МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических занятий
по дисциплине «МЕТОДЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ»
для студентов специальности 10.05.01 Компьютерная безопасность
специализация «Информационно-аналитическая и техническая экспертиза компьютерных систем»

Ставрополь
2026

СОДЕРЖАНИЕ

Практическая работа №1 Общие сведения о криптографии

Цель работы:

Создать криптографическую систему шифрования данных, которая базируется на алгоритме шифровании *DES*. Алгоритм *DES* является первым симметричным алгоритмом блочного шифрования данных.

Теоретические сведения:

Основные этапы алгоритма DES при шифровании текста:

Генерируется (задается) случайная последовательность Q из 56 бит.

В последовательность Q, для контроля четности, добавляются восемь контрольных битов в позиции

8, 16, 24, …, 64.

Получается блок U размером в 64 бита.

Для удаления контрольных битов из блока U и формирования ключа K для шифрования, блок U преобразуют, используя функцию G(U). Функция G определяется в виде стандартной таблицы, которую надо использовать в неизменном виде. В результате преобразования получают блок

K=G(U)

размером в 56 бит. Блок K разбивают на две половины C0 и D0 по 28 бит.

Используя C0 и D0, последовательно определяются Ci и Di , i = 1, 2, …, 16. Для формирования Ci и Di применяют операции циклического сдвига влево на один или два бита. Величина сдвига определяется стандартной таблицей. Операции сдвига для Ci и Di выполняются независимо. Последовательность C5 получается из C4 посредством циклического сдвига влево на 2 бита, а D5 – посредством циклического сдвига влево на 2 бита D4 (см. таблицу сдвигов для вычисления ключа). В результате четвертого этапа формируется 16 ключей для 16 раундов алгоритма DES.

Завершающий этап формирования ключей. Используя перестановку H, которая задается стандартной таблицей, каждый ключ размером в 56 бит преобразуется в блок из 48 бит. Данным преобразованием завершается этап формирования ключей.

Текущий блок открытого текста размером в 64 бит, представленный в виде двух 32 битовых блоков (L0R0) преобразуется начальной перестановкой IP, которая задается фиксированной стандартной таблицей.

Выполняется 16 раундов преобразований по следующим формулам

Li = Ri-1,

Ri = Li_$\oplus$f(Ri-1 ,Ki), i = 1.2, …, 16.

Функция f(Ri-1, Ki) представляет собой некоторую суперпозицию простых преобразований, детальное описание которых будет дано ниже. Отметим, что функция f(Ri-1, Ki) является нелинейной. Нелинейность функции f(Ri-1, Ki) обеспечивается S-блоками.

Алгоритм вычисления функции шифрования f(Ri-1, Ki)

К блоку Ri-1 применяют функцию расширения E(Ri-1), которая 32битовый блок Ri-1 преобразовывает в блок R´ = E(Ri-1) размером в 48 бит. Функция E(Ri-1) определяется стандартной таблицей. Алгоритм вычисления функции шифрования формулируется следующим образом.

Блок B размером в 48 бит разбивается на восемь блоков Bj, j=1,2,…, 8, по шесть битов каждый

B = B1B2B3B4B5B6B7B8.

Каждый блок Bj размером в шесть бит, используя свою функцию Sj , преобразуется в блок Sj(Bj) размером в четыре бита. В результате получается следующий блок данных

B′ = S1(B1) S2(B2) S3(B3) S4(B4) S5(B5) S6(B6) S7(B7) S8(B8)

размером в 32 бита.

Алгоритм преобразования блока Bj размером в шесть бит в блок Sj(Bj) размером в четыре бита следующий. Каждая функция Sj представляет собой стандартную таблицу, которая состоит из четырех строк с номерами 0, 1, 2, 3, и шестнадцати столбцов с номерами 0,1,2,…,15. Пусть, например, некоторый блок

Bj = b1 b2 b3 b4 b5 b6 = 110010, j = 1, 2, …, 8.

Тогда имеет место:

биты b1b6 = 10 формируют номер столбца (двоичное число 10 равно

десятичному числу 2) таблицы Sj, биты b2 b3 b4 b5 = 1001 формируют номер строки (двоичное число 1001 равно десятичному числу 9) таблицы Sj. Блок Bj = b1 b2 b3 b4 b5 b6 = 110010 заменяют двоичным значением числа таблицы Sj, которое находится на пересечение строки с номером b1b6 = 10 (2) со столбцом с номером b2 b3 b4 b5 = 1001 (9). Преобразуя каждое Bj, j = 1, 2, …, 8, из блока B, получим новый блок B′ размером в 32 бита.

Задания:

Программная реализация криптографической системы, основанной на алгоритме шифрования *DES*, должна быть оформлена как некоторая программная оболочка. В программной реализации должен быть разработан интерфейс, удобный для эксплуатации программы, в интерфейсе следует предусмотреть:

- два режима формирования ключа – ключ задан, ключ формируется по умолчанию;
- ввод начальной информации из сформированного заранее файла и из файла, который создается в оболочке программы;
- режимы шифрования, которые предусмотрены в DES;
- режимы шифрования и дешифрования информации.

Подготовить отчет по работе. В отчете описать алгоритм DES, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций. В отчет включить описание алгоритма генерации ключа, детали программной реализации, которые представляют интерес с точки зрения разработчика.

Контрольные вопросы:

1. По какому принципу построен блочный шифр DES?
2. Какой шифр предшествовал шифру DES?
3. Указать длину начального ключа в алгоритме DES.
4. Перечислить основные этапы формирование ключей в алгоритме DES.
5. Сколько раундов в алгоритме DES?
6. Сколько ключей формируется в алгоритме DES для шифрования текста?
7. Указать длину ключа при шифровании текста в алгоритме DES.
8. Сколько S-функций определено в алгоритме DES?
9. Какая арифметическая операции используется при преобразованиях в алгоритме DES.
10. Укажите длину шифруемого блока в алгоритме DES.

Практическая работа №2 Формализация криптографических методов

Цель работы:

Создать учебный вариант криптографической системы с открытым ключом на алгоритме *RSA*, которая является первой из криптосистем с открытым ключом.

Теоретические сведения:

Выбираем два различных простых числа p и q.

Вычисляем n = pq, $\varphi(n)=(p-1)(q-1)$.

Выбираем число e, взаимно простое с $\varphi(n)$. 4. Вычисляем число d из уравнения $de \equiv 1 \bmod \varphi(n)$.

Определяются открытые ключи e и n.

Определяются закрытые ключи d, p, q и $\varphi(n)$.

Алгоритм шифрования

Дан текст сообщения M.

Шифрованный текст C вычисляется по формуле

$C=EK(M)=M^e \bmod n$.

Алгоритм дешифрования

Дан шифрованный текст C.

Текст сообщения M вычисляется по формуле

$M=DK(C)=C^d \bmod n \equiv (M^e)^d \bmod n \equiv M$

Задания:

Программная реализация должна быть оформлена как некоторая программная оболочка. В программной реализации должен быть разработан интерфейс, удобный для эксплуатации программы, в интерфейсе предусмотреть режим задания параметров системы по умолчанию и режим генерирования параметров системы. В программу включить простейший алгоритм формирования простых чисел и проверки чисел на простоту. Оформить программы проверки чисел на простоту как самостоятельные модули, чтобы при необходимости эти модули можно было заменить на другие программные реализации. Подготовить отчет по работе. В отчет включить описание алгоритма RSA и алгоритм проверки чисел на простоту. Подготовить для демонстрации программы контрольный пример. В работу включить программы:

- формирование простого числа;
- проверки является ли число простым;
- программу, которая решает уравнение сравнения $ax = b \pmod n$, используя расширенный алгоритм Эвклида.

Контрольные вопросы:

1. Дать определение односторонней функции.
2. Дать определение односторонней функции с секретом.
3. На какой базе обычно создаются криптографические системы с открытыми ключами?
4. Перечислить число параметров в криптографической системе RSA.
5. Перечислить секретные параметры системы RSA.
6. Перечислить открытые параметры системы RSA.
7. На каком математическом результате базируется система RSA.
8. Какими свойствами должны удовлетворять параметры p и q системы RSA.
9. Какими свойствами должны удовлетворять параметры e и d системы RSA.
10. На какой достаточно трудной задаче из теории чисел базируется криптографическая система RSA.
11. Опишите схему шифрования текста с использованием алгоритма RSA.

12. Опишите схему дешифрования текста с использованием алгоритма RSA.
13. Опишите схему формирование цифровой подписи с применение алгоритма RSA.
14. Сформулировать теорему Ферма.
15. Сформулировать теорему Эйлера.

Практическая работа №3 Основные термины и определения

Цель работы:

Создать программу, которая вычислять профиль исходного текста. В дальнейшем программу можно использовать для выполнения лабораторной работы при реализации электронной цифровой подписи (ЭЦП).

Теоретические сведения:

Основные этапы алгоритма *MD5* при сжатии исходного текста

Добавляются биты к исходному сообщению. Если L – длина сообщения, то для L должна выполняться формула

$L \equiv 448 \ mod \ 512$.

Это означает, что длина дополненного сообщения в битах должна быть на 64 бита меньше, чем ближайшее кратное целого числа 512. Структура дополнения следующая: первый бит дополняемого значения равен 1, а остальные – равны 0.

К сообщению из шага 1 добавляется значение длины. К результату шага 1 добавляется 64-битовое значение длины исходного сообщения, т.е.

длины сообщения до шага 1.

Результат шага 2 разбивается на блоки по 512 бит

$M1, M2 \ \ldots, MK$,

где K число таких блоков, $L = K*512$. В свою очередь каждый блок $M2$, $i=1,2, \ldots,K$, разбивают на 16 массивов по 32 бита

$X[0], X[1], \ldots, X[k], k = 0, 1, \ldots, 15$.

Для записи промежуточных значений в 128 бит при вычислении хэш-функции используются буфер H, который состоит из четырех переменных *a, b, c* и *d* размером в 32 бита каждая. Эти переменные соответственно инициализируются ранее определенными константами *A, B, C* и *D*.

Преобразуются блоки M_i, $i = 1, 2, \ldots, K$, в 512 бит в профиль H_i длиной в 128 бит. Напомним, что блок M_i разбит на 16 частей (массивов) по 32 бита $X[0], X[1], \ldots, X[k], k = 0, 1, \ldots, 15$.

Основные этапы алгоритма *SHA*-1 при сжатии исходного текста

Добавление битов заполнителя. Сообщение дополняется так, чтобы его общая длина в битах была сравнима со значением 448 по модулю 512, т.е. длина равна 448 *mod* 512. Операция дополнения выполняется всегда, даже если сообщение уже оказывается требуемой длины. Поэтому число добавляемых битов должно быть от 1 до 512. Структура битов заполнителя следующая: первый бит равен 1, а все остальные равны 0.

К сообщению из шага 1 добавляется значение длины. К результату шага 1 добавляется 64-битовое значение длины исходного сообщения, т.е. длины сообщения до шага 1.

Результат шага 2 разбивается на блоки по 512 бит: $M1, M2 \ \ldots, MK$, где K есть число таких блоков, $L = K \times 512$. В свою очередь каждый блок M_i разбивают на 16 частей (массивов) по 32 бита $X[0], X[1], \ldots, X[k], k = 0, 1, \ldots, 15$.

Для записи промежуточных значений в 160 бит при вычислении хэшфункции используются буфер H, который состоит из пяти переменных *a, b, c, d* и *e* размером в 32

бита каждая. Эти переменные соответственно инициализируются ранее определенными константами *A, B, C, D* и *E*.

Обработка сообщения происходит 512-битовыми блоками. Логику модуля функции сжатия алгоритма можно представить в виде следующего цикла

Hi=Hi-1+H

Цикл от t=0 *до t*=79 *шаг* 1

Начало цикла

$H = (a <<< 5) + f_t(b,c,d) + e + W_t + K_t \quad e = d \quad d = c \quad c = b <<< 30$

$b = a$

$a = H$ Конец цикла

Hi+1=Hi+H

Здесь:

$f_t(b,c,d)$, W_t , K_t – функции, 32-битовые блоки и константы, которые определялись ранее. Отметим, что W_t формируется из текущего входного блока M_i.

H_i – промежуточные значения результата вычисления хэш-функции. + – операция сложения по модулю 2^{32}.

Задания:

Программная реализация должна быть оформленная как некоторая программная оболочка, которая включает два алгоритма *MD*5 и *SHA*-1 для вычисления профиля начальной информации. Подготовить отчет по работе. В отчете описать алгоритмы *MD*5 и *SHA*-1, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций. Подготовить для демонстрации программы контрольный пример.

Контрольные вопросы:

1. Основное назначение хэш-функции.
2. Перечислить класс задач, которые решаются с применением хэшфункции.
3. Перечислить, каким основным свойствам должна удовлетворять хэшфункция.
4. Длина входного блока в битах для функции сжатия алгоритма *MD*5.
5. Назвать число раундов в алгоритме *MD*5.
6. Назвать число шагов в каждом раунде алгоритма *MD*5.
7. Назвать, какие операции используются в функции сжатия алгоритма *MD*5.
8. Перечислить основные постоянные данные, которые используются в алгоритме *MD*5.
9. Сколько примитивных функций используются в алгоритме *MD*5?
10. Сколько элементов в массиве *T*, который используется в алгоритме *MD*5?
11. Назвать, какие операции используются в функции сжатия алгоритма *SHA*-1.
12. Перечислить основные постоянные данные, которые используются в алгоритме *SHA*-1.
13. Назвать число раундов в алгоритме *SHA*-1.
14. Сколько примитивных функций используются в алгоритме *SHA*-1.
15. Длина входного блока в битах для функции сжатия алгоритма *SHA*-1.
16. Длина буфера в битах для функции сжатия алгоритма *SHA*-1.
17. Число шагов в каждом раунде алгоритма *SHA*-1.
18. Число регистров буфера в алгоритме *SHA*-1.
19. Чем инициализируется регистры буфера алгоритма *SHA*-1?
20. Чем инициализируется регистры буфера алгоритма *MD*5?

Практическая работа №4 Основные требования, предъявляемые к криптосистемам.

Цель работы:

Создать простую программу, которая генерирует псевдослучайную последовательность.

Задания:

Для вычисления ПСП использовать ранее созданную программу шифрование DES в режиме счетчика (*Counter*). Реализовать конгруэнтные генераторы

$x_n+1 = (ax_n + b) \bmod m$;

$x_n+1 = (ax_n^2 + bx_n + c) \bmod m$; $x_n+1 = (ax_n^3 + bx_n^2 + cx_n + d) \bmod m$.

Здесь x_n – n-й член последовательности, x_n+1 – следующий член последовательности.Параметры a, b, c и d представляют собой параметры конгруэнтных генераторов, m – модуль.

Подготовить отчет по работе. В отчет включить блок схему режима счетчика и продемонстрировать работу генераторов на контрольном примере.

Контрольные вопросы:

1. Привести задачи криптографии, в которых используются псевдослучайные последовательности.
2. Перечислить требования, которым должны удовлетворять генераторы псевдослучайных последовательностей.
3. Перечислить криптографические алгоритмы, на базе которых можно создавать генераторы псевдослучайных последовательностей.
4. Можно ли использовать линейные конгруэнтные генераторы ПСП в криптографии.
5. Можно ли использовать шифры с открытыми ключами для генерирования псевдослучайных последовательностей.
6. Можно ли использовать блочные алгоритмы шифрования с симметричными ключами для генерации ПСП.

Практическая работа №5 Криптографическая стойкость шифров

Цель работы:

Создать программу, которая реализует учебный вариант схем ЭЦП, используя алгоритмы с открытыми ключами.

Теоретические сведения:

Алгоритм формирования схемы Эль-Гамаля

Выбираем простое число p.

Выбираем два случайных числа $g < p$ и $x < p$. 3. Вычисляем

$y \equiv g^x \ mod \ p$.

Открытым ключом схемы Эль-Гамаля являются числа y, g и p. Закрытым ключом – число x.

Алгоритм формирования цифровой подписи

Дано сообщение M, которое надо подписать.

Выбираем случайное число $k < p$ взаимно простое с p-1. 3. Вычисляем

$a \equiv g^k \ mod \ p$.

Из уравнения

$M = (xa+kb) \ mod \ (p\text{-}1)$

определяем b

$b= (M\text{-}xa)k^{-1} \text{mod} \ (p\text{-}1)$.

Формируем подпись. Подписью является пара чисел (a,b).

Число k является секретным ключом.

Замечание. При формировании цифровой подписи на шаге 4 вместо значения M можно использовать значение $\mu = H(M)$, где H – некоторая хешфункция.

Проверка подписи

Дано сообщение M и подпись (a,b).

Вычисляем

$y^a a^b \ mod \ p \equiv [g^x]^a a^b \equiv g^{xa} g^{kb} \equiv \equiv g^{xa+kb} \equiv g^M \ mod \ p$.

1. Вычисляем

$g^M \text{ mod p}$.

Если значение $y^a a^b \text{ mod p}$

совпало с $g^M \text{ mod p}$, то подпись верна.

Цифровая подпись на базе алгоритма *RSA*

Есть абонент A и текст для подписи M.

Определяются закрытые ключи системы *RSA*, т.е. d, p, q и $\varphi(n)$.

Определяются открытые ключи e и n. 4. Закрытым ключом вычисляется

$C=M^d \ mod \ n$.

Сообщение C рассматривается как подпись абонента A, т. к. закрытый ключ d известен только ему.

5. Используя открытый ключ e, проверка подписанного документа вычисляется по формуле $C^e = (M^d)^e \ mod \ n \equiv M$,

Цифровая подпись Шнорра

ЭЦП Шнорра основана на сложности задачи дискретного логарифмирования. Поэтому все параметры схемы Шнорра должны удовлетворять условиям существование дискретного логарифма.

Схема Шнорра

Параметры схемы:

p – простое число, для реальных задач $160 \leq p \leq 256$, q – простое число, $q|p$-1, т.е. q делит p-1,

g – число g, $g \in Zp$, где Zp – класс вычетов по модулю p, h – хэш-функция, x – случайное число из интервала $[1,q\text{-}1]$, x – секретный ключ схемы.

y – открытый ключ схемы,

$y = g^{-x}$.

Предположим, что существуют два участника A и B. Участник A должен подписать сообщение m для участника B.

