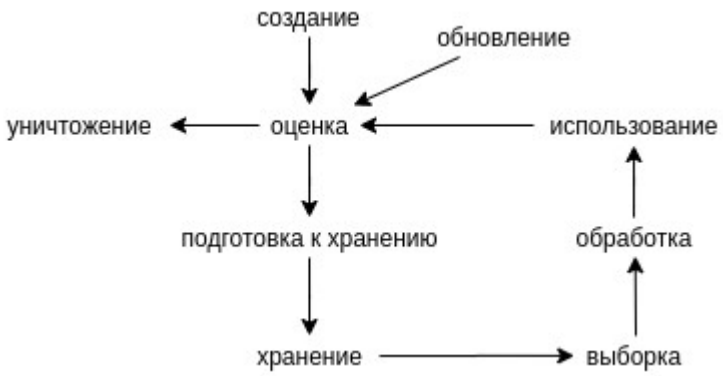
Лекция 1

Информация – это сведения, независимо от формы их представления.

Жизненный цикл информации:



На каждой стадии решаются вопросы защиты информации.

Документ – это информация, зафиксированная на материальном носителе и сопровождаемая реквизитами.

Электронные документы – документированная информация, представленная в электронной форме.

Защита информации – принятие мер (правовых, организационно-структурных и технических), направленных на:

- предотвращение неправомерных действий
- конфидециальность
- доступ к информации.

Правовые меры – законы. Организационно структурные – внутренние правила организации. Технические – программные и аппаратные средства.

Неправомерные действия:

- доступ
- копирование
- модификация
- блокирование
- предоставление и распространение
- уничтожение.

СТО БР ИББС – комплекс документов Банка России, описывающий единый подход к построению системы обеспечения ИБ организаций банковской сферы с учётом требований российского законодательства.

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

Информационная сфера:

- информация
- информационная инфраструктура
  - hardware
  - software
  - network
- субъекты
- процедуры

Над всеми четырьмя пунктами находится система регулирования.

Угроза – опасность, предполагающая возможность потерь или нанесения вреда.

Безопасность – состояние защищенности активов, информационных систем в условиях угроз.

Информационная безопасность – состояние защищенности систем в условиях угроз в информационной сфере.

Информационная безопасность включает в себя:

- доступность
- целостность
- конфидециальность
- *ответственность (неотказуемость)*
- *подотчетность*
- *аутентичность (подлинность)*
- *достоверность.*

Идентификация – присвоение уникального имени.

Аутентификация – установление подлинности предъявленного идентификатора:

- однофакторная – пароль
- двухфакторная – еще один канал, кроме пароля (биометрия, телефон)
- …

Авторизация – предоставление прав доступа.

Лекция 2

Ценность актива – мера ущерба, наносимого нарушениями безопасности актива.

Важность:

- жизненно важная
- важная
- полезная (рабочая)
- несущественная.

Что учитывать:

- простота
- полнота
- ответственность
- обоснованность доступа (необходимые и достаточные права доступа)
- разграничение потоков информации
- чистота повторного использования
- целостность средств защиты.

Методы защиты информации:

- системы аутентификации
  - пароль
  - ключ доступа
  - сертификат
  - биометрия
  - одноразовые коды
  - использование третьей доверенной стороны (ECUA)
- средства авторизации
  - модели доступа
  - журналирование
- криптографические средства
  - алгоритмы шифрования
  - электронная подпись
- системы анализа и моделирования информационных потоков:
  - мониторинг
  - моделирование и иммитация
  - межсетевое экранирование
- антивирусы + регулярные обновления
- регулярное резервное копирование
- резервирование hardware:

    - железо
    - питание
- физическая защита и режимные меры.

Персональные данные (ПДн):

- общедоступные источники Пдн
- специальные персональные данные
- биометрические данные
- трансграничная передача ПДн

Методы защиты от нелегального копирования:

- внутренняя самозащита (пароль на запуск, ограничение по времени)
- аутентификация и авторизация
- нарушение штатного функционала программы
- вирусное поведение
- аппаратные средства
- изменение формата хранения.

