов, в которых сумма частот оных примерно равна (например: 10,9,5,1 частоты -> 10+1, 9+5), строится дерево. 0 - налево по дереву, 1 - направо. Буквы можно кодировать, например:

ABCD

0 BC

0 B

1 C

1 AD

0 A

1 D

Таким образом, A=10

Подмножества выделяются до тех пор, пока множество не будет содержать один символ.

Недостаток - нет четкой формализации, как будет выглядеть модель сжатия (сжимать можно разными методами). Четкого соответствия нет - множества выбираются по каким-то внутренним принципам, которые могут расходиться в разных случаях.

## Алгоритм Хаффмана

Основное отличие – построение дерева начинается с листьев в корень.

1. *Таблица частот* (сортировка)
2. Строится *дерево*: два символа с наименьшим весом (встречаемостью) (C,D). Они образуют узел с весом их суммарной встречаемости. Суммарный вес (6) заменяет их в таблице частот. Дальше берем 6 и B, образуют узел 15. Дальше берем 15 и A, образуем узел 25.
3. Аналогично Ш-Ф, *помечаются дуги* дерева как 0 налево и 1 направо.
4. ~~Бит 1 в файле всегда обозначает лист, а количество нулей перед ним - какой именно из листов.~~

//*Нихрена подобного!* [*http://habrahabr.ru/post/144200/*](http://habrahabr.ru/post/144200/)

Вместе с файлом необходимо хранить и исходную таблицу частот (либо само дерево).

Необходимо также учитывать остающиеся в хвосте пустые биты - они могут раскодироваться в совершенно лишнюю информацию.

## Адаптивный алгоритм Хаффмана

Позволяет избавиться от необходимости хранить дерево. При считывании очередного символа - перестраивается модель данных. Считали второй символ - получили два символа с равной частотой, четвертый, пятый; каждый раз перекодируется дерево. Алгоритм будет катастрофически медленным. Узел должен встраиваться в дерево сразу же на правильную позицию.

Алгоритм LZW (Лемпэл, Зив 1978; Уэлч 1984)

Текст «Going, going, gone!».

Строится таблица: код символа | префикс | суффикс. Заполняется символами из АСКИ-таблицы с -1 во 2-м и темже символом в 3-м столбце.

|  |  |  |
| --- | --- | --- |
| Код | префикс | суффикс |
| 0 | -1 | 0 |
| 1 | -1 | 1 |
| . | . | . |
| 255 | -1 | 255 |

Считывается первый символ из файла (G=71) и второй (o=111):

|  |  |  |
| --- | --- | --- |
| 256 | 71 | 111 |

Третий (i=105):

|  |  |  |
| --- | --- | --- |
| 257 | 111 | 105 |

Далее:

|  |  |  |
| --- | --- | --- |
| 258 | 105 | 110 |
| 259 | 110 | 103 |
| 260 | 103 | 44 |
| . | . | . |
| 264 | 111 | 105 | //соответствует *oi* в *Going* |

Это позволяет нам заменить комбинацию 111 105 на 257 110

Закодированным файлом будет последовательность префиксов начиная с 256 строки. Когда считываются элементы, если символ выходит за пределы 256 - он уже где-то есть в таблице выше.

## Алгоритм арифметического кодирования

Опять таблица частот. BILL GATES ->

|  |  |  |
| --- | --- | --- |
| символ | вероятность | интервал |
| \_ | 0.1 | 0.0-0.1 |
| a | 0.1 | 0.1-0.2 |
| b | 0.1 | 0.2-0.3 |
| e | 0.1 | 0.3-0.4 |
| g | 0.1 | 0.4-0.5 |
| i | 0.1 | 0.5-0.6 |
| l | 0.2 | 0.6-0.8 |
| s | 0.1 | 0.8-0.9 |
| t | 0.1 | 0.9-1.0 |

Кодируется дробным числом. Интервал 0..1 разбивается на вероятности, получаем дерево интервалов. По нему можно построить другую таблицу:

|  |  |  |  |
| --- | --- | --- | --- |
| символ | низ | верх | интервал |
| B | .2 | .3 | .1 |
| I | .25 | .26 | .01 |
| L | .256 | .258 | .002 |

В конечном итоге, строка будет закодирована дробью: 0.257167752