Алгоритм схемы подписи Шнорра

Участник A выбирает случайное число k и вычисляется

$r = g^k \bmod p$.

Участник A формирует подпись, для этого вычисляются e и s по формулам

$e = h(r, A),\ s = k + xe$.

Подпись (e, s) и подписываемый текст m пересылается участнику B. 4. Участник B делает проверку подписи, вычисляя значения r' и e'

$r' = g^s y^e \bmod p,\ e' = h(r', m)$.

5. Если $e=e'$, то подпись принимается, в противном случае отвергается.

Цифровая подпись Рабина

Подпись Рабина базируется на криптографической системе с открытым ключом. Эта система основана на сложности вычисления квадратных корней по модулю n. При этом модуль n представляет собой произведение двух больших простых чисел. Параметры криптографической схемы Рабина:

– простое число, $p \equiv 3 \bmod 4$, p – секретный ключ; q – простое число, секретный ключ $\equiv 3 \bmod 4$,

$n = pq$ – открытый ключ криптографической системы.

Алгоритм криптографической системы Рабина

Пусть дано сообщение M, $M < n$, которое надо шифровать.

Участник A вычисляет значение C, которое является шифром сообщения M, по формуле

$C = M^2 \bmod n$.

Сообщение C отправляется адресату B, который является владельцем закрытого, секретного ключа.

Используя китайскую теорему об остатках и закрытый ключ, участник B вычисляет

$m1 \equiv C(p+1)/4 \bmod p$,

$m2 \equiv (p - C^{(p+1)/4}) \bmod p$, $m3 \equiv C(q+1)/4 \bmod q$, $m4 \equiv (q - C^{(q+1)/4}) \bmod q$.

Используя закрытый ключ, участник B вычисляет числа a и b по формулам

$a = q(q^{-1} \bmod p),\ b = p(p^{-1} \bmod q)$.

Возможными сообщениями, которые были отправлены, являются значения

$M1$, $M2$, $M3$ и $M4$,

которые вычисляются по формулам

$M1 = (am1 + bm3) \bmod n$,

$M2 = (am1 + bm4) \bmod n$,

$M3 = (am2 + bm3) \bmod n$, $M4 = (am2 + bm4) \bmod n$.

Если сообщение представляет собой осмысленный текст, то выбрать правильное M_i, $i = 1, 2, 3, 4$, легко. Если сообщение представляет собой набор битов, т.е. какое-то число, то к такому виду передаваемой информации добавляется какой-либо осмысленный текст заголовка. По заголовку и происходит выбор M_i, i=1,2,3,4. Если M число, то другого способа идентификации передаваемого сообщения M не существует.

Перейдем теперь к описанию алгоритма цифровой подписи Рабина.

Предположим, что определены параметры криптографической системы Рабина и сообщение M, которое надо подписывать. Сообщение M представляет собой некоторую последовательность

$m0, m1, m2, \ldots, mk$-1

из k бит

$mi \in \{0,1\}$, $i = 0, 1, \ldots, k\text{-}1$.

Алгоритм схемы подписи Рабина

Формирование подписи

Генерируется случайная последовательность R длиной из t бит

$R = (r0, r1, \ldots, rt\text{-}1)$, $ri \in \{0,1\}$, $i = 0, 1, \ldots, t\text{-}1$.

Сообщение M объединяется с последовательностью R

$M||R = (m0, m1, m2, \ldots, mk\text{-}1, r0, r1, \ldots, rt\text{-}1\ )$.

Последовательности битов $M||R$ ставится в соответствии число $a < n$.

Проверяются условия

$a(p\text{-}1)/2 \equiv 1 \ mod\ p$, $a(q\text{-}1)/2 \equiv 1 \ mod\ q$.

Если условия не выполняется, то перейти на шаг 1.

Вычисляются значения

$zp \equiv a(p+1)/4 \ mod\ p$, $zp \equiv a(q+1)/4 \ mod\ q$.

По китайской теореме об остатках вычисляется значение

$Z = a^{1/2} \ mod\ n$.

Формируется подпись из пары чисел (R, Z) к сообщению M.

Проверка подписи

Вычисляется значение a'

$a' \equiv Z^2 \ mod\ n$.

Если $a=a'$, то подпись принимается, в противном случае отвергается.

Задания:

Реализовать ЭЦП на базе алгоритма Эль-Гамаля и алгоритма *RSA*. При формировании ЭЦП на базе алгоритма *RSA* использовать результаты лабораторной работы № 2. Предусмотреть режимы формирования параметров криптосистемы Эль-Гамаля. Программу оформить, как интегрируемую среду с удобным интерфейсом формирования ЭЦП и ее проверки. Подготовить отчет, в который включить алгоритмы формирования системы Эль-Гамаля, описание функций из которых состоит программа. Подготовить для демонстрации контрольный пример. При формировании цифровой подписи предусмотреть схему ЭЦП с использованием хэш-функций (см. лабораторную работу № 3).

Контрольные вопросы:

1. Перечислить число параметров в криптографической системе ЭльГамаля.
2. Перечислить секретные параметры системы Эль-Гамаля.
3. Перечислить открытые параметры системы Эль-Гамаля.
4. На какой достаточно трудной задаче из теории чисел базируется криптографическая система Эль-Гамаля.
5. Описать схему формирования ЭЦП с использованием алгоритма ЭльГамаля.
6. Описать схему проверку ЭЦП с использованием алгоритма ЭльГамаля.
7. Описать схему формирование цифровой подписи с применение алгоритма *RSA*.
8. Описать схему проверки цифровой подписи с применение алгоритма *RSA*.
9. Что общего между обычной и цифровой подписями? Чем они различаются?
10. Какие задачи позволяет решить цифровая подпись?
11. В чем заключается принципиальная сложность в практическом применении систем цифровой подписи?
12. Почему в криптографических системах, основанных на открытых ключах, нельзя использовать для шифрования и цифровой подписи?
13. Проверить, что указанный в тексте способ подбора подписанных сообщений для схемы Эль-Гамаля действительно дает верные цифровые подписи.

Практическая работа №6 Стойкость криптографических систем

Цель работы:

Ознакомится с криптографическими протоколами, которые в настоящее время широко используются для обеспечения информационной безопасности. Освоить основные понятия, которые связаны с криптографическими протоколами.

Теоретические сведения:

Схема Диффи - Хеллмана

Протокол формирования общего ключа по открытому каналу связи

Есть два пользователя (абонента) A и B.

Дано простое число p, которое известно пользователям A и B.

Выбирается общее число g, $g < p$-1.

Абоненты A и B независимо друг от друга выбирают соответственно секретные ключи a, $0 < a < p$-1 и b, $0 < b < p$-1.

Абонент A посылает абоненту B сообщение

$g^a \ mod \ p$.

Абонент B посылает абоненту A сообщение

$g^b \ mod \ p$.

Абонент A вычисляет общий ключ

$k \equiv [g^b]^a \ mod \ p$.

Абонент B вычисляет общий ключ

$k \equiv [g^a]^b \ \text{mod} \ p$.

Протокол взаимной аутентификации

Существует несколько схем аутентификации источника данных ([1], [7]). Приведем условия и алгоритм реализации одной из них.

Даны два абонента A и B.

Даны простое число p и число g, $g<p$-1, которые известны абонентам A и B.

Абоненты A и B владеют соответственно открытыми функциями шифрования E_A и

E_B и закрытыми функциями де шифрования D_A и D_B, которые обладают свойствами

$D_A E_A(M)=M, \ D_B E_B(M)=M$,

где M – передаваемое сообщение. Отметим, что равенство не зависит от порядка применения функций.

Абоненты A и B независимо друг от друга выбирают соответственно секретные ключи a, $0 < a < p$-1 и b, $0 < b < p$-1.

Абонент A посылает абоненту B сообщение

$g^a \ mod \ p$.

Абонент B

▪ вычисляет общий ключ

$= [g^a]^b \ mod \ p$;

▪ используя закрытую функцию D_B, создает подпись

(сообщение)D_B ($g^a \ mod \ p$, $g^b \ mod \ p$); ▪ используя ключ k,

шифрует подпись $E_K(D_B \ (g^a \ mod \ p, g^b \ mod \ p))$,

где E_K общая функция шифрования*;*

▪ отправляет абоненту A сообщение

$g^b \ mod \ p$, $E_K(D_B(g^a \ mod \ p, g^b \ mod \ p))$

Абонент A

▪ вычисляет общий ключ $= [g^a]^b \ mod \ p$;

▪ используя закрытую функцию D_A , создает подпись (сообщение)

$D_A \ (g^a \ mod \ p, g^b \ mod \ p)$; ▪ используя ключ k, шифрует подпись

$E_K(D_A \ (g^a \ mod \ p, g^b \ mod \ p))$*;*

▪ отправляет абоненту B сообщение

$EK(DA(g^a \bmod p, g^b \bmod p))$

▪ проверяет подпись B, вычисляя

$EB(DK(EK(DB(g^a \bmod p, g^b \bmod p)))) = = (g^a \bmod p, g^b \bmod p)$,где DK - общая функция дешифрования.

Абонент B проверяет подпись A, вычисляя

$EA(DK(EK(DA(g^a \bmod p, g^b \bmod p)))) = = (g^a \bmod p, g^b \bmod p)$.

Задания:

Реализовать два простейших протокола. Протокол Диффи-Хелмана формирования общего ключа и протокол подбрасывания монеты. Подготовить отчет. В отчете дать алгоритмы протоколов и описать их программные реализации. Подготовить для демонстрации учебные варианты контрольных примеров, которые моделируют работу с использованием криптографических протоколов.

Контрольные вопросы:

1. Дать определение протокола.
2. Перечислить задачи защиты информации, в которых используются криптографические протоколы.
3. Способы классификации криптографических протоколов.
4. Классификация криптографических протоколов по функциональному назначению.
5. Назначение протокола аутентификации сообщений.
6. Назначение протокола идентификации.
7. Назначение протокола обмена секретами.
8. Перечислить основные виды атак на протоколы.

Практическая работа №7 Абсолютно стойкие шифры

Цель работы:

Создать криптографическую систему шифрования данных, которая базируется на алгоритме шифровании ГОСТ, являющимся первым российским симметричным алгоритмом блочного шифрования данных.

Теоретические сведения:

Схема алгоритма режим простой замены

Текст, предназначенный для шифрования, разбивается на блоки длиной по 64 бит. Полученные блоки в дальнейшем будем называть начальными данными.

В ключевое устройство

$X0, X1, \ldots, X7$

записывается сгенерированный ключ длиной в 256 бит по 32 бита в каждый

Xi, $i = 1, 2, \ldots, 7$, то есть

$X0 = (w32, \ldots, w1)$,

$X1 = (w64, \ldots, w33)$,

…

$X7 = (w256, \ldots, w225)$.

Очередной блок начальных данных Ti длиной в 64 бит

$Ti(a1, a2, \ldots, a32, b1, b2, \ldots, b32)$

засылается в накопители $N1$ и $N2$ по следующей схеме

$N1 = (a32, \ldots, a1)$, $N2 = (b32, \ldots, b1)$.

Содержимое $N1$ суммируем по модулю 2^{32} с содержимым $X0$, результат засылается в $N1$.

Результат суммирования преобразовывается в блоке подстановки K.

Результат 5 шага преобразуется в регистре сдвига R. Происходит циклический сдвиг на 11 бит в сторону старших разрядов.

Результат сдвига суммируется по модулю 2 со значением в накопителе $N2$.Содержимое $N1$ пересылается в $N2$, а результат 7 шага записывается в $N1$.

Процесс шифрования с четвертого по восьмой шаг алгоритма повторяется еще 31 раз в следующем порядке использования частей ключа (с учетом проделанного преобразования с частью ключа $X0$):

1-й раунд шифрования из 8 шагов;

$X0, X1, \ldots, X7$ 2-й раунд шифрования из 8 шагов;$X0, X1, \ldots, X7$ 3-й раунд шифрования из 8 шагов;$X0, X1, \ldots, X7$

4-й раунд шифрования из 8 шагов

$X7, X6, \ldots, X0$

Результат последнего 32 шага шифрования, т.е. результат 8 шага алгоритма, сохраняется в накопителе $N2$, а в $N1$ сохраняется результат 31 шага.

Результат шифрования текущего блока Ti содержится в накопителях $N1$ с 1-го по 32 разряд и $N2$ с 1-го по 32 разряд.

Далее процесс шифрования в режиме простой замены продолжается с третьего шага до тех пор, пока не исчерпаются блоки с начальными данными.

Схема алгоритма: режим гаммирования

Текст, предназначенный для шифрования, разбивается на блоки длиной по 64 бит. Полученные блоки в дальнейшем будем называть начальными данными.

В ключевое устройство

$X0, X1, \ldots, X7$

записывается сгенерированный ключ длиной в 256 бит по 32 бита в каждый

X_i, $i = 1, 2, \ldots, 7$, то есть

$X0 = (w32, \ldots, w1)$,

$X1 = (w64, \ldots, w33)$,

…

$X7 = (w256, \ldots, w225)$.

Генератором последовательности случайных чисел формируется двоичная последовательность длиной в 64 бита (синхропосылка)

$S(s1, s2, \ldots, s32, s33, s34, \ldots, s64)$

и засылается в накопители $N1$ и $N2$ по следующей схеме

$N1 = (s32, \ldots, s1)$

$N2 = (s64, \ldots, s33)$.

Содержимое $N1$ и $N2$ шифруется в режиме простой замены.

Содержимое накопителей $N1$ и $N2$, в которых находится результат шифрования, переписываются соответственно в накопители $N3$ и $N4$.

Содержимое накопителя $N4$ суммируется по модулю $(2^{32}-1)$ с накопителем $N6$, вкотором хранится стандартная константа

$C2 = 00000001000000010000000100000001$.

Результат суммирования записывается в накопитель $N4$.

Содержимое накопителя $N3$ суммируется по модулю 2^{32} с

накопителем $N5$, в котором хранится стандартная

константа$C1 = 10000000100000001000000010000000$.

Результат суммирования записывается в накопителя

$N3$.Содержимое накопителя $N3$ переписывается в $N1$.

Содержимое накопителя $N4$ переписывается в $N2$.

Содержимое $N1$ и $N2$ шифруется в режиме простой

замены.Результат 10 шага образует текущий блок гаммы

Γ.

Текущий блок гаммы суммируется по модулю 2 с текущим блоком заданного текста. Результат суммирования формирует результат шифрования в режиме гаммирования.

Далее процесс шифрования в режиме гаммирования продолжается с четвертого шага до тех пор, пока не исчерпаются блоки с начальными данными.

Схема алгоритма: режим гаммирования с обратной связью

Текст, предназначенный для шифрования, разбивается на блоки длиной по 64 бит. Полученные блоки в дальнейшем будем называть начальными данными.

В ключевое устройство

$X0, X1, \ldots, X7$

записывается сгенерированный ключ длиной в 256 бит по 32 бита в каждый

X_i, $i = 1, 2, \ldots, 7$, то есть

$X0 = (w32, \ldots, w1)$,

$X1 = (w64, \ldots, w33)$,

…

$X7 = (w256, \ldots, w225)$.

Генератором последовательности случайных чисел формируется двоичная последовательность длиной в 64 бита (синхропосылка)

$S(s1, s2, \ldots, s32, s33, s34, \ldots, s64)$ и засылается в накопители $N1$ и $N2$ по следующей схеме

$N1 = (s32, \ldots, s1)$,

$N2 = (s64, \ldots, s33)$.

Содержимое $N1$ и $N2$ шифруется в режиме простой замены.

Результат четвертого шага образует текущий блок гаммы Γ.

Текущий блок гаммы суммируется по модулю 2 с текущим блоком заданного текста. Результат суммирования формирует зашифрованный блок

C по следующему правилу

$C(c1, c2, \ldots, c32, c33, c34, \ldots, c64)$

в режиме гаммирования с обратной связью.

7. Зашифрованный блок C является исходным состоянием накопителей $N1$ и $N2$ для выработки следующего блока гаммы. Последовательность битов

$c1, c2, \ldots, c32, c33, c34, \ldots, c64$

засылается в накопители $N1$ и $N2$ по следующей схеме

$N1 = (c32, \ldots, c1)$,

$N2 = (c64, \ldots, c33)$.

7. Далее процесс повторяется с четвертого шага, пока не исчерпаются блоки с начальными данными.

Схема алгоритма: режим выработки имитовставки

Текст, предназначенный для шифрования, разбивается на блоки длиной по 64 бит. Полученные блоки в дальнейшем будем называть начальными данными.

В ключевое устройство

$X0, X1, \ldots, X7$

записывается сгенерированный ключ длиной в 256 бит по 32 бита в каждый

Xi, $i = 1, 2, \ldots, 7$, то есть

$X0 = (w32, \ldots, w1)$,

$X1 = (w64, \ldots, w33)$,

…

$X7 = (w256, \ldots, w225)$.

Очередной блок начальных данных T_i длиной в 64 бит

$T_i(a1, a2, \ldots, a32, b1, b2, \ldots, b32)$

засылается в накопители $N1$ и $N2$ по следующей схеме

$N1 = (a32, \ldots, a1)$, $N2 = (b32, \ldots, b1)$.

Содержимое $N1$ и $N2$ подвергается преобразованию, которое соответствует первым шестнадцати циклам алгоритма в режиме простой замены.

Полученное после шестнадцати циклов содержимое накопителей в $N1$ и $N2$ суммируются по модулю 2 со вторым блоком открытых данных.

Результат суммирования заносится в накопители $N1$ и $N2$ и повторяется алгоритм с четвертого шага, только суммирование в шаге 5 происходит со следующим блоком открытого текста и т.д., пока не исчерпается исходное сообщение.

После завершение режима работы выбирается отрезок длиной в l бит. Значения параметра l определяется действующими криптографическими требованиями. При этом учитывается, что вероятность навязывания ложных данных равна 2^{-l}.

Задания:

Программная реализация криптографической системы, основанной на алгоритме шифрования ГОСТ, должна быть оформлена как некоторая программная оболочка. В программной реализации должен быть разработан интерфейс, удобный для эксплуатации программы. В интерфейсе следует предусмотреть:

− два режима формирования ключа – ключ задан и ключ формируется по умолчанию;

− ввода начальной информации из сформированного заранее файла, и файла, который создается в оболочке программы;

- режимы шифрования, которые предусмотрены в ГОСТ;
- режимы шифрования и дешифрования информации;

Подготовить отчет по работе. В отчете описать алгоритм ГОСТ, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций. В отчет включить описание алгоритма генерации ключа, детали программной реализации, которые представляют интерес с точки зрения разработчика.