Виды параметров:

- постоянные (серийный номер ЖД)
- изменяемые (имя пользователя, имя компьютера)

Критерии выбора параметров:

- уникальность
- неизменность
- доступность.

Лекция 3

Уязвимость – это свойство системы, допускающее или способствующее реализации угрозы.

Моделирование угроз и нарушителей

Цель: заставить разработчика конструктивно мыслить при проектировании систем с точки зрения информационной безопастности (конструктивно – на основе формального описания).

1. Определение активов (что защищать?)
2. Описание архитектуры (где?) – фиксируются границы системы, определяется ее функционал, технологии
3. Декомпозиция системы – определяются области защиты внутри границ системы, формируются политики безопасности
4. Определение угроз. Выделяют следующие источники:
    - природные
    - техногенные
    - антропогенные
        - случайные
        - умышленные
5. Документирование угроз. Для каждой угрозы фиксируется:
    - цель
    - категория STRIDE
        - Spoofing – нарушение подлинности
        - Tampering – нарушение целостности
        - Repudation – нарушение ответственности
        - Information disclosure – нарушение конфидециальности
        - Denial of service – отказ в доступе
        - Elevation of privelege – поднятие полномочий
6. Оценить серьезность угроз и выбрать метод борьбы. Оценка угроз производится по методике DREAD:
    - Damage potential (что сломается)

- Reproducibility (воспроизводимость)
- Exploitability (используемость)
- Affected users (пострадавшие)
- Discoverability (возможность обнаружения)

Модель нарушителя

Уровни возможностей:

- низкий
- средний (запуск собственных средств)
- высокий (управление поведением системы)
- абсолютный (создатель)

Классификация хакеров:

- увлеченные
- профессионалы (зарабатывают деньги)

Мотивы увлеченных:

- развлечение
- слава
- недооцененность
- обучение
- доступ (бесплатный)

Модеди доступа:

1. Дискретная (HRU модель):

матрица доступа

	объекты	...
субъекты	Права (RWEX)	
...		

2. Ролевая модель

	R1	R2
S1	x	
S2	x	x

	01	02	03
R2	Права (RWEX)		
R2			

3. Мандатная модель

	ССОВ	СС	С	ΔСП
ССОВ				
СС				
СС				
ΔСП				

Доступ на чтение ко всему, что ниже, включая свой уровень. На запись – только свой уровень.

4. Модель совместного доступа

Лекция 4

<u>Шифрование</u> – это преобразование окрытого текста (plain text) в зашифрованный текст (cipher text) с целью защиты его конфиденциальности.

Основные принципы (принципы Кергхофа):

- сложность расшифровки
- малые изменения исходного текста влекут значительные изменения шифр-текста
- не требуется секретность системы
- секретность алгоритма определяется секретностью ключа
- область значений ключа должна исключать его перебор
- стоимость дешифрации должна быть выше стоимости данных

Алгоритмы шифрования:

- перестановки
- подстановки

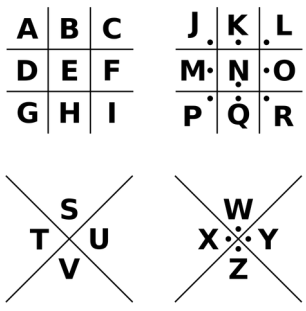
1. Скитала – древний метод шифрования. Представляет собой цилиндр и узкую полоску пергамента, на которой писалось сообщение, обматывавшуюся вокруг него по спирали. Цилиндров было два, абсолютно одинаковых — один у отправителя, другой у получателя. Автор секретного послания брал пергаментную ленту и наматывал её на цилиндр по восходящей (как на спираль) так, чтобы края тесно, но без нахлёстов примыкали друг к другу. И затем писал на витках сообщение — вдоль длинной стороны цилиндра. При размотке ленты текст превращался в бессмысленный набор букв. Прочитать депешу мог только адресат, наложив полоску тем же способом на идентичную скиталу.
2. Перестановка по правилу

Подстановки (замены):

1. Шифр Цезаря
   C = (ord(M) + S) mod |L|
   M – шифруемая буква
   S – сдвиг
   L – алфавит
2. Квадрат Полибия. Для шифрования на квадрате находили букву текста и вставляли в шифровку нижнюю от неё в том же столбце. Если буква была в нижней строке, то брали верхнюю из того же столбца.