Контрольные вопросы:

1. Перечислить принципы формирования шифров, которые сформулировал Шеннон.
2. Какое правило Кергоффса является актуальным для современной криптографии.
3. Дать определение шифру замены.
4. Дать определение шифру перестановки.
5. Каким принципам должны удовлетворять преобразования, которые обеспечивают шифрование информации.
6. Какие арифметические операции используются в алгоритме ГОСТ.
7. Что является ключом шифра простой замены.
8. Число ключей в шифре замены.
9. Привести примеры шифров простой замены.
10. Привести примеры шифров многоалфавитной замены.
11. Существуют ли абсолютно стойкие шифры?
12. Чему равна длина ключа в абсолютно стойком шифре?
13. Перечислить режимы работы ГОСТ.

Практическая работа №8 Гаммирование

Цель работы:

Используя алгоритм Эвклида создать программу, которая для чисел a и b определяет наибольший общий делитель.

Теоретические сведения:

Простые числа

Натуральное число p, больше единицы называется простым, если оно делится нацело только на единицу и на себя.

Теорема (Эвклид). Множество простых чисел бесконечно.

Обозначим через $\pi(x)$ функцию, которая равна числу простых чисел p в интервале $1 < x \leq p$. Российский математик П.Л. Чебышев в 1850г. показал, что имеет место

Простые числа являются важным понятием в криптографии. Многие современные криптографические системы строятся на базе простого числа. Поэтому алгоритмы генерации простых чисел и проверки на простоту сформированного числа в настоящее время являются важными инструментами при создании криптографической системы.

Заметим, что существует около 10^{151} простых чисел длиной от 1 до 512 бит включительно [5]. Для чисел, близких n, вероятность произвольно выбранному числу оказаться простым числом, равна $(1 / \ln n)$. При случайном выборе двух простых чисел в диапазоне от 1 до 151 бита вероятность совпадения этих чисел ничтожно мала.

Пусть даны два целых числа a и b. Говорят, что число a делит b, если существует такое целое число d, что $b=ad$. Число a в этом случае называют делителем b. Для факта, что a делит b, принято обозначение $a|b$. Справедливы следующие свойства делимости:

если $a|b$ и c – любое число, то $a|(bc)$;

если $a|b$ и $b|c$, то $a|c$;

если $a|b$ и $a|c$, то $a|(b \pm c)$.

Здесь уместно сделать замечание, что важную роль в арифметике целых чисел имеет теорема о делении.

Теорема о делении. Для любого целых чисел a и b, $b > 0$, существуют, и притом единственные, целые числа q и r, такие, что

$a = bq + r$, $0 \leq r < b$.

Определение. Натуральное число p, $p > 1$, называется составным, если число p имеет по крайней мере один положительный делитель, отличный от единицы и p.

Если число p составное, то справедлива следующая теорема.

Теорема. Для любого составного числа наименьший отличный от единицы положительный делитель является простым числом.

Одним из основных утверждений арифметики является факт, что любое натуральное число p можно единственным образом представить в виде произведения простых чисел. Например,

4290 = 3*2*5*11*13,

В общем случае каноническим разложением целого числа a называется представление в виде $a = p1\alpha1\ p2\alpha2\ldots\ pk\alpha k$,

где pi, $i = 1, 2, \ldots, k$, – простые числа. Например, $133100 = 2^2 5^2 11^3$.

Заметим, что, вообще говоря, представление большого числа в канонической записи, т.е. в виде произведения простых сомножителей, является трудной и очень важной задачей в криптографии. **Определение.** Общим делителем целых чисел $a1, a2, \ldots, ak$ называется любое целое число d, такое, что

$d|a1, d|a2, \ldots, d|ak$.

Определение. Наибольшим общим делителем (НОД) целых чисел $a1, a2, \ldots, ak$ называется такой положительный общий делитель этих чисел, который делится на любой другой делитель этих чисел.

Если d – наибольший общий делитель для чисел a и b, то для него вводится обозначение $(a, b) = d$.

Теорема. Если натуральное число p не делится ни на одно простое число $d \le p$,

то число p – простое.

Определение. Числа $a1, a2, \dots ak$ называются взаимно простыми, если наибольший общий делитель этих чисел равен 1.

Определение. Числа $a1, a2, \dots ak$ называются попарно взаимно простыми, если $(ai, aj) = 1, i \neq j, 1 \le i \le k, \ 1 \le j \le k$.

Если числа попарно взаимно простые, то все они взаимно простые.

Пример.

Числа 15, 21, 77 – взаимно простые, но эти числа не являются попарно взаимно простыми, потому что (15, 21) = 3.

Числа 34, 53, 99, 115 – попарно взаимно простые числа.

Алгоритм Евклида

Для двух целых чисел a и b существует сравнительно быстрый метод вычисления наибольший общий делитель. Упомянутый метод вычисления наибольший общий делитель называется алгоритмом Евклида. Приведем схему работы этого алгоритма.

Делим число a на число b, получаем

$= bq0 + r1;$

Делим число b на число $r1$, имеем

$= r1q1 + r2;$

Делим число $r1$ на число $r2$, запишем

$r1 = r2q2 + r3;$

Делим число $r2$ на число $r3$, получаем

$r2 = r3q3 + r4;$

$rt\text{-}1 = rtqt +$

$rt+1.$

Если остаток от деления $rt+1 = 0$, то в этом случае наибольший общий делитель равен числу rt и алгоритм вычисления наибольший общий делитель завершается.

Кратко алгоритм Эвклида можно сформулировать следующим образом.

Даны два целых числа a и b. Для определенности предположим, что $a > b$. Для поиска наибольшего общего делителя следует выполнить следующие операции.

Разделить a на b. Пусть остаток равен

$r, 0 < r \le a$.

Если $r = 0$, то алгоритм завершается, наибольший общий делитель равен b.

Положим $a = b$ и $b = r$.

Возвращаемся на шаг 1.

Пример. Найти наибольший общий делитель чисел $a = 1173$ и $b = 323$.

Делим число 1173 на число 323, получаем

1173 = 323*3 + 204;

Делим число 323 на число 204, получаем323 = 204*1 + 119;

Делим число 204 на число 119, получаем

204 = 119*1 + 85;

Делим число 119 на число 85, получаем119 = 85*1 + 34;

Делим число 85 на число 34, получаем

85 = 34*2 + 17;

Делим число 34 на число 17, получаем

34 = 17*23 + 0;

Итак, в итоге получаем

(1173, 323) = 17.

Задания:

В программной реализации алгоритма Евклида должен быть разработан интерфейс, удобный для эксплуатации. В интерфейсе следует предусмотреть:

- ввода начальной информации из сформированного заранее файла, и файла, который создается в оболочке программы;
- ввод начальной информации с клавиатуры.

Подготовить отчет по работе. В отчете описать алгоритм Евклида, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций.

Контрольные вопросы:

1. Дать определение простого числа.
2. Дать определение составного числа.
3. Сформулировать алгоритм Евклида.
4. Дать определение наибольшего общего делителя.
5. Сформулировать теорему о делении двух целых чисел.

ия блочных шифров

Цель работы:

Пусть число d – наибольший общий делитель для целых чисел a и b. Используя расширенный алгоритм Евклида, создать программу для определения таких двух чисел xи y, для которых выполняется равенство $d = ax + by$.

Теоретические сведения:

Теорема. Пусть d – наибольший общий делитель для a и b, т.е. d=(a,b), где $a > b$.

Тогда существуют такие целые числа x и y, что

$d = ax + by$.

Иными словам наибольший общий делитель двух чисел можно представить в виделинейной комбинации этих чисел с целыми коэффициентами.

Схема расширенного алгоритма Евклида

Определить

$a0 = 1, a1 = 0, b0 = 0,$

$b1 = 1, \alpha = a, \beta = b.$

Пусть число q – частное от деления числа a на число b, а число r – остаток от деленияэтих чисел, т.е.

$a = qb + r.$

Если остаток от деления r равен нулю, то выполняем шаг 6.

Определяем

$a = b, b = r, t = a0, a0 = a1,$

$a1 = t – a1q, t = b0, b0 = b1; b1 =$

$t – b1q$; Возвращаемся на шаг 2.

6. Определяем$x = x0, y = y0 \quad d$

$= \alpha x + \beta y.$

Пример. Дано a = 1769, b = 551. Используя расширенный алгоритм Эвклида, найтицелые числа x и y такие, что

$d = ax + by,$

где d – наибольший общий делитель

чисел a и b.**I этап последовательности**

вычислений Определить

$a0 = 1, a1 = 0, b0 = 0, b1 = 1, \alpha = 1769, \beta = 551.$

Частное от деления

$q = a/b = 1769/551 = 3,$

а остаток от деления r = 116.

Если остаток от деления r равен нулю, то выполняем шаг 6.

Определяем

$a = 551, b = 116, t = a0 = 1, a0 = a1 = 0, a1 = t – a1q = 1 – 0 \cdot 3 = 1$

$t = b0 = 0, b0 = b1 = 1, b1 = t – b1q = -3;$

В результате текущего шага получили следующие промежуточные значенияпараметров

$a= 551, b = 116, a0 = 0, a1 = 1, b0 = 1, b1 = -3.$

Так как остаток от деления $r \neq 0$, то возвращаемся на шаг 2.

II этап последовательности вычислений

Значение параметров

$a= 551, b = 116, a0 = 0,$

$a1 = 1, b0 = 1, b1 = -3.$

Частное от деления $q = a/b = 551/116 = 4,$

а остаток от деления $r = 87$.

–3, Если остаток от деления r равен нулю, то выполняем шаг 6.

Определяем

$a = 116,\ b = 87,\ \ t = a0 = 0,\ \ a0 = a1 = 1, a1 = t – a1q = 0 – 1 \cdot 4 = –4, t = b0 = 1,\ \ b0 = b1 =$

$b1 = t – b1q = 1 – (–3) \cdot 4 = 13.$

В результате текущего шага получили следующие промежуточные значения

параметров

a= 116, b = 87, $a0$ = 1, $a1$ = –4, $b0$ = –3, $b1$ = 13.

Так как остаток от деления $r \neq 0$, то возвращаемся на шаг 2.

III этап последовательности вычислений

Значение параметров

a= 116, b = 87, $a0$ = 1, $a1$ = –4, $b0$ = –3, $b1$ = 13.

Частное от деления $q = a/b = 116/87 = 1$, а остаток от деления $r = 29$.

Если остаток от деления r равен нулю, то выполняем шаг 6.

Определяем

a = 87, b = 29, $t = a0 = 1$, $a0 = a1 = –4$, $a1 = t – a1q = 1 – (–4)\cdot1 = 5$; $t = b0 = –3$, $b0 = b1 = 13$; $b1 = t – b1q = –3 – (13)\cdot 1 = –16$.

В результате текущего шага получили следующие промежуточные значения параметров

a= 87, b = 29, $a0$ = –4, $a1$ = 5, $b0$ = 13, $b1$ = –16.

Так как остаток от деления $r \neq 0$, то возвращаемся на шаг 2.

IV этап последовательности вычислений

Значение параметров

a= 87, b = 29, $a0$ = –4, $a1$ = 5, $b0$ = 13, $b1$ = –16.

Частное от деления $q = a/b = 87/29 = 3$, а остаток от деления $r = 0$.

Если остаток от деления r равен нулю, то выполняем шаг 6.

Определяем

a = 87, b = 29, $t = a0 = –4$, $a0 = a1 = 5$, $a1 = t – a1q = –4 – 5\cdot3 = –19$, $t = b0 = 13$, $b0 = b1 = –16$,

$b1 = t – b1q = 13 – (–16)\cdot3 = 61$.

В результате текущего шага получили следующие промежуточные значения параметров

a= 87, b = 29, $a0$ = 5, $a1$ = –19, $b0$ = –16, $b1$ = 61.

Так как остаток от деления r = 0, то выполняем шаг 6. 6. Вычисляем наибольший общий делитель по формуле

$d = \alpha x + \beta y$,

где

$x = x0 = 5$, $y = y0 = –16$, $\alpha = 1769$, $\beta = 551$.

Подставляя значение параметров, получаем

$d = \alpha x + \beta y = 1769*5 – 551*16 = = 8845 – 8816 = 29$.

Задания:

В программной реализации расширенного алгоритма Эвклида должен быть разработан интерфейс, удобный для эксплуатации в интерфейсе предусмотреть:

− ввода начальной информации из сформированного заранее файла, и файла, который создается в оболочке программы;

− ввод начальной информации с клавиатуры.

Подготовить отчет по работе. В отчете описать алгоритм Евклида, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций.

Контрольные вопросы:

1. Дать определение простого числа.
2. Дать определение составного числа.
3. Сформулировать алгоритм Евклида.
4. Дать определение наибольшего общего делителя.
5. Сформулировать теорему о делении двух целых чисел.

Практическая работа №10 Итеративные блочные шифры

Цель работы:

Изучить на практике основные принципы криптоанализа шифров гаммирования с короткой гаммой.

Теоретические сведения:

Здесь и далее под «короткой гаммой» подразумевается гамма, в несколько раз меньшей длины, чем защищаемый открытый текст. Основной принцип криптоанализа состоит в том, что наложение короткой гаммы не позволяет в полной мере маскировать статистические свойства открытого текста, а, следовательно, ими можно воспользоваться для криптоанализа. В случае использования двоичного вида открытого текста и шифртекста шифр модульного гаммирования с короткой гаммой получил название «простой XOR».

Как мы увидим далее, простой XOR - не более чем помеха для криптоаналитика. По сути, это ничто иное, как полиалфавитный шифр Вижинера. Простой XOR все еще употребляем, хотя и в значительно меньшей степени, чем еще несколько лет назад. По крайней мере, сейчас Microsoft честно признается в том, что это слабое шифрование, как показано на рисунке 9.1.

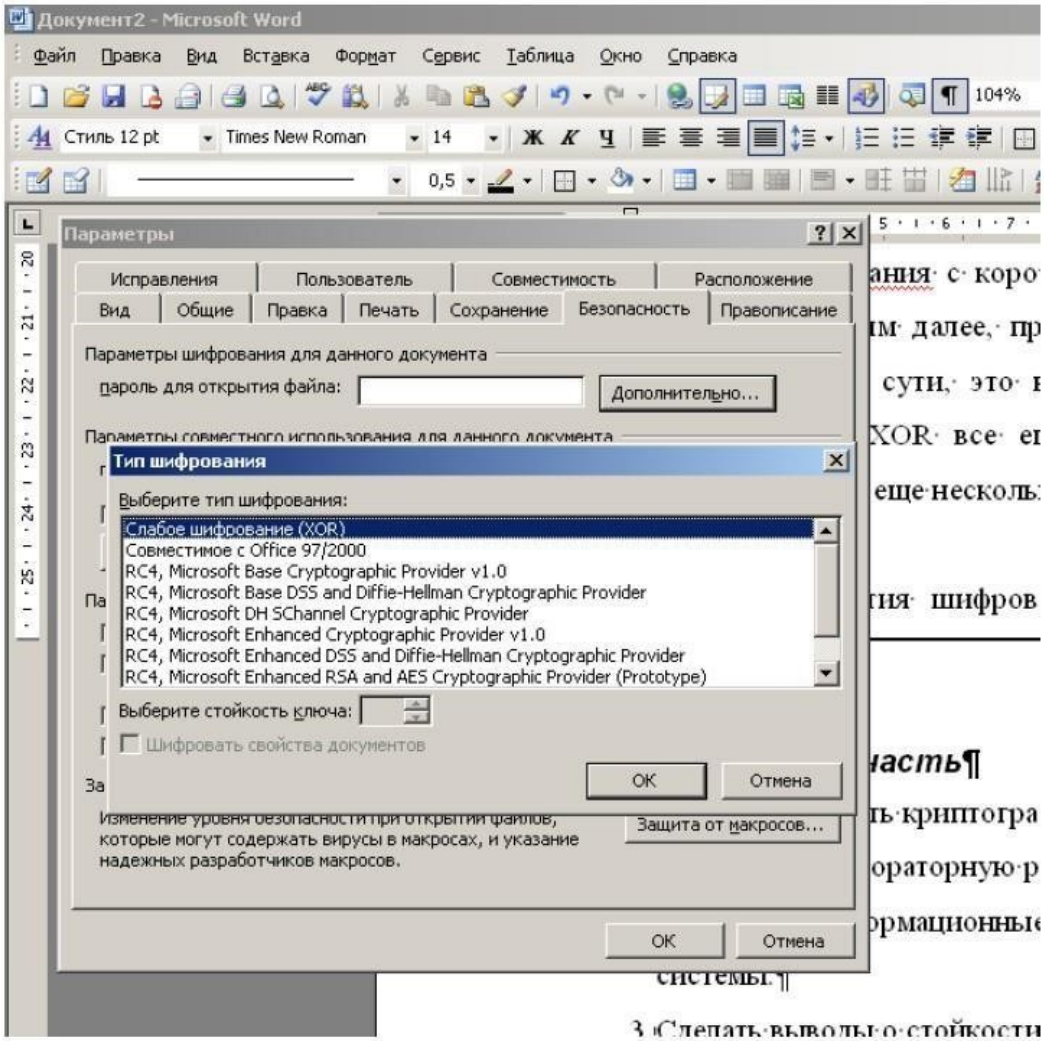

Рисунок 9.1 - Диалог «Тип шифрования» в Microsoft Word 2003

Приведем алгоритм вскрытия шифров модульного гаммирования с короткой гаммой.

Определим длину ключа с помощью процедуры, известной как подсчет совпадений. Применим операцию зашифрования к шифртексту, используя в качестве ключа сам шифртекст с различными смещениями, и подсчитаем совпадения. В случае простого XOR, например, должно совпасть свыше 6% байтов, в противном случае совпадет менее 0,4%. Минимальное смещение и есть длина ключа.

Сместим теперь шифртекст на длину ключа и выполним операцию зашифрования над смещенным и первоначальным шифртекстами таким образом, чтобы удалить ключ. Для простого XOR: пусть Y - шифртекст, У' - сдвинутый на длину ключа шифртекст.

Полученный на предыдущем этапе результат можно улучшить, используя гипотезы о характере открытого текста в совокупности с длиной ключа и последовательно восстановить его.

Задания:

Дешифровать заданный шифртекст. Использован шифр модульного гаммирования по mod30 (русский редуцированный алфавит). Подсказка: первые два слова: *старый бродяга.*

Л Ь Х Б Ф Т Ц Б З О Ф Ф Щ М Х Х Ю Т Ж С Ы П Ц ЦИ Ч Я Я К
Т Ч К В Т Б Ю З Н Ю Ц И Ф Ы Э Я Ц Х
А К Т Ж Ь Щ Ф Я Я Е Ц Ы И Я Щ В Я Э Ч Я Я И Е Ы
Л Ш Е Щ Ь П З Я Е Ы Г В М К Ч Ь Я Ц В М Е Т Э Э ЗТ Л В
М Т Л Я Ь К Х С

Контрольные вопросы:

1. Можно ли считать шифр модульного гаммирования с короткой гаммой стойким?
2. В чем отличия рассмотренного шифра от шифра одноразового блокнота?

Практическая работа №11 Симметричные криптоалгоритмы

Цель работы:

Изучить возможности использования криптографических хеш-функций при организации контроля целостности цифровых объектов.

Теоретические сведения:

Понятие Хеш-функции

Хеш-функция - это вычислительно эффективная функция, отображающая двоичную строку произвольной длины ***l*** в двоичную строку некоторой фиксированной длины ***m*** (обычно ***m<<l),*** называемую сверткой (хеш- значением, дайджестом) [1].

Особый интерес представляют ***однонаправленные хеш-функции,*** для которых практически невозможно восстановить входную последовательность по ее свертке. Большую важность имеют ***хеш-функции, свободные от коллизий,*** т.е. хеш-функции, для которых вычислительно сложно найти два сообщения с одинаковыми хеш-значениями. Значимым свойством хеш-функций является ***свойство полного перемешивания*** - изменение хотя бы одного бита входной последовательности приводит к изменению в среднем половины битов хешзначения. Хеш-функции, обладающие свойствами свободы от коллизий, однонаправленности и полного перемешивания, называются *криптографическими хеш-функциями.*

Криптографическая хеш-функция называется *ключевой,* если значение свертки зависит не только от хешируемого сообщения, но и от секретного ключа. Значение ключевой хеш-функции обычно называется кодом аутентификации (MAC - Message Autentification Code) или имитовставкой.

Криптографические хеш-функции на основе симметричных блочных алгоритмов

Криптографическую хеш-функцию можно построить, используя симметричный блочный алгоритм. Наиболее очевидный подход состоит в том, чтобы шифровать сообщение M посредством блочного алгоритма в режиме CBC или CFB с помощью фиксированного ключа и некоторого вектора инициализации V. Последний блок шифртекста можно рассматривать в качестве хеш-значения сообщения M.

При таком подходе не всегда возможно построить безопасную однонаправленную хеш-функцию, но всегда можно получить код

аутентификации сообщения MAC. [2]

Более безопасный вариант хеш-функции можно получить, используя блок сообщения в качестве ключа, предыдущее хеш-значение - в качестве входа, а текущее хеш-значение - в качестве выхода. Реальные хеш-функции проектируются еще более сложными. Длина блока обычно определяется длиной ключа, а длина хеш-значения совпадает с длиной блока. Поскольку большинство блочных алгоритмов являются 64-битовыми, некоторые схемы хеширования проектируют так, чтобы хеш-значение имело длину, равную двойной длине блока.

Если принять, что получаемая хеш-функция корректна, безопасность схемы хеширования базируется на безопасности лежащего в ее основе блочного алгоритма. Схема хеширования, у которой длина хеш-значения равна длине блока, показана на рисунке 11.1. Ее работа описывается выражениями:

$H0 = IH$

$Hi = EA(B) \oplus C$, где IH - некоторое случайное начальное значение; A, B и C могут принимать значения $M^i H^{i-1}$, $(M_i \oplus H_{i-1})$ или быть константами.

Рисунок 11.1 - Обобщенная схема формирования хеш-функции

Сообщение M разбивается на блоки M_i принятой длины, которые обрабатываются поочередно.

Три различные переменные A, B и C могут принимать одно из четырех возможных значений, поэтому в принципе можно получить 64 варианта общей схемы этого типа. Из них 52 варианта являются либо тривиально слабыми, либо небезопасными. Описание остальных 12 вариантов можно найти в[3].

Отечественный стандарт криптографической хеш-функции

Российский стандарт ГОСТ Р 34.11-94 определяет алгоритм и процедуру вычисления хеш-функции для любых последовательностей двоичных символов, применяемых в криптографических методах обработки и защиты информации. Этот стандарт базируется на блочном алгоритме шифрования ГОСТ 28147-89, хотя в принципе можно было бы использовать и другой блочный алгоритм шифрования с 64-битовым блоком и 256-битовым ключом.

Данная хеш-функция формирует 256-битовое хеш-значение.

Функция сжатия $H_i = f(M_i, H_{i-1})$ (оба операнда M_i и H_{i-1} являются 256битовыми величинами) определяется следующим образом:

Генерируются 4 ключа шифрования K_j, $j = 1...4$, путем линейного смешивания M_i, H_{i-1} и некоторых констант C_j.

Каждый ключ K_j, используют для шифрования 64-битовых подслов h_i слова H_{i-1} в режиме простой замены: $S_j = E_{K_j}(h_j)$. Результирующая последова-

тельность S4, S3, S2, S1 длиной 256 бит запоминается во временной переменной S.

Значение Hi является сложной, хотя и линейной функцией смешивания S, M_i и H_{i-1}.

При вычислении окончательного хеш-значения сообщения M учитываются значения трех связанных между собой переменных:

Hn - хеш-значение последнего блока сообщения;

Z - значение контрольной суммы, получаемой при сложении по модулю 2 всех блоков сообщения;

L - длина сообщения.

Эти три переменные и дополненный последний блок M' сообщения объединяются в окончательное хеш-значение следующим образом:

$H = f(Z \; 0 \; M', f(L, f(M', Hn)))$.

Данная хеш-функция определена стандартом ГОСТ Р 34.11-94 для использования совместно с российским стандартом электронной цифровой подписи.

Проблема контроля целостности

Целостность - это свойство информации, заключающееся в ее существовании в неискаженном виде (неизменном по отношению к некоторому фиксированному ее состоянию).

Каким способом можно обеспечить целостность информации? Можно, конечно же, поступить очень просто, записав эту информацию на некоторый физически неизменяемый носитель (например, CD-R). Но решит ли это проблему? Носитель может быть подменен, и, значит, мы просто подменим проблему обеспечения целостности информации проблемой обеспечения целостности носителя. Более того, в большинстве случаев информация, для которой должно быть соблюдена целостность, хранится на сетевых дисках, должна быть

общедоступной и открытой для изменения пользователем, имеющем соответствующие права.

Криптография предоставляет универсальное средство контроля целостности сообщений - ***хеш-функции.***

Хеш-функции могут быть использованы для обеспечения целостности данных следующим образом. В определенный момент времени вычисляется свертка h(M), соответствующая конкретным входным данным M. Будем называть h(M) эталонным хеш-значением. При помощи организационных и/или инженерно-технических мер обеспечивается целостность h(M). В дальнейшем для контроля целостности M вычисляется свертка проверяемой информации M' - h(M') и сравнивается с эталонным хеш-значением h(M). Если h(M) = h(M'), информация целостна, иначе целостность нарушена. Конкретным применением такой технологии является контроль целостности программного обеспечения.

Если необходимо, чтобы проверка целостности была доступна только определенному лицу (группе лиц), а также в качестве дополнительного уровня защиты от подмены эталонного хеш-значения, может быть использована ключевая криптографическая хеш-функция.

Следует отметить, что контроль целостности с помощью криптографических хеш-функций по определению не является абсолютно достоверным, однако может быть выполнен с требуемой точностью.

Задания:

1. Создать отдельную папку Lab11. В ней создать следующие папки:

Source - для исходных тестовых файлов;

Changedlbit - для файлов, с измененным одним битом; - Changed lbyte - для файлов, с измененным одним байтом; - Changed_2byte - для файлов, с измененными двумя байтами.

2. Скопировать три тестовых файла в папку Source, в качестве которых взять следующие:

документ Microsoft Word, размером больше 100Кб;

изображение в формате BMP, размером больше 50Кб (в системной папке Windows);

исполняемый файл EXE, размером более 100Кб.

Найти хеш всех файлов и полученные значения занести в таблицу 1 (Приложение А).

Скопировать тестовые файлы в остальные папки.

С помощью HEX-редактора (например, в Far'е) у файлов, находящихся в каталоге Changed_1bit, изменить один бит где-нибудь в начале каждого файла. Например, по адресу 0000000760 находится байт F8 (11111000). Меняем второй бит и получаем FA (11111010). Для контроля правильности модификации желательно использовать калькулятор (Calc). Адреса изменяемых байтов и их значения до и после модификации занести в таблицу 1 (Приложение А).

Найти хеш измененных файлов и полученные значения занести в таблицу 1.

В папке Changed_1byte у файлов изменить один байт, т.е. 8 бит должны поменять свои значения. Это можно сделать инверсией всех битов одного байта, инверсией четырех бит у двух байтов и т.д. Адреса изменяемых байтов и их значения до и после модификации занести в таблицу 1 (Приложение А).

Найти хеш измененных файлов и полученные значения занести в таблицу 1 (Приложение А).

В папке Changed_2byte у файлов изменить два байта, т.е. 16 бит.

Найти хеш измененных файлов и полученные значения занести в таблицу 1 (Приложение А).

Повторить действия пунктов 4 - 10, но теперь изменения внести по другим адресам (ближе к середине или концу файлов).

Сделать вывод о наличии или отсутствии коллизий и предоставить полученную таблицу для отчета.

Контрольные вопросы:

1. Понятие хеш-функции.
2. Понятие криптографической хеш-функции.
3. Понятие коллизии.
4. Понятие ключевой хеш-функции.
5. Свойства однонаправленности и полного перемешивания.
6. Возможности построения хеш-функций на основе симметричных блочных алгоритмов.
7. Использование хеш-функций при решении задачи контроля целостности.

Практическая работа №12 Структура алгоритма симметричного шифрования

Цель работы:

Изучить возможности использования пакета КРИПТОН® Подпись для организации документооборота с использованием электронной цифровой подписи (ЭЦП) документов. Получить представление о задачах, возлагаемых на администратора и пользователей, работающих с подписанными документами.

Теоретические сведения:

Пакет программ КРИПТОН® Подпись предназначен для использования электронной цифровой подписи (ЭЦП) электронных документов.

ЭЦП обеспечивает:

установление авторства документов;

проверку целостности документов.

Для штатной работы с программами пакета КРИПТОН® Подпись каждый пользователь, предполагающий использовать ЭЦП в электронном документообороте, снабжается парой ключей - секретным и открытым. Пары

112 ключей создаются на подготовительном этапе с помощью программы "Мастер ключей подписи" либо самостоятельно пользователем, либо специально выделенным администратором.

Секретный ключ пользователя является тем самым ключевым элементом, с помощью которого формируется ЭЦП данного пользователя, поэтому ключевой носитель (дискета, смарт-карта и т.д.), содержащий данный ключ, должен храниться пользователем особо тщательно, обеспечивая тем самым отсутствие подделки своей подписи. Секретный ключ запрашивается программами пакета перед выполнением каких-либо действий, таким образом, не имея секретного ключа ЭЦП, войти в программы пакета невозможно.

Открытые ключи подписи используются для проверки ЭЦП получаемых документов-файлов. Владелец (или администратор) должен обеспечить наличие своего открытого ключа у всех, с кем он собирается обмениваться подписанными документами.

При этом следует исключить возможность подмены открытых ключей как на этапе передачи, так и на этапе их использования.

Используемые пакетом КРИПТОН® Подпись ключевые схемы подробно описаны ниже, кроме того, данный документ содержит ряд рекомендаций по защите секретных ключей ЭЦП от несанкционированного копирования или подмены.

Все действия программ пакета КРИПТОН® Подпись протоколируются в специальном журнале, который может быть впоследствии просмотрен с помощью программы "Менеджер журнала операций".

В данном документе рассмотрены параметры запуска программ пакета КРИПТОН® Подпись, меню, используемые при работе, и сообщения программ, а также описана технология использования пакета КРИПТОН® Подпись.

Целостность электронных документов подтверждается подписью, помещаемой в конец подписываемых документов-файлов. При формировании подписи используется текст документа и секретный ключ.

В качестве электронного документа в программе используется любой файл. При необходимости, несколько владельцев могут подтвердить достоверность документа, т.е. один документ-файл может быть подписан несколько раз. При этом не изменяются ни имя файла, ни его расширение.

В подпись записывается следующая информация:

дата формирования подписи;

срок окончания действия открытого и секретного ключей;

информация о лице, сформировавшем подпись(Ф.И.О., должность, краткое наименование фирмы);

секретный ключ (имя файла секретного ключа);

собственно код ЭЦП.

ЭЦП может быть записана также в отдельный файл. Это файл, имеющий имя, соответствующее подписанному файлу, и расширение sg*. В данном файле хранится вся вышеуказанная информация, а также имя файла, который был подписан. При таком способе простановки ЭЦП исходный файл не изменяется, что может быть полезно, например, при подписывании файлов программ и динамических библиотек (файлы *.exe, * dll), а также файлов - документов Microsoft Office, поскольку ЭЦП, хранящаяся в отдельном файле, не изменяет структуру подписанного файла. Способ подписывания определяется при настройке программ пакета с помощью программы "КРИПТОН® Подпись - Конфигурация".

Ключи ЭЦП представляют собой обычные файлы на дискете или каком-либо другомключевом носителе. В именах этих файлов используются следующие расширения:

sk - для секретного ключа;

pk - для открытого ключа.

Генерация случайного кода для создания открытого ключа выполняется аппаратно устройством криптографической защиты данных (УКЗД) серии "Криптон" или программно драйвером-эмулятором УКЗД (Crypton Emulator).

Для формирования подписи файла необходимо выбрать этот файл и затем выполнить команду "Поставить подпись".

Команда "Проверить подпись" используется для проверки наличия и подлинности подписи у файла, а также для получения дополнительной информации об авторе документа. Данные команды также выполняются после выбора проверяемого файла.

При необходимости можно удалить последнюю подпись или группу последних подписей. Для этого в программе используется команда "Удалить подпись". Для её выполнения необходимо также выбрать документы-файлы, у которых удаляются подписи.

Задания:

1. Создать отдельную папку ***Lab12.*** В ней создать следующие папки:
2. ***SecretKey*** - для секретных ключей;
3. ***PublicKey*** - для открытых ключей;
4. ***Source*** - для тестовых файлов;
5. ***Sign*** - для файлов подписей
6. С помощью «Мастера ключей подписи» создать пару секретный/открытый ключ, указав их расположение в соответствующих папках в ***Lab6.***
7. В программе "КРИПТОН® Подпись - Конфигурация" указать соответствующие папки секретных и открытых ключей из предыдущего пункта.
8. Скопировать в папку Source тестовые файлы различных форматов
9. (*.txt, *.doc, *.jpg).
10. Подписать тестовые файлы с использованием сгенерированного 115 ключа.
11. Проверить подпись тестовых файлов.
12. Проверить, корректно ли открываются подписанные файлы.
13. Удалить подпись тестовых файлов.
14. В программе "КРИПТОН® Подпись - Конфигурация" указать со хранение подписей в отдельные файлы и каталог для хранения полученных подписей - Sign.
15. Подписать тестовые файлы с сохранением подписей в отдельные файлы.
16. Проверить подпись тестовых файлов.
17. Проверить, корректно ли открываются подписанные файлы.

18. Удалить подпись тестовых файлов.

19. Оформить отчет о проделанных шагах 2 - 13, в который включить краткое описание результатов каждого шага.

Контрольные вопросы:

1. Понятие электронной цифровой подписи.
2. Назначение электронной цифровой подписи.
3. Назначение секретных и открытых ключей.
4. Состав информации, добавляемой к файлу при подписывании.
5. Параметры, настраиваемые в пакете КРИПТОН® Подпись.

Практическая работа №13 Потоковый шифр

Цель работы:

Изучить мероприятия по созданию, распределению и хранению ключей при использовании пакета КРИПТОН® Подпись для организации электронного документооборота. Получить представление о задачах, возлагаемых на администратора и пользователей системы электронных документов.

Теоретические сведения:

Хранение ключей

При работе с пакетом КРИПТОН® Подпись каждый пользователь должен иметь, как минимум, один секретный ключ для формирования своей подписи и множество открытых ключей для проверки чужих подписей. Понятно, что секретный ключ должен быть недоступен для других. Иначе любой, имеющий его, может вместо Вас подписывать документы.

Открытые ключи не являются секретными, но существует опасность их подмены. Рассмотрим следующую ситуацию.

У других пользователей есть доступ к Вашему компьютеру, на котором Вы храните открытые ключи. Один из них читает данные (Ф.И.О., должность, ...) из интересующего его открытого ключа. Далее генерирует секретный и открытый ключи с этими данными, заменяет на вашем компьютере открытый ключ, архивирует любой документ и присылает Вам. В этом случае проверка подписи дает результат "подпись лица (Ф.И.О., должность ...) верна", что может, мягко говоря, ввести Вас в заблуждение.

Таким образом, необходима защита и открытых ключей. Такую защиту можно обеспечить несколькими способами.

Использование персональной дискеты с ключами

Секретный и открытый ключи могут быть записаны на персональную дискету, доступ к которой должен быть только у ее владельца. Однако, при большом количестве открытых ключей такой вариант нецелесообразен, поскольку замедляется проверка подписи.

Рекомендуется использовать вариант персональной дискеты (или другого носителя), состав которой описан в таблице 13.1.

Минимально на персональной дискете могут находиться только два собственных ключа. В этом случае, в качестве ключа-сертификата (при отсутствии последнего) могут использоваться эти ключи.

На этой же дискете рекомендуем хранить файлы для инициализации УКЗД "Криптон" или драйвера-эмулятора: gk.db3, uz.db3.