А	Б	В	Г
Д	Е	Ё	Ж
…	…	…	...

3. Шифр PigPen. Есть не что иное, как геометрический шифр подстановки , в котором каждой букве алфавита ставится свое место в одной из четырех сеток.



4. Великий шифр Людовика 14
5. Таблица омофонов. каждый символ открытого текста заменяется на один из нескольких символов шифралфавита, причём количество заменяющих символов для одной буквы пропорционально частоте этой буквы. Это

позволяет скрыть настоящую частоту появления данной буквы в зашифрованном тексте.

Выше были рассмотрены одноалфавитные земены, где для преобразования каждого символа используется один и тот же алгоритм. Далее рассмотрены многоалфавитные замены.

6. Шифр Виженера. Метод полиалфавитного шифрования буквенного текста с использованием ключевого слова. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова.
   Ключевое слово записывается циклически до тех пор, пока его длина не будет соответствовать длине исходного текста.
   Если n – количество букв в алфавите, m_j – буквы открытого текста, k_j – буквы ключа, то шифрование Виженера можно записать следующим образом:
   $c_j = (m_j + k_j) \bmod n$
   И расшифровывание:
   $m_j = (c_j + n — k_j) \bmod n$.
7. Кодовая книга (до сих пор используется в военной отрасли). А → Й, Б → ...
8. Энигма
   Состоит из роторов и рефлекторов. Сам по себе ротор производил очень простой тип шифрования: элементарный шифр замены. Например, контакт, отвечающий за букву Е, мог быть соединён с контактом буквы Т на другой стороне ротора. Но при использовании нескольких роторов в связке (обычно трёх или четырёх) за счёт их постоянного движения получается более надёжный шифр. Ротор прокручивается после каждого зашифрованного символа и когда приходит обратно в исходное состояние, начинает прокручиваться следующий ротор.
   За последним ротором следует *рефлектор*. Рефлектор нужен, чтобы процесс шифрования был обратимым (А → В, В → А). Рефлектор позволил отказаться от расшифровывающей машины. Для расшифровки нужно лишь сбросить машину в исходное состояние.
   i → value – прямой ход
   value → i – обратный ход.

Лекция 5

Ассметричное шифрование

Проблема распространения ключей в симметричном шифровании. Эту проблему решили математическим путем.

Даффи — Хелман/Маркл

Главная концепция:

- алгоритм общедоступен
- ключ шифрование общедоступен
- ключ расшифровки является секретом
- обратное преобразование с помощью открытого ключа очень сложно

Алгоритм RSA (Ривест, Шамир, Эдлман)

- Разложение числа на простыне множетели – сложная задача

1. Вычисление ключей
   - Два произвольных простых числа P, Q

- N = P * Q, N – длина алфавита. Должно быть больше числа возможных значений блока данных.
- Функция Эйлера: $f_i = (P — 1)(Q — 1)$
- E – любое взаимно простое с f_i число. Открытый ключ состоит из пары (E, N)
- Поиск закрытого ключа D: (E * D) mod f_i = 1. Закрытый ключ есть комбинация (D, N).

2. Шифрование
   - $C = (M^E) \bmod N$
3. Расшифровка
   - $M' = (C^D) \bmod N$

Чтобы сделать алгоритм более криптостойким, нужно увеличить P и Q. Рекомендованая длина ключа – 1 Кб.