Рассмотрим последовательность действий по защите ключей. В общем случае необходимо выполнить следующую последовательность шагов:

Создание собственных ключей на персональной дискете. Секретный ключ необходимо закрыть паролем, который не позволит злоумышленнику воспользоваться им при похищении или копировании его. Также это даст необходимое время для регистрации нового открытого ключа в случае утери дискеты.

Создание отдельного раздела (каталога) для размещения открытых ключей (например, PKDIR).

Создание резервных копий открытых ключей, полученных путем прямого обмена с другими пользователями. Эти ключи могут понадобиться при решении спорных вопросов, поэтому необходимо обеспечить их сохранность. С этой целью осуществляют запись открытых ключей в раздел с открытыми ключами (PKDIR), удаляют у них подпись, формируют подпись этих открытых ключей собственным секретным ключом (для обеспечения целостности открытых ключей в процессе работы).

Создание резервной копии открытых ключей, сертифицированных на ключе-сертификате. Эти открытые ключи записываются в соответствующий раздел (PKDIR). Ключ-сертификат записывается на персональную дискету. Предпочтительнее этот вариант,

чем предыдущий.

Задания:

1. Создать отдельную папку Labl3. В ней создать папки для хранения ключей трех пользователей.
2. Изучить параметры, настраиваемые в программе КРИПТОН®
3. Подпись - Конфигурация по Руководству пользователя (раздел 4.2).
4. В программе КРИПТОН Подпись - Конфигурация установить в соответствующих параметрах созданные папки хранения ключей
5. Создать три пары ключей в соответствующих папках.
6. От имени пользователя 1 (Администратора) сертифицировать все три ключа на своем ключе.

Контрольные вопросы:

1. Возможные угрозы для секретных и открытых ключей.
2. Мероприятия по защите секретных и открытых ключей.
3. Состав персональной дискеты.
4. Схема прямого обмена открытыми ключами.
5. Схема обмена открытыми ключами через сертификационный центр.
6. Задачи, возлагаемые на сертификационный центр.
7. Генерация секретных ключей.
8. Сертификация открытых ключей.

Практическая работа №14 Синхронные потоковые шифры

Цель работы:

Изучить протокол Диффи-Хеллмана, его расширения для трех и более участников. Уяснить основные уязвимости протокола Диффи-Хеллмана.

Теоретические сведения:

Протокол Диффи-Хеллмана

Пусть абоненты сети $A^1,A^2,...,A^N$ используют некоторый симметричный криптоалгоритм $y = E(x, k)$ с секретным ключом. Как известно для того, чтобы для связи каждой пары абонентов использовать уникальную пару ключей, необходимо иметь ^{2} ключ. Притом для обеспечения высокой степени секретности для каждого очередного сеанса связи необходимо использование уникального сеансового ключа $k_{session}$. Таким образом, для эффективной работы такой сети нужен механизм распределения ключей, способный обеспечить конфиденциальность ключевой информации и оперативность распределения ключей. Протокол открытого распределения ключей Диффи- Хеллмана, предложенный на заре развития криптографии с открытым ключом, призван решить сформулированную задачу.

Протокол Диффи-Хеллмана может быть расширен на случай, если три или более абонентов сети используют одинаковый ключ. Для этого достаточно выполнить следующие действия:

А ^{i} берет в общем справочнике число yl . А ^{j} берет в общем справочнике число yl . А ^{i} берет в общем справочнике число yl .

А ^{i} вычисляет yli yl_{xl} (modp). А ^{j} вычисляет $ylj = y$ l (modp).

Абонент А ^{i} отсылает yli абоненту А ^{i}

А ^{j} вычисляет $kljj = ^{ylix}j$ (modp).

Абонент А ^{j} отсылает ylj абоненту А ^{i}

А $_i$ вычисляет kj = ^{ylix}j (modp). 10. А ^{i} берет в общем справочнике число ^yj .

А ^{i} вычисляет $^yji = ^yxjj$ (modp).

Абонент А ^{i} отсылает ^yji абоненту А ^{i} .x]

A] вычисляет $kjjl = yjj$ (modp).

Полученный $к = kjjl = kjjl = kljj$ общий секретный ключ абонентов А ^{i} , А ^{j},*Уязвимости протокола Диффи-Хеллмана*

Схема Диффи и Хеллмана подвержена атаке “человек посередине”, суть которой состоит в том, что активный противник перехватывает компоненты ключа и формирует с каждым из участников протокола свой собственный ключ. Протокол Диффи-Хеллмана не содержит встроенных средств контроля присутствия активного противника. В качестве способа защиты может быть предложено использование электронной цифровой

подписи с участием независимого арбитра, сертифицирующего ключи ЭЦП участников протокола 6.

Задания:

На основе параметров протокола, заданных в таблице вариантов (таблица 1 Приложения А) вычислить общий сеансовый ключ для пары участников.

Отчет по лабораторной работе представить виде расчетного файламатематического пакета по выбору студента (например, MathCAD)

Отчет по лабораторной работе представить в виде расчетного файламатематического пакета по выбору студента (например, MathCAD)

Контрольные вопросы:

1. Протокол открытого распределения ключей Диффи-Хеллмана.
2. Уязвимости протокола Диффи-Хеллмана.
3. Способы усиления протокола Диффи-Хеллмана.
4. Сущность атаки «человек посредине».

Практическая работа №15 Моделирование угроз безопасности информационных ресурсов

Цель работы:

Изучить возможности использования пакета КРИПТОН® Шифрование для защиты электронных документов от несанкционированного доступа (НСД).

Получить представление об особенностях работы и выполняемых функциях пакета.

Теоретические сведения:

Основные термины ***Зашифрование***

Процесс преобразования открытых данных в закрытые (зашифрованные). В данном программном продукте используется алгоритм шифрования ГОСТ 2814789. Это симметричный алгоритм, что означает, что для шифрования и расшифрования используется одинаковый ключ.

Расшифрование

Процесс преобразования закрытых данных в открытые(расшифрованные).

Ключ, ключевая система

Конкретное секретное значение криптографического преобразования, определяющее ход процесса преобразования данных. В данном пакете программ в качестве ключей шифрования могут использоваться: Ключ Пользователя; Сетевой ключ.

При работе с ключами всегда производится анализ имитоприставки. Узел

Замены (УЗ)

Долговременный элемент ключевой системы ГОСТ 28147-89. Обычно хранится в файле uz.db3 на ключевом носителе и является первым ключевым элементом, вводимым в устройство шифрования при инициализации. Все компьютеры, между которыми предполагается обмен зашифрованной информацией (например, локальная сеть), должны использовать один и тот жеУЗ,

т. к. несоответствие Узлов Замены приведет к невозможности расшифрования файлов с другой машины.

Главный Ключ (ГК)

Секретный ключ, используется для шифрования других ключей. Обычно хранится в файле gk.db3 на ключевом носителе. Может быть зашифрован на пароле. Создается администратором. При инициализации устройства шифрования загружается в него и находится там до выключения питания ЭВМ. Пароль

Последовательность символов, вводимых с клавиатуры. Пароль защищает ключи от несанкционированного использования в случае их хищения или потери. В пакете КРИПТОН® Шифрование максимальная длина пароля для ключей шифрования - 37 символов; минимальная длина - 4 символа. Длина пароля определяет стойкость системы. Поэтому рекомендуется использовать длинные пароли с неповторяющимися символами. Символы разных регистров (прописные и строчные буквы) различаются. Использовать символы с кодами меньше 32 и больше 127 (управляющие символы, псевдографику и русские буквы) не рекомендуется.

Ключ пользователя (ПК)

Секретный ключ, используется для шифрования файлов и других ключей. Хранится в файле с расширением ".KEY", обычно в зашифрованном виде.

Сетевой ключ

Секретный ключ, используется для шифрования файлов с целью передачи их между узлами криптографической сети. Все узлы сети нумеруются. Для каждого узла, с которым планируется обмен информацией, необходимо иметь свой сетевой ключ. Задача обеспечения сетевыми ключами возлагается на администратора сети. Сетевые ключи хранятся в Сетевом Наборе.

Имитовставка, имитоприставка

Применяется для обеспечения защиты системы шифрования от навязы- вания

ложных данных. Имитовставка представляет собой отрезок информации фиксированной длины, полученный из открытых данных иключа. Создается при зашифровании данных и добавляется к ним. При расшифровании данных также вычисляется имитовставка и сравнивается с хранимой.

В случае несовпадения можно выделить следующие причины: изменен Узел Замены;изменен ключ, на котором были зашифрованы данные; изменены сами зашифрованные данные;

если для зашифрования использовался пароль, то при расшифро вании он был неверно введен;

неисправно устройство шифрования.

Сетевая Таблица

Для обмена зашифрованной информацией между N узлами необходимо N*(N-1) ключей (каждый с каждым). Фактически, это таблица, где в заголовках строк и столбцов проставлены номера узлов, а в ячейках хранятся ключи. Эта матрица симметрична, т.е. ключ для передачи от узла А узлу Б (сетевой ключ АБ) в точности равен сетевому ключу Б-А. Сетевая Таблица присоздании зашифровывается на Ключе Сетевой Таблицы (КСТ).

Сетевой Набор

Из полной Сетевой Таблицы необходимо для каждого из узлов сформировать набор ключей для связи с другими узлами. Фактически, такой набор представляет собой одну из строк таблицы. Сетевой Набор хранится в файле NNNNN.SYS в каталоге сетевых ключей, где NNNNN - пятизначный десятичный номер данного узла. Он всегда зашифрован на Ключе Сетевого Набора (КСН), хранящемся в файле NNNNN.KEY в каталоге сетевых ключей. КСН получают вместе с Сетевым Набором от администратора криптографической сети. При получении КСН обычно незашифрован, поэтому рекомендуется перешифровать его.

Назначение КРИПТОН® Шифрование

Пакет КРИПТОН® Шифрование предназначен для защиты электронных документов (файлов) от несанкционированного доступа при хранении их на персональном компьютере или передаче по открытым каналам связи. Защита документов осуществляется путем их шифрования по ГОСТ 28147-89.

Иерархическая структура ключей

На рисунке 16.1 приведена иерархическая структура ключей, использующаяся в пакете КРИПТОН® Шифрование.

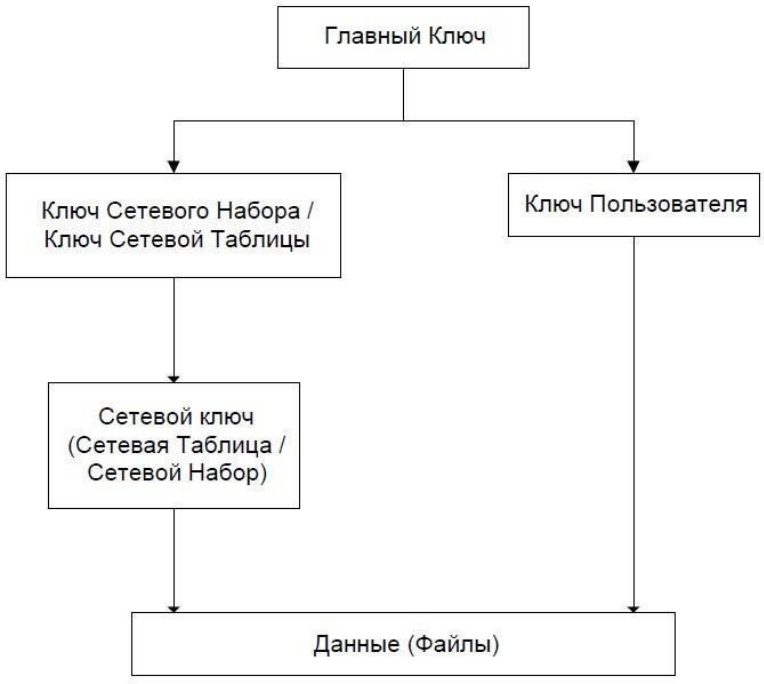

Рисунок 16.1 - Иерархическая структура ключей

Задания:

1. Изучить возможности шифрования пакета КРИПТОН® Шифрование
2. Создать отдельную папку Lab 16. В ней создать следующие папки:
3. Keys - для хранения пользовательских ключей
4. Source - для исходных тестовых файлов
5. Crypt - для зашифрованных файлов
6. Создать два ключа пользователя в каталоге Keys.
7. Скопировать тестовые файлы в папку Source. Занести размеры файлов в таблицу 1 (приложение А).
8. Сжать тестовые файлы при помощи WinRAR с максимальной степенью сжатия. Занести размеры полученных архивов в таблицу 1 приложения А.
9. Зашифровать тестовые файлы. Занести размеры зашифрованных файлов в таблицу 1 приложения А.
10. Сжать зашифрованные файлы при помощи WinRAR с максимальной степенью сжатия. Занести размеры полученных архивов в таблицу 1 приложения А.
11. 8.Сделать выводы о закономерностях изменения размера файла.
12. 9.Изучить возможности гарантированного удаления (уничтожения выбранных файлов без возможности их восстановления) пакета КРИПТОН® Шифрование.
13. 10.Уничтожить все архивы и зашифрованные файлы. В режиме командной строки (в пакетном режиме) выполнить задание своего варианта согласно таблице 2 (приложение А). Команды включить в отчет.

Контрольные вопросы:

1. Способы решения задачи защиты электронных документов от НСД методами криптографии.
2. Функции и состав пакета КРИПТОН® Шифрование.
3. Иерархия ключей в пакете КРИПТОН® Шифрование 4. Понятие сетевого набора и сетевой таблицы.
4. Особенности использования гарантированного удаления.

Практическая работа №16 Анализ существующих подходов Современные криптоалгоритмы с открытым ключом

Цель работы:

Изучить возможности использования пакета КРИПТОН® Шифрование для организации криптографической сети обмена электронными документами. Получить представление о задачах, возлагаемых на администратора и пользователей криптографической сети.

Теоретические сведения:

Основные термины

Криптографическая сеть

Множество компьютеров, называемых узлами, между которыми тем или иным способом осуществляется направленный обмен зашифрованной информацией, называется криптографической сетью. Направленный обмен подразумевает возможность зашифровать передаваемую информацию таким образом, чтобы расшифровать ее мог только тот узел, для которого она предназначалась. Обмен информацией между узлами криптографической сетиможет осуществляться как посредством локальных и глобальных сетей, так и с помощью сменных носителей. В пакете КРИПТОН® Шифрование для защиты данных в криптографической сети используется симметричный алгоритм шифрования - ГОСТ 28147-89.

Сетевой ключ

Секретный ключ, используется для зашифрования файлов с целью передачи их между узлами криптографической сети. Все узлы сети нумеруются. Для каждого узла, с которым планируется обмен информацией, необходимо иметь свой сетевой ключ. Задача обеспечения сетевыми ключами возлагается на администратора сети. Сетевые ключи хранятся в Сетевом Наборе.

Сетевая Таблица

Для обмена зашифрованной информацией между N узлами необходимо N*(N-1) ключей (каждый с каждым). Фактически, это таблица, где в заголовках строк и столбцов проставлены номера узлов, а в ячейках хранятся ключи. Эта матрица симметрична, т. е. ключ для передачи от узла А узлу Б (сетевой ключ АБ) в точности равен сетевому ключу Б-А. Сетевая Таблица присоздании зашифровывается на Ключе Сетевой Таблицы (КСТ).

Сетевой Набор

Из полной Сетевой Таблицы необходимо для каждого из узлов сформировать набор ключей для связи с другими узлами. Фактически, такой набор представляет собой одну из строк таблицы. Сетевой Набор хранится в файле NNNNN.SYS в каталоге сетевых ключей, где NNNNN - пятизначный десятичный номер данного узла. Он всегда зашифрован на Ключе Сетевого Набора (КСН), хранящемся в файле NNNNN.KEY в каталоге сетевых ключей.

КСН получают вместе с Сетевым Набором от администратора криптографической сети. При получении КСН обычно зашифрован на главномключе, поэтому рекомендуется перешифровать его на пароле.

Организация криптографической сети

Для организации криптографической сети администратор должен выполнить следующие действия

Создать Сетевую Таблицу. Для этого необходимо указать количество узлов криптографической сети и имя Сетевой Таблицы. В результате автоматически создается новый Ключ Сетевой Таблицы NET_SYS.KEY, на котором автоматически зашифровывается Сетевая Таблица KEY_NET.SYS

Создать Сетевые Наборы (СН) для узлов криптографической сети.

По умолчанию создаются СН для всех узлов и в каждом сетевом наборе все узлы являются доступными. При необходимости можно создать такие СН, чтобы узел мог шифровать данные только для ограниченного множества узловкриптографической сети. В

результате получается множество Ключей Сетевого Набора (NNNNN.KEY) и Сетевых Наборов (NNNNN.SYS) для узловкриптографической сети.

Каждому узлу присвоить свой номер NNNNN. Распределить Ключи Сетевого Набора и Сетевые Наборы по соответствующим узлам криптографической сети. Узлу с номером NNNNN необходимо передать два файла: NNNNN.KEY и NNNNN.SYS.

Шифрование файлов для передачи в криптографической сети

Файл данных, передаваемый узлом А узлу Б, зашифровывается на Файловом (сеансовом) Ключе. Файловый Ключ создается автоматически при зашифровании файла данных и передается вместе с ним. Так как файловый ключ не может передаваться в открытом виде, то он зашифровывается на Сетевом Ключе А-Б. Этот ключ узел А берет из своего Сетевого Набора.

Сетевой Набор узла А зашифрован на Ключе Сетевого Набора (КСН) узла А, который, в свою очередь, тоже может быть зашифрован на каких-либо ключах узла А (как правило, ГК).

Узел Б, по информации, присовокупленной к зашифрованному файлу, понимает, что файл пришел от узла А. Используя свои ключи, узел Б расшифровывает свой КСН.

Затем, используя КСН, узел Б расшифровывает свой набор и извлекает из него Сетевой Ключ А-Б. Так как этот Сетевой Ключ совпадает с темСетевым Ключом, который использовал узел А для зашифрования, то узел Б может расшифровать Файловый Ключ, пришедший вместе с файлом.

Наконец, с помощью Файлового Ключа расшифровывается сам файл.

Настройка параметров

После запуска Мастера ключей шифрования и выбора настройки параметров (рисунок 17.1) правая сторона панели содержит следующие настраиваемые параметры: *"Каталог Ключей Пользователя ”.*

Следует ввести каталог Ключей Пользователя. В этом каталоге будет производиться поиск Ключей Пользователя в следующих ситуациях: - Зашифровании файла на Ключе Пользователя.