Герерация случайных простых чисел:

- Генерация случайного числа и проверка, что оно простое
- Все делители числа x лежат в диапазоне от 2 до x^2
- Решето Эратосфена
- Теорема Рабина
- Тест Рабина-Миллера

Алгоритм Евклида

Пусть a и b — целые числа, не равные одновременно нулю, и последовательность чисел

$a > b > r_1 > r_2 > \ldots > r_n$

определена тем, что каждое r_k – это остаток от деления предпредыдущего числа на предыдущее, а предпоследнее делится на последнее нацело, то есть

$a = bq_0 + r_1$

$b = r_1q_1 + r_2$

$r_1 = r_2 q_2 + r_3$

…

$r_{k-2} = r_{k-1}q_{k-1} + r_k$

$r_{n-2} = r_{n-1}q_{n-1} + r_n$

$r_{n-1} = r_nq_n$

Тогда НОД(a, b) равен r_n, последнему ненулевому члену этой последовательности.

Расширенный алгоритм Евклида

НОД(a, b) = $r_n = as + bt$

s, t – коэффициенты Безу.

$$
\begin{aligned}
&r_1 = a - bq_0, \\
&r_2 = b - r_1 q_1 = b - (a - bq_0)q_1 = b(1 + q_0 q_1) - aq_1, \\
&r_3 = r_1 - r_2 q_2 = (a - bq_0) - (b(1 + q_0 q_1) - aq_1)q_2 = a(1 + q_1 q_2) - b(q_0 + q_2 + q_0 q_1 q_2), \\
&\ldots \\
&r_n = r_{n-2} - r_{n-1} q_{n-1} = \cdots = ax + by.
\end{aligned}
$$

Внести вычисление этих коэффициентов в алгоритм Евклида несложно, достаточно вывести формулы, по которым они меняются при переходе от пары (a, b) к паре (b % a, b) (знаком процента мы обозначаем взятие остатка от деления).

$x = y_1 — [b/a]\, x_1$

$y = x_1$

Алгоритм быстрого возведения в степень по модулю

Основным алгоритмом быстрого возведения в степень является схема «слева направо». Она получила своё название вследствие того, что биты показателя степени просматриваются слева направо, то есть от старшего к младшему[5].

Пусть

$n = (\overline{m_k m_{k-1} \dots m_1 m_0})_2$ — двоичное представление степени n, то есть,

$n = m_k \cdot 2^k + m_{k-1} \cdot 2^{k-1} + \cdots + m_1 \cdot 2 + m_0,$

где $m_k = 1, m_i \in \{0, 1\}$. Тогда

$x^n = x^{((\dots((m_k \cdot 2 + m_{k-1}) \cdot 2 + m_{k-2}) \cdot 2 + \dots) \cdot 2 + m_1) \cdot 2 + m_0} = ((\dots(((x^{m_k})^2 \cdot x^{m_{k-1}})^2 \dots)^2 \cdot x^{m_1})^2 \cdot x^{m_0}$ [5].

Последовательность действий при использовании данной схемы можно описать так:

1. Представить показатель степени n в двоичном виде
2. Если m_i = 1, то текущий результат возводится в квадрат и затем умножается на x. Если m_i = 0, то текущий результат просто возводится в квадрат[6]. Индекс i изменяется от k-1 до 0[7].

Таким образом, алгоритм быстрого возведения в степень сводится к мультипликативному аналогу схемы Горнера[6]:

$$\left\{ \begin{array}{c} s_1 = x \\ s_{i+1} = s_i^2 \cdot x^{m_{k-i}} \\ i = 1, 2, \dots, k \end{array} \right\}.$$

Лекция 6

Хеш-функции

Свойства:

- длина H хеш-функции от произвольного сообщения M = const
- длина M сообщения – произвольная
- легкость вычисления
- необратимость
- два сообщения a != b => H(a) != H(b)

Коллизия одинаковых ключей (birthday-коллизия)

Атаки: birthday-атака

Назначение:

- аутентификация
- защита данных

Алгоритмы:

- семейство MD (MD5) – message digest
- семейство SHA (SHA0, SHA1, SHA2, SHA3) – secure hash algorithm

Лавинный эффект (avalanche effect) – зависимость всех выходных бит от каждого входного бита.