Расшифровании файла, зашифрованного на Ключе Пользователя. Перешифровании файла, зашифрованного на Ключе Пользователя.

Требуемый каталог можно указать также с помощью кнопки обзора каталогов. По данной кнопке на экран выводится меню "Диски"/"Карточки", позволяющее указать каталог с помощью стандартного диалога Windows или диалога выбора устройств SCApi.

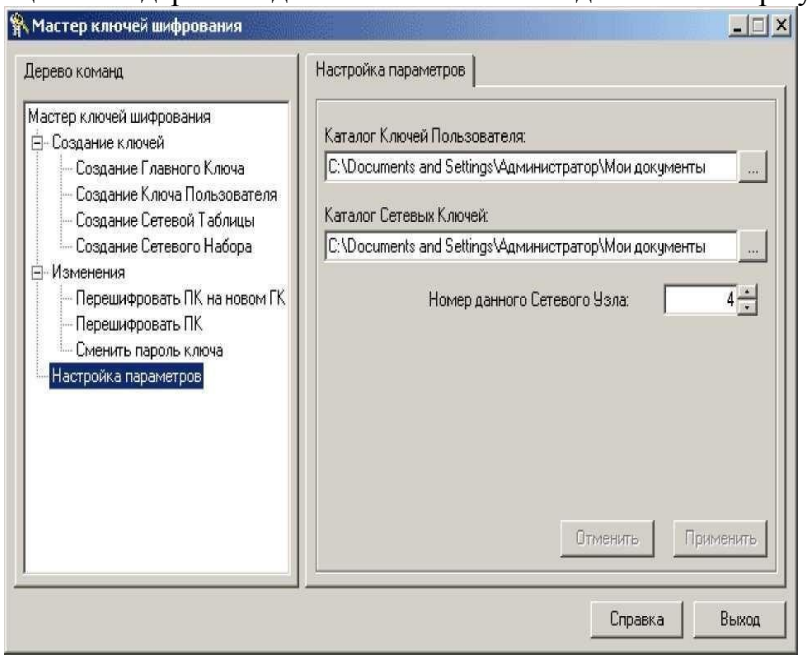

Рисунок 17.1 - Мастер ключей шифрования пакета КРИПТОН® Шифрование

"Каталог Сетевых Ключей".

Следует ввести каталог Сетевых Ключей. В этом каталоге будет производиться поиск Сетевых Ключей в следующих ситуациях: - Зашифровании файла на Сетевом Ключе.

Расшифровании файла, зашифрованного на Сетевом Ключе. Перешифровании файла, зашифрованного на Сетевом Ключе.

Требуемый каталог можно указать также с помощью кнопки обзораката логов, по которой вызывается стандартный диалог обзора папок Windows. Сетевые Ключи хранятся в Сетевых Наборах в фалах NNNNN.sys, где
NNNNN - номер данного сетевого узла - см. главу "Создать Сетевой Набор".

"Номер данного сетевого узла".

Следует ввести номер сетевого узла - целое число в диапазоне от 1 до 30000, идентифицирующее данный сетевой узел (пользователя или компьютер
- в зависимости от организационных решений).

Для ввода в действие сделанных изменений в конфигурации следуетнажать кнопку "Применить". Конфигурация будет изменена.

Для отмены внесенных изменений следует нажать кнопку "Отменить".

Следует учесть, что в Windows NT/2000/XP данные параметры являются персональными для каждого локального пользователя Windows. Параметры сохраняются в системном реестре Windows.

Задания:

Создать отдельную папку Lab17. В ней создать следующие папки:

Key_net - для Ключа СТ и самой СТ;

00001, 00002, 00003, 00004 - для Ключей СН исамихСН; - ***Source*** - для тестовых файлов и их зашифрованных вариантов.

Создать Сетевую Таблицу для четырех узлов в каталоге ***Key_net.***

Создать Сетевые Наборы для всех узлов. Разложить полученные Ключи СН и сами СН по соответствующим папкам ***00001,*** ... ***00004.***

Установить номер своего узла равный 2 и указать в параметрах соответствующий каталог сетевых ключей.

Зашифровать тестовые файлы для узла 3.

Установить номер своего узла равный 3 и указать в параметрах соответствующий каталог сетевых ключей.

Расшифровать тестовые файлы с автоматическим переименованием и затем полученные файлы уничтожить.

Установить номер своего узла равный 1 и указать в параметрах соответствующий каталог сетевых ключей.

Расшифровать тестовые файлы (зашифрованные для узла 3) с автоматическим переименованием.

Оформить отчет о проделанных шагах 2 - 9, в который включить краткое описание результатов каждого шага (например, создана сетевая таблица для четырех узлов и т. д.).

Из папки ***Материалы*** для данной лабораторной работы в папку ***Lab17*** скопировать Ключ СН и сам СН, соответствующий номеру варианта.

Установить номер своего узла, соответствующий номеру варианта и указать в параметрах каталог сетевых ключей ***Lab17***.

Зашифровать тестовые файлы для узла 11.

Полученные зашифрованные файлы предоставить для проверки.

Контрольные вопросы:

1. Понятие сетевого набора и сетевой таблицы.
2. Особенности защиты данных в криптографической сети с помощью симметричных и асимметричных алгоритмов шифрования
3. Назначение Сетевых Ключей и Ключей Сетевых Наборов.

4. Параметры, настраиваемые в пакете КРИПТОН®
5. Шифрование, для работы в криптографической сети.
6. Процесс обмена зашифрованными файламив криптографической сети.
7. Создание сетевых наборов для ограниченного числа узлов криптографической сети.

Практическая работа №17 Имитостойкость шифров

Цель работы:

Изучить мероприятия, необходимые для совершения обмена открытыми ключами при использовании пакета КРИПТОН® Подпись для организации электронного документооборота. Получить представление о мерах защиты от подмены открытых ключей, возлагаемых на пользователей системы.

Теоретические сведения:

Организация взаимосвязи пользователей. Прямой обмен открытыми ключами/

Все пользователи, обменивающиеся документами, предварительно должны обменяться своими открытыми ключами. При этом необходимо
исключить подмену на этапе пересылки.

Создается персональная дискета с собственными ключами. Секретный ключ закрывается паролем.

Для собственного открытого ключа формируется подпись на собственном секретном ключе, и открытый ключ записывается на дискету дляпередачи.

Подготавливается юридический документ на бумаге (например, письмо), в котором указываются: данные о владельце (Ф.И.О., должность, место работы), сам открытый ключ (распечатка в шестнадцатеричном виде), полномочия владельца (перечень документов, которые уполномочен удостоверять владелец открытого ключа). Данный документ должен быть оформлен таким образом, чтобы иметь юридическую силу в случае возникновения спорных вопросов о принадлежности подписи и полномочиях владельца. Если в письме не установлено полномочий, то они определяются по должности и месту работы. Например, бухгалтер одного предприятия не может удостоверять платежные поручения другого, а программист не может удостоверять платежные поручения и своего предприятия.

Дискеты и соответствующие сопроводительные документы направляются по адресам предприятий и пользователей, с которыми будет производиться обмен документами.

Получив такой комплект, необходимо убедиться в юридической силе полученного документа, а также в идентичности копий открытого ключа на дискете и в документе. Если открытый ключ верен, необходимо поместить его в каталог открытых ключей для последующего использования.

Такую операцию по пересылке ключей необходимо выполнить столько раз, сколько существует пользователей, с которыми Вы работаете. Это возможно, но для большого числа пользователей неэффективно.

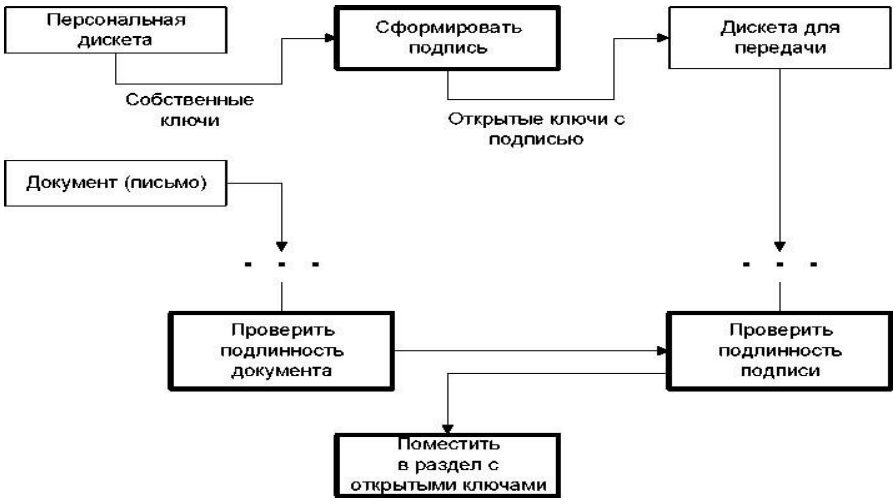

Рисунок 18.1 - Прямой обмен открытыми ключами

Задания:

Создать отдельную папку Lab18. В ней создать папку PUBLIC и персональные папки для каждого человека, выполняющего лабораторную работу на данном компьютере. В персональных папках создать следующие папки:

PKDIR - для хранения открытых ключей;

FLOPPY - каталог, имитирующий персональную дискету.

Изучить параметры, настраиваемые в программе КРИПТОН® Подпись - Конфигурация по Руководству пользователя (раздел 4.2).

Поочередно в программе КРИПТОН Подпись - Конфигурация установить в соответствующих параметрах созданные папки персональной дискеты и открытых ключей хранения ключей, для каждого - соответственно - свои.

Каждому в своей папке Floppy создать пару ключей, секретный ключ закрыть собственным паролем, отрытый ключ сертифицировать на своем же ключе, скопировав его в папку своих открытых ключей. При создании ключа обязательно указать свое полное ФИО и место работы (студент СГУ, факультет ..., группа ...)

Каждому в своей папке создать документ (MS Word) - сопроводительное письмо по форме, приведенной в приложении 1, для пересылки вместе со своим открытым ключом другим пользователям.

Скопировать свой открытый ключ вместе с сопроводительным письмом в папку PUBLIC. Эта папка будет играть роль курьера, доставляющего ключи и сопроводительные письма другим пользователям. Теперь необходимо, чтобы остальные пользователи получили ключи и сохранили у себя.

Каждый заходит в папку PUBLIC, изучает сопроводительные письма остальных пользователей, проверяет сертификацию открытых ключей и соответствие их распечатке в соответствующем сопроводительном письме. В случае отсутствия несоответствий, открытые ключи копируются в свои персональные папки в каталоги открытых ключей (PKDIR).

У всех полученных открытых ключей каждый удаляет подпись, и сертифицирует на своем ключе.

Теперь каждому пользователю необходимо получить открытые ключи, которые расположены в папке материалов к лабораторной работе в соответствии со своим вариантом. Для этого необходимо проделать те же операции, что указаны в п. 7, 8 по отношению к ключам и открытым ключам в папке материалов.

Для отчета необходимо будет показать все созданные файлы.

Контрольные вопросы:

1. Возможные угрозы для секретных и открытых ключей.
2. Мероприятия по защите секретных и открытых ключей.
3. Состав персональной дискеты.
4. Схема прямого обмена открытыми ключами.
5. Схема обмена открытыми ключами через сертификационный центр.
6. Задачи, возлагаемые на сертификационный центр.
7. Генерация секретных ключей.
8. Сертификация открытых ключей.

Практическая работа №18 Помехоустойчивость шифров

Цель работы:

Используя решето Эратосфена создать программу построения простых чисел, которые не превосходят некоторого заданного числа N.

Теоретические сведения:

Для составления таблицы всех простых чисел, которые не превосходят заданного числа *N*, надо из таблицы чисел 2, 3, …, *N* вычеркнуть все числа, которые делятся на два, кроме 2. Затем вычеркнуть все числа, которые делятся на три, кроме 3. Далее этот процесс (оставить наименьшее из оставшихся простых и вычеркивать все кратные этому простому) продолжаем до тех пор, пока очередное выбранное число не превысит N.

Данный метод позволяет строить множество простых чисел, но он неудобен для проверки простоты заданного числа.

Задания:

В программной реализации решета Эратосфена должен быть разработан интерфейс, удобный для эксплуатации в интерфейсе предусмотреть:

- ввода начальной информации из сформированного заранее файла и файла, который создается в оболочке программы;
- ввод начальной информации с клавиатуры.

Подготовить отчет по работе. В отчете описать алгоритм Эратосфена, описать структуру представления данных в программе, основные функции программы, назначение функций, входные и выходные параметры функций.

Контрольные вопросы:

1. Возможные угрозы для секретных и открытых ключей.
2. Мероприятия по защите секретных и открытых ключей.
3. Состав персональной дискеты.
4. Схема прямого обмена открытыми ключами.
5. Схема обмена открытыми ключами через сертификационный центр.
6. Задачи, возлагаемые на сертификационный центр.
7. Генерация секретных ключей.
8. Сертификация открытых ключей.

6 семестр

Практическая работа №1 Принципы построения криптоалгоритмов с открытым ключом. Современные криптоалгоритмы с открытым ключом.

Цель работы:

Изучить мероприятия для совершения обмена открытыми ключами через сертификационный центр при использовании пакета КРИПТОН® Подпись для организации электронного документооборота. Получить представление о задачах, возлагаемых на сертификационный центр и пользователей системы электронных документов.

Теоретические сведения:

Организация взаимосвязи пользователей.

Обмен открытыми ключами через сертификационный центр (СЦ)

Организуется центр для сертификации пользователей. В СЦ поступают открытые ключи и сопровождающие их документы (см. рисунок 19.1). В ответ пользователь получает:

зарегистрированные открытые ключи (или базу данных (БД) зарегистрированных открытых ключей) всех владельцев (в том числе и свой);

файл с полномочиями этих владельцев (и с подписями);

ключ-сертификат как в виде файла, так и в виде юридического документа.

Владелец при получении должен проверить истинность ключа- сертификата, а затем проверить подписи всех полученных открытых ключей и файлов. Также необходимо проверить свой открытый ключ. При положительных результатах проверки БД открытых ключей записываются в соответствующий каталог.

Время от времени СЦ должен пополнять Вашу базу данных открытых ключей и полномочий.

При такой организации работ пользователь формирует подпись документов и не заботится об обмене открытыми ключами и полномочиями. Однако большая нагрузка ложится на СЦ по рассылке баз открытых ключей и полномочий. Кроме этого, администратор этого центра в принципе может включить в базу данных ложный открытый ключ, что обязательно выявится. Если есть сомнения, можно запросить открытый ключ и полномочия напрямую.

Можно оставить за СЦ только сертификацию ключей и полномочий, освободив его от рассылки баз данных. В этом случае при первой посылке в любой адрес документов пользователю необходимо послать по этому адресу также зарегистрированные открытые ключи и полномочия. Можно зарегистрироваться в разных не связанных друг с другом СЦ или же связать СЦ в любую сеть для обмена либо только ключами-сертификатами, либо дополнительно еще и базами данных. Тогда пользователю достаточно зарегистрироваться только в одном из СЦ.

Пакет КРИПТОН® Подпись обеспечивает возможность организации работы по всем описанным выше вариантам.

Генерация секретных ключей

После установки пакета КРИПТОН® Подпись и создания дискеты с ключами для первоначального входа можно приступить к генерации ключей.Обычно этим занимается специально выделенный человек, называемый администратором.

Для генерации ключей необходимо:

запустить программу конфигурации пакета КРИПТОН® Подпись;

указать каталоги для секретного и открытых ключей (для секретного ключа можно сразу указать дискету или другой ключевой носитель);

запустить программу "Мастер ключей подписи";

вставить ключевой носитель и выполнить команду "Файл \ Ключи \ Сгенерировать";

повторить последнюю команду столько раз, сколько ключей необходимо создать;

создать новую базу данных открытых ключей;

выполнить команду "Добавить ключи из файла", добавить все ключи в базу данных.

Два последних пункта не являются обязательными, но обмен открытыми ключами удобнее вести, если они находятся в базе данных, а не в отдельных файлах.

Таким способом генерируются ключи для операторов, самого администратора и работника, который будет проводить сертификацию открытых ключей (это может быть и сам администратор). Последние в обязательном порядке закрываются паролем. Ключи могут передаваться по специальной связи, однако секретные ключи более правильно вообще не передавать по сети.

Оператор при получении секретного ключа может закрыть его на своем собственном пароле.

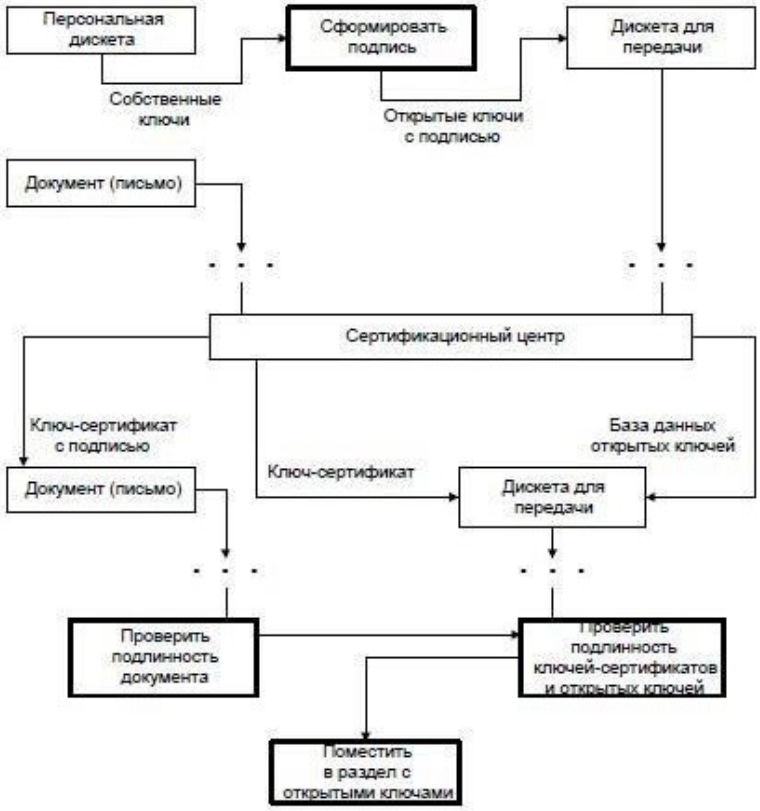

Рисунок 19.1 - Обмен открытыми ключами через сертификационный центр

Сертификация открытых ключей

Сертификация рекомендуется выполнять в сертификационном центре на защищенной ЭВМ специальным человеком в следующей последовательности:

при помощи программы "КРИПТОН® Подпись - Конфигурация" устанавливаются параметры: "Каталог расположения или имя файла секретного ключа" должен указывать на секретный ключ сертификационного центра, "Каталог расположения ключей-сертификатов" должен указывать на каталог, в котором расположены файлы ключей-сертификатов или базы данных ключей-сертификатов;

запускается на выполнение "Мастер ключей подписи";

формируются подписи у всех открытых ключей (команда "Файл \Ключи \ Сертифицировать");

создается новая база открытых ключей (или сертифицированные открытые ключи добавляются в старую);

выход из менеджера баз данных.