Лавинный эффект порядка X: 1 бит → X бит

Что приводит к лавинному эффекту:

- диффузия – реализация с помощью перестановок
- конфузия – реализация с помощью замен

Диффузия позволяет скрыть структуру исходного сообщения от семантического анализа. Конфузия устраняет зависимость от ключа.

Базовые алгоритмы:

- MD5 – был разработан в начале 90-х (1991) – признан небезопасным
  Вход: блоки по 128 биь
  Выход: хеш 160 бит
  В 1993 были найдены коллизии
- SHA0 – признан небезопасным
  Вход: блоки по 512 бит
  Выход: хеш 160 бит
- SHA1 – стандарт с 1995 года – устранение коллизий SHA0 – признан небезопасным
  Вход: 512 бит
  Выход: 160 бит

- SHA2 – семейство хеш-функций. SHA234, SHA256, SHA384, SHA512
  Вход: 512 / 1024 бит
  Выход: 234/256 / 384/512 бит
- SHA3 – имеет более длинные константы, чем SHA2 и имеет больше раундов

Главное примнение хеш-функций – электронная подпись.

Электронная подпись

Свойства собственноручной подписи:

- аутентичность (однозначно аутентифицирует подписанта)
- добровольное согласие
- неотказуемость подписавшегося
- скорость
- непереносимость (связь подписи и документа)
- целостность документа (документ нельзя изменять после его подписания)

Алгоритм формирования электронной подписи (секретный ключ):

- вычисление хеш-функции
- шифрование алгоритмом с открытым ключом (секретный ключ)

Проверка подписи:

- вход: данные, подпись (ЭП), открытый ключ
- вычисление хеш-функции данных – H2
- расшифровка подписи с открытым ключом. Получим хеш исходного сообщения – H1
- Сравнение H1 и H2

Электронная подпись (информация, подпись → позволяет определить лицо, подписавшее информацию) (63 – ФЗ).

Виды подписи:

- Простая – любое подтверждение, зафиксированное договором
- Усиленная – использование криптографических алгоритмов
  - Неквалифицированная – иностранные алгоритмы
  - Квалифицированная – аккредитация в госорганах

Архитектура клиентской системы:

- доверие к ОС
- Криптографические сервис-провайдеры (CSP) – CryptoAPI
- любая программа – CryptoAPI

Модели работы с ключами:

- децентрализованная (все со всеми обмениваются открытыми ключами)
- централизованная – Public Key Infrastructure – PKI. Централизованный орган обеспечивает распределение открытых ключей. Удостоверяющий центр (УЦ) – имеет корневой сертификат. Certification Authority.

Сертификат

X. 509 – стандарт. В сертификате хранится:

- версия и название алгоритма создания ЭП
- реквизиты (данные и ЭП) организации, выдавшей сертификат
- срок действия
- данные владельца
- открытый ключ владельца

Список отозванных сертификатов (Certificate Revocation List – CRL).

Лекция 7

Сжатие

Сокращение бит, информация — const.

Устранение избыточности.

Виды:

- замена часто встречающихся данных короткими кодами (энтропийное сжатие)
- замена повторяющихся фрагментов ссылками на уже имеющиеся данные (указание повторов)
- с потерей информации

Повторяющиеся последовательности символов:

11111177777776666644444 → [1,6][7,7][6,5][4,5]

Алгоритмы с деревьями:

- построение кодового дерева (таблица частот)
- построения отображения символ → код
- архивация

Код Шенона-Фано

Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано — префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

1. Символы первичного алфавита m_1 выписывают по убыванию вероятностей.

2. Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.

3. В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».

4. Полученные части рекурсивно делятся, и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Всё множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует концевая вершина кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, вообще говоря, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами, оптимальное поведение на каждом шаге пути ещё не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей.

Алгоритм Хафмана

Таблица частот P(L) - отсортировали по возрастанию. 2 узла с минимальной частотой L0 и L1-> новый узел L(1+2).

В отличие от алгоритма Шеннона — Фано, алгоритм Хаффмана остаётся всегда оптимальным. Этапы:

- Построение оптимального кодового дерева.
- Построение отображения код-символ на основе построенного дерева.

Адаптивный Хафман

- Инициализация дерева
- увеличение веса считанного символа P(Mi)++
- перестановка узлов дерева

- по упорядоченному (по весу) списку узлов ищем первый узел с большим весом
- меняете местами найденный узел с Mi
- повторить для родителя Mi (увеличив вес)

Адаптивное сжатие позволяет не передавать модель сообщения вместе с ним самим и ограничиться одним проходом по сообщению как при кодировании, так и при декодировании.

Все алгоритмы перестроения дерева при считывании очередного символа, включают в себя две операции:

Первая — увеличение веса узлов дерева. Вначале увеличиваем вес листа, соответствующего считанному символу, на единицу. Затем увеличиваем вес родителя, чтобы привести его в соответствие с новыми значениями веса потомков. Этот процесс продолжается до тех пор, пока мы не доберемся до корня дерева. Среднее число операций увеличения веса равно среднему количеству битов, необходимых для того, чтобы закодировать символ.

Вторая операция — перестановка узлов дерева — требуется тогда, когда увеличение веса узла приводит к нарушению свойства упорядоченности, то есть тогда, когда увеличенный вес узла стал больше, чем вес следующего по порядку узла. Если и дальше продолжать обрабатывать увеличение веса, двигаясь к корню дерева, то дерево перестанет быть деревом Хаффмана.

Чтобы сохранить упорядоченность дерева кодирования, алгоритм работает следующим образом. Пусть новый увеличенный вес узла равен W+1. Тогда начинаем двигаться по списку в сторону увеличения веса, пока не найдем последний узел с весом W. Переставим текущий и найденный узлы между собой в списке, восстанавливая таким образом порядок в дереве (при этом родители каждого из узлов тоже изменятся). На этом операция перестановки заканчивается.

Алгоритм LZW

- Заполнить словарь символами ASCII
- w=M[0]
- k=M[i]
- ? есть w+k есть в словаре, то замена на код (в префикс)
- заносим в словарь

```
код       0 1 2 3 4 ... 255 | 256 257  258   259 .... 511 (граница 9 бит)
префикс - - - - -...  - - -  | 21   70   176  ~~21~~256
суффикс 0 1 2 3 4 ... 255 | 70   176  21   ~~70~~

   Разархивация
   21   70   176  256

   код   0 1 2 3 4 ... 255 | 256 257  258   259 .... 511 (граница 9 бит)
префикс - - - - -...  - - -  | 21   70   176       21
суффикс 0 1 2 3 4 ... 255 |  70   176  ~~256~~ 21   70
```

1. Инициализация словаря всеми возможными односимвольными фразами. Инициализация входной фразы W первым символом сообщения.

2. Если КОНЕЦ_СООБЩЕНИЯ, то выдать код для W и завершить алгоритм.

3. Считать очередной символ K из кодируемого сообщения.

4. Если фраза WK уже есть в словаре, то присвоить входной фразе W значение WK и перейти к Шагу 2.

5. Иначе выдать код W, добавить WK в словарь, присвоить входной фразе W значение K и перейти к Шагу 2.

Алгоритму декодирования на входе требуется только закодированный текст: соответствующий словарь фраз легко воссоздаётся посредством имитации работы алгоритма кодирования.

Арифметическое сжатие

Оин из алгоритмов энтропийного сжатия.

Пусть имеется некий алфавит, а также данные о частотности использования символов (опционально). Тогда рассмотрим на координатной прямой отрезок от 0 до 1.

Назовём этот отрезок рабочим. Расположим на нём точки таким образом, что длины образованных отрезков будут равны частоте использования символа, и каждый такой отрезок будет соответствовать одному символу.

Теперь возьмём символ из потока и найдём для него отрезок среди только что сформированных, теперь отрезок для этого символа стал рабочим. Разобьём его таким же образом, как разбили отрезок от 0 до 1. Выполним эту операцию для некоторого числа последовательных символов. Затем выберем любое число из рабочего отрезка. Биты этого числа вместе с длиной его битовой записи и есть результат арифметического кодирования использованных символов потока.