Сертифицированный секретный ключ имеет срок своего действия, по истечении которого должна проводиться плановая смена всех ключей.

Базы данных с сертифицированными открытыми ключами рассылаются всем

владельцам. Возможно заново сертифицировать открытые ключи, удалив у них старую подпись.

Задания:

1. Создать отдельную папку ***Lab19.*** Скопировать в нее персональные папки, сделанные в лабораторной работе № 18 и удалить из папок с открытыми ключами все ключи, кроме своего. Если папки не сохранились, то быстро создать их заново, сгенерировать каждому ключевую пару, и подготовить сопроводительное письмо.
2. Создать персональную папку для сертификационного центра, сгенерировать на персональную дискету СЦ ключевую пару, сертифицировать открытый ключ СЦ на самом себе.
3. Создать в ***Lab19*** папку ***CS PUBLIC*** и скопировать в нее все свои открытые ключи и сопроводительные письма и открыть мастер ключей подписи от имени сертификационного центра (указав нужные пути в параметрах конфигурации).
4. На данном этапе у нас имеется несколько пользователей и сертификационный центр. Каждый пользователь отослал свой открытый ключ с сопроводительным письмом с СЦ (скопировал в ***CS PUBLIC).*** Теперь СЦ создает новую базу данных ключей, проверяет соответствие присланных ключей сопроводительным письмам и, при отсутствии неточностей, заносит открытые ключи в базу, сертифицируя их на своем ключе.
5. СЦ добавляет в БД свой ключ.
6. СЦ создает сопроводительное письмо (в электронном виде), в котором указывает для всех (в том числе и себя) зарегистрированных в БД пользователей данные из их сопроводительных писем (ФИО, полномочия, сроки действия, распечатки ключа), и подписывает его.
7. Полученная БД вместе с сопроводительным письмом рассылается всем пользователям (выкладывается в ***CSPUBLIC).***
8. Каждый пользователь открывает полученную БД, проверяет истинность ключа-сертификата, а затем проверяет подписи всех полученных открытых ключей и файлов. Также необходимо проверить свой открытый ключ. При положительных результатах проверки БД открытых ключей записываются в соответствующий каталог.
9. На свою персональную дискету сохраняются извлеченные из БД сертификационный ключ (ключ СЦ) и свой открытый ключ, сертифицированный в СЦ.

Контрольные вопросы:

1. Возможные угрозы для секретных и открытых ключей.
2. Мероприятия по защите секретных и открытых ключей.
3. Состав персональной дискеты.
4. Схема обмена открытыми ключами через сертификационный центр.
5. Задачи, возлагаемые на сертификационный центр.
6. Генерация секретных ключей.
7. Сертификация открытых ключей.

Практическая работа №2 Криптосистема с открытым ключом

Цель работы:

Изучить математические алгоритмы модульных вычислений; разложения чисел на простые множители и проверки чисел на простоту.

Теоретические сведения:

Рассмотрим основные вопросы криптографии, связанные с использованием операций с целыми числами. Пусть $\boldsymbol{N}$ – множество натуральных чисел, $\boldsymbol{N^0}$ - множество натуральных чисел с 0, $\boldsymbol{Z}$ –множество целых чисел. Если $a,b \in \boldsymbol{Z}$, $b \neq 0$, то по теореме Евклида существует единственная пара целых чисел q, r таких, что

$a = b \cdot q + r$, $0 \le r < b$, (1)где

r – остаток.

Существует несколько обозначений для остатков: $r = Rb\ (a)$; $r = a(\mathrm{mod}\ b)$ и некоторыедругие.

Простейшие свойства остатков.

$R{-}b\ (a) = Rb\ (a)$, так как

$$a = q \cdot b + r \Leftrightarrow \begin{cases} a = (-q)(-b) + r \\ 0 \le r < |b| \end{cases}$$

$Rb(a + ib) = Rb(a), \quad \forall\, i \in \boldsymbol{Z}$,

так как $a + ib = (q + i)b + r$.

Если $r = 0$, то b делит a.

Арифметика остатков

Опр. 1. a сравнимо с b по модулю n $a \equiv b(n)$, если $Rn(a) = Rn(b)$.

Отношение $a \equiv b\ n(\ )$ есть отношение эквивалентности. Это отношение разбивает Z на непересекающиеся классы вычетов. Класс вычетов будем обозначать $[a]$, $a\ b, \in [\ ] a \Leftrightarrow a \equiv b\ n(\ )$.

$Z\ n/ = \{[\ ]\} a$ – коммутативное кольцо:

[a] + [b] = [a + b] коммутативная, ассоциативная операция.

[a]·[b] = [a· b] коммутативная, ассоциативная операция.

([a] + [b]) · [c] = [$a \cdot c$] + [$b \cdot c$].

Следующие свойства остатков важны для дальнейшего.

$(c * d)(\mathrm{mod}\ b) = (c(\mathrm{mod}\ b) * d(\mathrm{mod}\ b))(\mathrm{mod}\ b)$, где операция $* \in \{+, -, \times\}$.

$(a \times (c + d))(\mathrm{mod}\ b) = ((a \times c)(\mathrm{mod}\ b) + (a \times d)(\mathrm{mod}\ b))(\mathrm{mod}\ b)$.

Использование вычетов в криптографии основано на существовании алгоритма быстрого возведения в степень. Пусть

$a^{2\,2\cdot}(\mathrm{mod} n) = (a^2)^2(\mathrm{mod} n) = (a^2(\mathrm{mod} n))^2(\mathrm{mod} n)$,

где $a^2(\mathrm{mod} n)$ – остаток от деления a^2 на n, следовательно, его проще возводить в квадрат, чем число a^2. Таким образом, любая степень числа два позволяет быстрее возводить в степень, если последовательно приводить промежуточные результаты по модулю n.

$a^{2k}(\mathrm{mod} n) = ((a^2(\mathrm{mod} n))^2(\mathrm{mod} n)...)^2(\mathrm{mod} n)$

Если x – произвольное число, то существует представление $x = a0 \cdot 2^0 + a1 \cdot 2^1 + ... + ak \cdot 2^k$

и тогда возведение в степень опирается на быстрый алгоритм возведения в степень числа 2:

$a^x(\mathrm{mod} n) = (a^{a0 \cdot 20}(\mathrm{mod} n) \cdot a^{a1 \cdot 21}(\mathrm{mod} n) \cdot \cdot ... \ a^{ak \cdot 2k}(\mathrm{mod} n))(\mathrm{mod} n)$

Задания:

Для заданных значений a и N найти $a^{-1}\ \mathrm{mod}\ N$:

а) методом перебора полной системы вычетов без нуля;

б) используя функцию Эйлера;

в) используя расширенный алгоритм Евклида.

Для заданной пары чисел a и b найти наибольший общий делитель (a, b) и такие числа $u1$ и $u2$, что $a \cdot u1 + b \cdot u2 = (a, b)$.

Найти (хотя бы один) примитивный элемент мультипликативной группы GF(p):

а) используя известную факторизацию числа $p - 1$

б) используя полный перебор мультипликативной группы GF(p).

Контрольные вопросы:

1. Сформулируйте малую теорему Ферма
2. Сформулируйте китайскую теорему об остатках
3. Сформулируйте теорему Эйлера
4. Изложите суть расширенного алгоритма Евклида
5. Сформулируйте основные свойства остатков

Практическая работа №3 Современные криптоалгоритмы с открытым ключом

Цель работы:

Изучить математические алгоритмы модульных вычислений; разложения чисел на простые множители и проверки чисел на простоту.

Теоретические сведения:

Общие сведения о шифрах простой замены

Наиболее известными и часто используемыми шифрами являются *шифры замены*. Они характеризуются тем, что отдельные части сообщения (буквы, слова, ...) заменяются на какие-либо другие буквы, числа, символы и т.д. При этом замена осуществляется так, чтобы потом, по шифрованному сообщению, можно было однозначно восстановить передаваемое сообщение.

Пусть, например, зашифровывается сообщение на русском языке и при этом замене подлежит каждая буква сообщения. Формально в этом случае шифр замены можно описать следующим образом. Для каждой буквы α исходного алфавита строится некоторое множество символов M_α так, что множества M_α и M_βпопарно не пересекаются при $\alpha \neq \beta$, то есть любые два различные множества не содержат одинаковых элементов. Множество M_αназывается множеством *шифробозначений* для

буквы α. является *ключом* шифра замены. Зная ее, можно осуществить как зашифрование, так и расшифрование.

При зашифровании каждая буква α открытого сообщения, начиная с первой, заменяется любым символом из множества M_α. Если в сообщении содержится несколько букв α, то каждая из них заменяется на любой символ из M_α. За счет этого с помощью одного ключа можно получить различные варианты зашифрованного сообщения для одного и того же открытого сообщения. Например, если ключом является таблица

а	б	в	г	д	е	ж	з	и	к	л	м	н	о	п	р
21	37	14	22	01	24	62	73	46	23	12	08	27	53	35	04
40	26	63	47	31	83	88	30	02	91	72	32	77	68	60	44
10	03	71	82	15	70	11	55	90	69	38	61	54	09	84	45

с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
20	13	59	25	75	43	19	29	06	65	74	48	36	28	16
52	39	07	49	33	85	58	80	50	34	17	56	78	64	41
89	67	93	76	18	51	87	66	81	92	42	79	86	05	57

то сообщение «я знаком с шифрами замены» может быть зашифровано, например, любым из следующих трех способов:

я	з	н	а	к	о	м	с	ш	и	ф	р	а	м	и	з	а	м	е	н	ы
16	55	54	10	69	09	61	89	29	90	49	44	10	08	02	73	21	32	83	54	74
41	55	77	10	23	68	08	20	66	90	76	44	21	61	90	55	21	61	83	54	42
57	30	27	10	91	68	32	20	80	02	49	45	40	32	46	55	40	08	83	27	42

Так как множества $M_а$, $M_б$, $M_в$… $M_я$ попарно не пересекаются, то по каждому символу шифрованного сообщения можно однозначно определить, какому множеству он принадлежит, и, следовательно, какую букву открытого сообщения он заменяет. Поэтому расшифрование возможно и открытое сообщение определяется единственным образом.

Рассмотрим некоторые примеры шифров замены. Пусть каждое множество $M_а$ состоит из одной буквы. Например,

а	б	в	г	д	е	ж	з	и	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
г	л	ь	п	д	р	а	м	ц	в	э	ъ	х	о	б	н	с	ж	я	и	ю	к	щ	ф	е	у	ы	ч	ш	т	з

Такой шифр называется шифром простой однобуквенной замены. По ключу удобно проводить зашифрование и расшифрование: при зашифровании каждая буква открытого текста заменяется на соответствующую букву из второй строки («а» на «г» и т.д.) При

расшифровании, наоборот, «г» заменяется на «а» и т.д. При шифровании и расшифровании надо помнить вторую строчку, то есть ключ.

Запомнить произвольный порядок букв алфавита достаточно сложно. Поэтому всегда пытались придумать какое-либо правило, по которому можно просто восстановить вторую строчку.

Одним из первых шифров, известных из истории, был так называемый *шифр Цезаря*, для которого вторая строка является последовательностью, записанной в алфавитном порядке, но начинающейся не с буквы «а»:

а	**б**	**в**	**г**	**д**	…	**ь**	**э**	**ю**	**я**
г	д	е	ж	з	…	я	а	б	в

Другим примером шифра замены может служить лозунговый шифр. Здесь запоминание ключевой последовательности основано на лозунге – легко запоминаемом слове. Например, выберем слово-лозунг «учебник» и заполним вторую строку таблицы по следующему правилу: сначала выписываем слово-лозунг, а затем выписываем в алфавитном порядке буквы алфавита, не вошедшие в слово-лозунг. Вторая строка в примет вид

у	ч	е	б	н	и	к	а	в	г	д	ж	з	л	м	о	п	р	с	т	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

В данном случае число вариантов ключа существенно больше числа букв алфавита.

Рассмотренные шифры имеют одну слабость. Если в открытом сообщении часто встречается какая-либо буква, то в шифрованном сообщении часто будет встречаться соответствующий ей символ или буква. Поэтому при вскрытии шифра замены обычно стараются наиболее часто встречающимся символам шифрованного сообщения поставить в соответствие буквы открытого сообщения с наибольшей предполагаемой частотой появления. Если шифрованное сообщение достаточно большое, то этот путь приводит к успеху, даже если вы не знаете ключа.

Кроме частоты появления букв, могут быть использованы другие обстоятельства, помогающие раскрыть сообщение. Например, может быть известна разбивка на слова и расставлены знаки препинания. Рассматривая небольшое число возможных вариантов замены для предлогов и союзов, можно попытаться определить часть ключа

При анализе шифрованного сообщения следует исходить из того, что число различных вариантов для части определяемого ключа не такое уж большое, если вы находитесь на правильном пути. В противном случае либо вы получите противоречие, либо число вариантов ключа будет сильно возрастать. Обычно, начиная с некоторого момента определение открытого сообщения становится делом техники.

Шифрование даже относительно небольших текстов на одном ключе для рассмотренных шифров замены создает условия для вскрытия открытых сообщений. Поэтому такие шифры пытались усовершенствовать. Одно из направлений – построение шифров разнозначной замены, когда каждой букве ставится в соответствие один или два символа. Например,

а	**б**	**в**	**г**	**д**	**е**	**ж**	**з**	**и**	**к**	**л**	**м**	**н**	**о**	**п**	**р**	**с**
73	22	74	51	07	24	1	5	99	10	45	68	4	2	59	47	89

т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	т	у	ф
44	12	21	39	09	97	52	32	34	55	58	88	66	78	36	32	66

Если шифрованное сообщение написано без пробелов между символами, то появляется дополнительная трудность при разбиении шифрованного сообщения на отдельные символы и слова.

Другое направление создания шифров замены состоит в том, чтобы множества шифробозначений содержали более одного элемента. Такие шифры получили название шифров многозначной замены. Они позволяют скрыть истинную частоту букв открытого сообщения, что существенно затрудняет вскрытие этих шифров. Главная трудность, которая возникает при использовании таких шифров, заключается в запоминании ключа. Надо запомнить не одну строчку, а для каждой буквы алфавита α – множество ее шифробозначений $M\alpha$. Как правило, элементами множеств $M\alpha$ являются числа.

Задания:

Алгоритм вскрытия шифра простой замены достаточно сложно формализовать. Ниже приведены основные этапы криптоанализа, упускающие некоторые специальные нюансы.

Приближенный алгоритм криптоанализа шифров простой замены:

1. Определить, какой алфавит использован в криптограмме;
2. Подсчитать относительные частоты встречаемости букв (%);
3. Построить диаграмму встречаемости букв криптограммы;
4. Проанализировать внешний вид диаграммы встречаемости, сделать выводы о возможности шифра простой замены. Если сделан вывод о возможности шифра простой замены, перейти к шагу 5;
5. Выписать текст криптограммы с интервалом в две строки: промежуточная строка потребуется далее для записи букв открытого текста под соответствующими буквами шифртекста;
6. Построить матрицу биграмм (см. *Приложение 1*);
7. По таблице частот биграмм (*Приложение 4*) сделать вывод о соответствии самой частой (в криптограмме) биграммы;
8. По таблице частот букв (*Приложение 3*) сделать вывод о соответствии самой частой (в криптограмме) буквы;
9. Проверить гипотезы шагов 7 и 8.

Контрольные вопросы:

1. Построить математическую модель шифра простой замены
2. Существуют ли зависимости частот k-грамм алфавита от характера текста?
3. Что является ключом шифра простой замены?
4. Каково максимальное количество ключей шифра простой замены?

Практическая работа №22 Имитостойкость шифра

Цель работы:

Изучить задачу о целочисленном ранце, возможности ее применения для задач асимметричной криптографии (алгоритм Меркла-Хеллмана).

Теоретические сведения:

Задача о целочисленном ранце

Прежде всего сформулируем задачу о ранце. Она известна в двух формулировках:

Вариант 1. (*задача о целочисленном ранце*): для произвольных натуральных чисел $cj, j = 1 .. n$ и S требуется узнать, существует ли такой набор целых чисел $xj, j = 1 .. n$,

при котором выполнено:

$$\sum_{j=1}^{n} cj\ xj = S$$

Вариант 2. (*задача о* (0, 1) *ранце*): для произвольных натуральных чисел $cj, j = 1 .. n$ и S

требуется узнать, существует ли такой набор целых чисел $xj = \{0, 1\}, j = 1 .. n$, при котором выполнено:

$$\sum_{j=1}^{n} cj\ xj = S$$

Задача о целочисленном ранце является NP-полной. Данная задача получила свое название в связи с тем, что поставленная задача (случай (0, 1) ранца) может быть переформулирована также в следующем виде: имеется набор предметов с известными весами и рюкзак, который может выдержать вес, не превышающий заданной величины.

Можно ли выбрать набор предметов для погрузки в рюкзак так, чтобы они в точности имели максимально возможный вес.

Согласно теоретико-сложностному определению NP-полных задач, всякая NPполная задача может оказаться трудной лишь на некоторой бесконечной последовательности входных слов (но не для всех входных слов). Так и для задачи о ранце существует подкласс входных слов, для которого эта задача не является сложной. Это так называемые супервозрастающие последовательности.