- L=0 R=1
- **a** c d....
- L=0 R=0,6
- L=0,48 R=0,54
- L=

a: P(a)=60% - (0; 0,6]
b: P(b)=20% -(0,6; 0,8]
c: P(c)=10% - (0,8; 0,9]
d: P(d)=10% - (0,9; 1]

c-> l = R-L
L = L+l*Li = 0,48+0,06*0,9 =0,534
R = L+l*Ri = 0,48+0,06*1 = 0,54

код - 0,539 -> a
l=Ra-La
код1 = (код - La)/l =(0,539-0)/0,6=0,89 -> c

код2=(код1 - Lc)/0,1=(0,89-0,8)/0,1=0,9 -> d

Алгоритм DES

DES (англ. *Data Encryption Standard*) — алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт.

Размер блока для DES равен 64 битам. В основе алгоритма лежит сеть Фейстеля с 16 циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

Шаги:

1. Начальная перестановка. Исходный текст (блок 64 бит) преобразуется с помощью начальной перестановки , которая определяется таблицей IP.

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

Основная функция шифрования (функция Фейстеля) [править | править код]

Аргументами функции f являются 32-битовый вектор R_{i-1} и 48-битовый ключ k_i, который является результатом преобразования 56-битового исходного ключа шифра k. Для вычисления функции f последовательно используются

1. функция расширения E,
2. сложение по модулю 2 с ключом k_i
3. преобразование S, состоящее из 8 преобразований S-блоков $\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3 \ldots \mathrm{S}_8$,
4. перестановка P.

Функция E расширяет 32-битовый вектор R_{i-1} до 48-битового вектора $\mathrm{E}(R_{i-1})$ путём дублирования некоторых битов из R_{i-1}; порядок битов вектора $\mathrm{E}(R_{i-1})$ указан в таблице 2.

Таблица 2. Функция расширения E

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

Первые три бита вектора $\mathrm{E}(R_{i-1})$ являются битами 32, 1, 2 вектора R_{i-1}. По таблице 2 видно, что биты 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 32 дублируются. Последние 3 бита вектора $E(R_{i-1})$ — это биты 31, 32, 1 вектора R_{i-1}. Полученный после перестановки блок $\mathrm{E}(R_{i-1})$ складывается по модулю 2 с ключами k_i и затем представляется в виде восьми последовательных блоков $B_1, B_2, \ldots B_8$.

$$\mathrm{E}(R_{i-1}) \oplus k_i = B_1 B_2 \ldots B_8$$

Каждый B_j является 6-битовым блоком. Далее каждый из блоков B_j трансформируется в 4-битовый блок B'_j с помощью преобразований S_j. Преобразования S_j определяются таблицей 3.

Предположим, что $B_3 = 101111$, и мы хотим найти B'_3. Первый и последний разряды B_3 являются двоичной записью числа a, 0<=a<=3, средние 4 разряда представляют число b, 0<=b<=15. Строки таблицы S3 нумеруются от 0 до 3, столбцы таблицы S3 нумеруются от 0 до 15. Пара чисел (a, b) определяет число, находящееся в пересечении строки a и столбца b. Двоичное представление этого числа дает B'_3 . В нашем случае $a = 11_2 = 3$, $b = 0111_2 = 7$, а число, определяемое парой (3,7), равно 7. Его двоичное представление B'_3=0111. Значение функции $f(R_{i-1}, k_i)$ (32 бит) получается перестановкой P, применяемой к 32-битовому блоку $B'_1 B'_2 \ldots B'_8$. Перестановка P задана таблицей 4.

Таблица 4. Перестановка P

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

$$f(R_{i-1}, k_i) = P(B'_1 B'_2 \ldots B'_8)$$

Согласно таблице 4, первые четыре бита результирующего вектора после действия функции f — это биты 16, 7, 20, 21 вектора $B'_1 B'_2 \ldots B'_8$

Генерирование ключей k_i [править | править код]

Ключи k_i 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.

Таблица 5.

57	49	41	33	25	17	9	1	58	50	42	34	26	18	C_0
10	2	59	51	43	35	27	19	11	3	60	52	44	36	
63	55	47	39	31	23	15	7	62	54	46	38	30	22	D_0
14	6	61	53	45	37	29	21	13	5	28	20	12	4	

Эта перестановка определяется двумя блоками C_0 и D_0 по 28 бит каждый. Первые 3 бита C_0 есть биты 57, 49, 41 расширенного ключа. А первые три бита D_0 есть биты 63, 55, 47 расширенного ключа. C_i, D_i i=1,2,3...получаются из C_{i-1}, D_{i-1} одним или двумя левыми циклическими сдвигами согласно таблице 6.

Таблица 6.

i	**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

Ключ k_i, i=1,...16 состоит из 48 бит, выбранных из битов вектора $C_i D_i$ (56 бит) согласно таблице 7. Первый и второй биты k_i есть биты 14, 17 вектора $C_i D_i$

Таблица 7.

14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2	41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

Конечная перестановка [править | править код]

Конечная перестановка IP^{-1} действует на T_{16}^{-1} (где $T_{16}^{-1} = R_{16} + L_{16}$) и является обратной к первоначальной перестановке. Конечная перестановка определяется таблицей 8.

Таблица 8. Обратная перестановка IP^{-1}

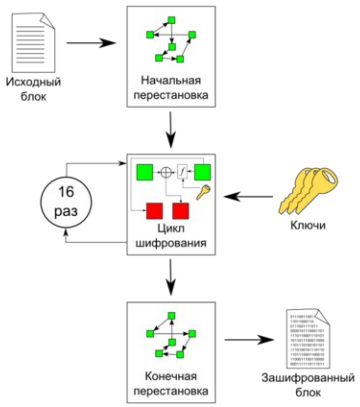
40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

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

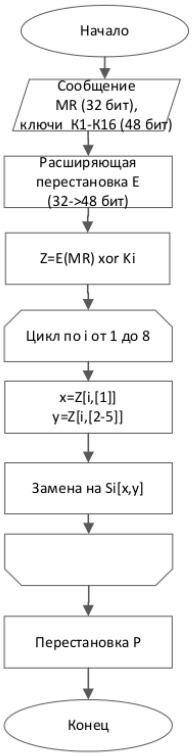
$R_{i-1} = L_i$
$L_{i-1} = R_i \oplus f(L_i, k_i)$

Схема расшифрования указана на Рис.6.

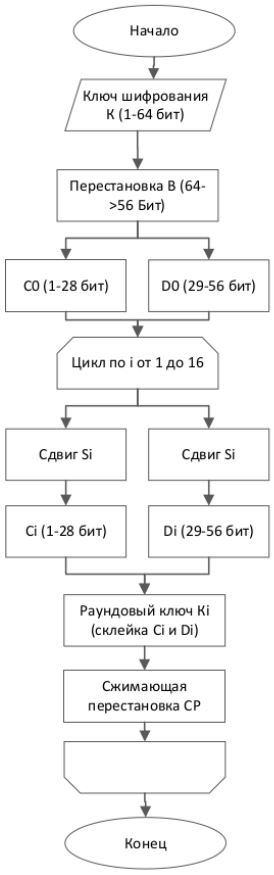
Ключ k_i, i=16,...,1, функция f, перестановка IP и IP^{-1} такие же, как и в процессе шифрования. Алгоритм генерации ключей зависит только от ключа пользователя, поэтому при расшифровании они идентичны.



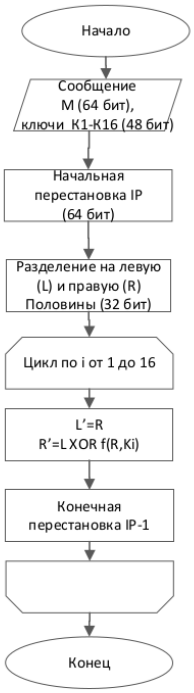
Шифр Фейстеля



Генерация раундовых ключей



Шифрование



Расшифровка

Алгоритм AES

Симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США.