> Для супервозрастающей последовательности существует полиномиальный алгоритм укладки. Возьмем в качестве текущего полный вес, который надо получить, и сравним его с весом самого тяжелого предмета в последовательности. Если текущий вес меньше веса данного предмета, то его в рюкзак не кладут. Если текущий вес больше или равен весу этого предмета, то положим его в рюкзак. Уменьшим текущий вес на вес положенного предмета и перейдем к следующему по весу предмету в последовательности.Будем повторять эти шаги до окончания процесса. Если текущий вес уменьшится до нуля, то решение найдено, в противном случае его не существует. Формально описанный алгоритм выглядит так: **Входные данные:** ($a\ a1, 2,\ldots,an$), S
> **Выход:** ($x\ x1, 2,\ldots,xn$)

Шаг 1. Положить $i := n$

Шаг 2. Если $i > 1$, то

Если $S > a_i$, то положить $x_i : 1=$ и $S := S - a_i$,

в противном случае положить $x_i : 0=$;

Шаг 3. Положить

$i := i - 1$ и возвратиться к **шагу 2**.

Пример 1. $S = 70$, $A = \{2, 3, 6, 13, 27, 52\}$

Текущий вес ранца полагаем $S : 70=$

Начальный элемент 52 – самый «тяжелый», кладем его в ранец. Уменьшим теперь текущий вес на вес уложенного элемента:

$$S := S - 52 \Rightarrow \qquad S = 18$$

Следующий самый «тяжелый» элемент в последовательности – 27. Однако его вес больше текущего веса ранца, поэтому его нельзя уложить. Элемент $13 < 18$, следовательно, он может быть уложен. Текущий вес:

$$S := S - 18 \Rightarrow \qquad S = 5.$$

И так далее: элемент 6 положить нельзя, однако элемент 3 можно. Текущий вес:

$$S := S - 3 \Rightarrow \qquad S = 2.$$

Вес последнего элемента в точности совпадает с текущим весом S.

Результат:

$$52 \cdot 1 + 27 \cdot 0 + 13 \cdot 1 + 6 \cdot 0 + 3 \cdot 1 + 2 \cdot 1 = 70$$

Не супервозрастающие или нормальные ранцы представляют собой сложную задачу, эффективный алгоритм решения которой не известен. Не исключен, конечно, способ полного перебора всех возможных комбинаций наполнения ранца, однако такой способ нельзя назвать эффективным. В общем случае для задачи о ранце неизвестно эффективных алгоритмов решения (лучшие временные оценки выполнения экспоненциально зависят от длины входной последовательности)

Алгоритм Меркла – Хеллмана

На базе задачи о (0, 1) целочисленном ранце была создана криптосистема МерклаХеллмана. Идея алгоритма Меркла-Хеллмана состоит в том, что элементы множества A выбираются с помощью блока открытого текста, длина которого (в битах) равна количеству элементов множества A. При этом биты открытого текста соответствуют значениям x_i , а шифртекстом станет полученный вес ранца S.

Задания:

1. Решить задачу об укладке ранца для своего варианта (*Приложение 1*).
2. Пользуясь алгоритмом, Меркла – Хеллмана, зашифруйте слово «university». В качестве параметров шифрования используйте последовательность A,числа m и W (*Приложение 2*). Произвестирасшифрование полученной криптограммы (*Приложение 3*).
3. Сообщение {97, 198, 101,283, 104, 237, 49, 140, 0, 234, 146, 49, 85, 140, 0} зашифровано с использованием алгоритма Меркла – Хеллмана с открытым ключом B = {55, 52, 49, 97, 85}. В качестве секретного ключа выбраны супервозрастающая последовательность A = {1, 7, 13, 28, 52}, числа m = 111 и W = 55. Расшифруйте сообщение.

Контрольные вопросы:

1. Сформулируйте задачу об укладке ранца.
2. Каковы основные принципы построения криптосистем на основе задаче об укладке ранца.
3. Сформулируйте определение супервозрастающей последовательности.
4. Приведите пример.
5. Объясните алгоритм Меркла – Хеллмана.
6. Приведите пример атаки не криптосистему Меркла – Хеллмана.

Практическая работа №5 Помехоустойчивость шифра

Цель работы:

Изучить основные детерминированные и вероятностные тесты на простоту. Освоить вычислительные методы тестирования чисел на простоту. На практике ознакомиться с конкретными реализациями тестов на простоту числа.

Теоретические сведения:

Общие сведения о простых числах

Натуральное число p, большее единицы, называется простым, если оно делится нацело только на 1 и на себя. Основная теорема арифметики гласит, что любое натуральное число n, большее единицы, может быть разложено в произведение простых чисел, причем это разложение единственно с точностью до порядка простых сомножителей. Каноническое разложение натурального числа n имеет вид

$n = p1\alpha1 \cdot p2\alpha2 \cdot \ldots pk\alpha k$,

где: $p1 < p2 < \ldots < pk$ различные простые числа, $\alpha1,\ldots,\alpha k \in N$.

Задача проверки простоты натурального числа и задача построения больших простых чисел имеют важные приложения в криптографии. Для установления простоты числа существуют два класса методов: детерминированные и вероятностные. Детерминированные методы определения позволяют точно определить, является ли число простым, однако их выполнение может занять значительное время. На практике часто используются вероятностные методы установления простоты – они позволяют дать ответ за весьма краткое время с заданной вероятностью.

Детерминированные методы проверки простоты чисел

Метод пробных делений

Пусть $n \in N$. Как проверить, является ли n простым? Если n – составное, то $n = ab$, где $1 < a \le b$, причем $a \le \sqrt{n}$. Поэтому для $d = 2, 3, \ldots, [\sqrt{n}]$ мы проверяем, делится ли n на d? Если делитель числа n не будет найден, то n – простое. В противном случае будет найден минимальный простой делитель числа n, т.е. мы даже разложим n на два множителя. Сложность метода составляет $O(\sqrt{n})$ арифметических операций с целыми числами. Возможны модификации этого метода. Например, мы можем проверить, делится ли n на 2 и на 3, и если нет, то перебираем далее только числа d вида $1 + 6j$ и $5 + 6j$, $j = 1, 2, \ldots$ Сложность метода отличается от сложности предыдущего лишь постоянной в $O(\cdot)$.

Решето Эратосфена

Если мы хотим составить таблицу всех простых чисел среди чисел 2, 3, ..., N, то нужно сперва вычеркнуть все числа, делящиеся на 2, кроме 2. Затем взять число 3 и вычеркнуть все последующие числа, делящиеся на 3. Затем взять следующее невычеркнутое число (т. е. 5) и вычеркнуть все последующие делящиеся на него числа, и так далее. В итоге останутся лишь простые числа. Для реализации метода нужен большой объем памяти ЭВМ, однако для составления таблиц простых чисел он является наилучшим.

Тест на основе малой теоремы Ферма

Для проверки простоты чисел n порядка $10^{30} – 10^{40}$ метод пробных делений уже неприменим. Следующий тест основан на малой теореме Ферма: если n простое, то для любого $a \in \boldsymbol{Z}$ выполнено сравнение

$a^n \equiv a \pmod{n}$.

Если же при этом $(a, n) = 1$, то

$a^{n-1} \equiv 1 \pmod{n}$.

Поэтому для проверки простоты n мы выбираем какое-либо $a \in \boldsymbol{Z}$ и проверяем выполнение малой теоремы Ферма за $O(\log n)$ арифметических операций (с помощью бинарного возведения в степень в кольце $(\boldsymbol{Z}/n\boldsymbol{Z})$. Если малая теорема Ферма не выполнена, то n – составное. Если же она выполнена, то мы еще не можем сделать вывод о простоте n, поскольку теорема дает лишь необходимое условие. Этот тест является эффективным для обнаружения составных чисел.

Однако существуют такие составные числа n, называемые числами Кармайкла, что для любого $a \in \boldsymbol{Z}$ выполнено сравнение

$a^n \equiv a \pmod{n}$

Таким образом, для проверки простоты чисел малой теоремы Ферма недостаточно. Введем понятие строго псевдопростого числа:

Опр. 1. Пусть $n > 1$ – нечетное натуральное число, $n - 1 = 2^s \cdot d$, где d – нечетно. Число n называется строго псевдопростым в базе a, $a \in \boldsymbol{N}$, если $(a, n) = 1$ и либо $a^d \equiv 1 \pmod{n}$, либо $a^{d \cdot 2^r} \equiv -1 \pmod{n}$ при некотором r, $0 \le r \le s$

На основании данного определения можно предложить следующий эмпирический тест:

шаг. Проверить строгую псевдопростоту n в базах 2, 3, 5, 7. Если n не строго псевдопростое в одной из баз, то оно составное.

шаг. Если $n = 3215031751$, то n – составное, иначе – простое.

Вероятностные тесты на простоту

Пусть $p \in \boldsymbol{N}$, p нечетно, $p > 1$. Вероятностный тест на простоту проводится следующим образом. Выбирается случайное $a \in \boldsymbol{N}$, $1 \le a < p$ и для него проверяется выполнение некоторых условий. Если какое-то из условий не выполнено, то число p – составное, поскольку для простых чисел эти условия являются необходимыми. Если же все условия выполнены, то из этого еще не следует простота p. Однако можно будет считать, что «p – простое число с некоторой вероятностью».

Кроме того, обычно доказывают оценку снизу для этой вероятности.

Чем больше значений а мы протестируем, тем ближе эта вероятность к единице.

Задания:

На практике, как правило, возникают два типа задач: задача генерации случайного числа и задача исследования заданного числа на простоту. Обе эти задачи имеют важное практическое значение и могу встречаться как отдельно, так и в суперпозиции. В ходе лабораторной работы предлагается решить следующую задачу:

В десятичной записи числа своего варианта найти первую подпоследовательность из подряд идущих 10 символов, являющуюся простым числом.

Возможная схема решения:

1. Любым доступным способом получить десятичную запись числа своего варианта (число знаков после запятой ≥ 10);
2. Полученную последовательность разбить на блоки R_i длиной 10 символов, начиная с i-го знака записи ($i = 1, 2, \ldots$)
3. Каждый блок R_i исследовать на простоту.
4. Получить ответ на сформулированную задачу в виде десятизначного простого числа.

Задачу решить двумя способами: с использованием детерминированного метода и с использованием вероятностного метода (конкретный алгоритм тестирования по выбору студента). Для вероятностного метода указать вероятность того, что это число является простым. Исследовать найденное число на принадлежность к числам специального вида.

Контрольные вопросы:

1. Понятие простоты числа.
2. Тесты на простоту: вероятностные и детерминированные.
3. Свойства простых чисел.
4. Теорема Ферма.
5. Сильные простые числа.
6. Числа Кармайкла.

Практическая работа №6 Помехоустойчивость зашифрованных данных

Цель работы:

Изучить применение регистров сдвига с линейной обратной связью (РСЛОС) в криптографии, теоретические основы их действия, способы построения.

Теоретические сведения:

Общие сведения о линейных регистрах сдвига

Широкое распространение в криптографических приложениях линейных регистров сдвига над конечными полями и кольцами обусловлено целым рядом факторов. Среди них можно отметить:

использование только простейших операций сложения и умножения, аппаратно реализованных практически на всех вычислительных средствах;

высокое быстродействие создаваемых на их основе криптографических алгоритмов; большое количество теоретических исследований свойств линейных рекуррентных последовательностей (ЛРП), свидетельствующих об их удовлетворительных криптографических свойствах.

Последовательности, имеющие максимально возможный период, получили название линейных рекуррентных последовательностей максимального периода или просто максимальных линейных рекуррентных последовательностей.

Значения периодов линейных рекуррентных последовательностей определяются свойствами их минимальных многочленов. В частности, для того чтобы линейная рекуррентная последовательность порядка m над полем из q элементов имела максимальный период, необходимо и достаточно, чтобы ее минимальный многочлен был примитивным многочленом. Так называется неприводимый многочлен, корень которого имеет в мультипликативной группе поля разложения Q порядок $q^m - 1$.

Несмотря на то, что имеется достаточно много критериев проверки неприводимости многочленов над конечными полями и методов их построения, доказать то, что заданный неприводимый многочлен примитивен, удается в исключительных случаях.

Для конечного поля из q элементов будем использовать стандартное обозначение GF(q).

Утверждение 1. Если $F(x)$ — неприводимый многочлен над полем GF(2) степени m, и $2^m - 1$ – простое число, то $F(x)$ – примитивный многочлен.

Утверждение 2. Неприводимый многочлен F(x) примитивен в том и только в том случае,

q^m-1

—

когда для любого простого числа p, делящего $q^m - 1$, многочлен x^p не сравним с 1 помодулю многочлена $F(x)$.

Построение примитивных многочленов представляет собой сложную задачу, решение которой даже в частных случаях сопряжено со значительными трудностями вычислительного характера. На практике используются таблицы неприводимых и примитивных многочленов над конечными полями.

Утверждение 3. Пусть $F(x)$ – примитивный многочлен степени n над полем P и θ – корень $F(x)$ в поле Q. Тогда любая ненулевая мультиграмма ($\alpha 1,\ldots, \alpha k$) встречается

на периоде ЛРП u ровно

$\nu\alpha\ \alpha(\ 1,..., k\ ) = q^{m\ k-}$

раз, а число вхождений нулевой мультиграммы на единицу меньше.

Линейные регистры сдвига с обратной связью

Регистр сдвига с обратной связью состоит из двух частей: регистра сдвига и функции обратной связи (рис. 1). Регистр сдвига представляет собой последовательность битов. (Длина регистра сдвига выражается числом битов. Если длина регистра равна n битам, то

регистр называют *n*-битовым регистром сдвига). При каждом извлечении бита все биты регистра сдвигаются вправо на 1 позицию. Новый старший бит рассчитывается как функция от всех остальных битов регистра. На выходе регистра сдвига оказывается 1 бит, обычно младший значащий бит.

Рис. 1. Регистр сдвига с линейной обратной связью

К простейшему типу регистра сдвига с обратной связью относится регистр сдвига с линейной обратной связью (РСЛОС) (Linear Feedback Shift Register, LFSR) (см. Рис. 2). Обратная связь представляет собой просто операцию XOR над некоторыми битами регистра; перечень этих битов называется последовательностью отводов (или точек съема). Иногда такую схему называют конфигурацией Фибоначчи. В криптографии РСЛОС используются чаше других регистров сдвига.

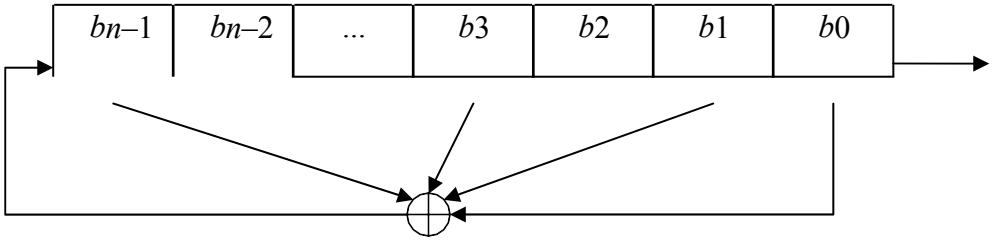

Рис. 2. Регистр сдвига с линейной обратной связью

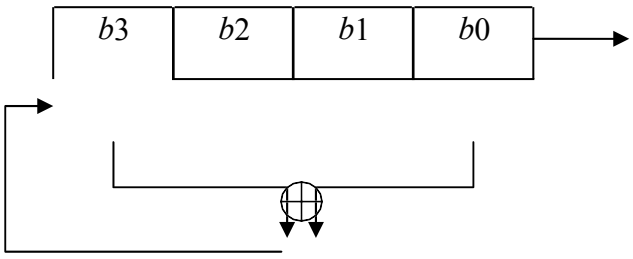

Рис. 3. 4-битовый РСЛОС

На рис. 3 показан 4-битовый РСЛОС с отводом от первого и четвертого битов. Если его инициализировать значением 1111, регистр будет принимать следующие внутренние состояния до тех пор, пока они не начнут повторяться (табл. 5.1).

Таблица 5.1 Внутренние состояния 4-битового РСЛОС

1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
0	1	1	0
0	0	1	1

1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
1	1	0	0
1	1	1	0

Выходной последовательностью регистра будет строка младших значащих битов: 111101011001000....

n-битовый РСЛОС может находиться в одном из $2^n - 1$ внутренних состояний. Это значит, что теоретически такой регистр может генерировать псевдослучайную последовательность с периодом $2^n - 1$ битов. (Это число равно $2^n - 1$, а не 2^n, поскольку заполнение РСЛОС нулями влечет вывод регистром бесконечной последовательности нулей, что совершенно бесполезно). Только при определенных последовательностях отводов РСЛОС циклически пройдет через все $2^n - 1$ внутренних состояний. Такие регистры называются регистрами РСЛОС с максимальным периодом.

Для обеспечения максимального периода конкретного РСЛОС, соответствующий многочлен, образованный из последовательности отводов регистра, должен быть примитивным по модулю 2. Степень многочлена является длиной регистра сдвига. Биты нумеруются от $n - 1$ до 0. Все степени, включая и нулевую, задают последовательность отводов, отсчитываемую от правого (младшего) края регистра сдвига. Член x^n обозначает вход, который подается на левый (старший) край регистра.

Например, запись (32, 7, 5, 3, 2, 1, 0) означает, что для данного 32-битового регистра сдвига новый бит генерируется с помощью операции XOR над седьмым, пятым, третьим, вторым, первым и нулевым битами. Период итогового РСЛОС будет максимальным, последовательно проходя последовательность $2^{32} - 1$ значений до ее повторения.

Задания:

Теоретический подраздел

Для многочлена своего варианта доказать его примитивность;

Определить максимальный период РСЛОС, построенного на многочлене своего варианта;

Рассчитать частоту появления триграммы своего варианта на периоде линейной рекуррентной последовательности над полем GF(2) с характеристическим многочленом своего варианта;

Определить длину отрезка, необходимую для восстановления минимального многочлена заданной линейной рекуррентной последовательности с помощью алгоритма Берлекемпа-Мэсси.

Практический подраздел

Запрограммировать РСЛОС своего варианта (*Приложение 1*). Используя написанную программу:

Получить выходную последовательность регистра;

Проверить, совпадает ли период полученной выходной последовательности с периодом, рассчитанным теоретически;

Определить частоту появления триграммы своего варианта и сравнить ее с теоретически рассчитанной.

Контрольные вопросы:

1. Понятие неприводимого и примитивного многочлена;
2. Понятие линейной рекуррентной последовательности;
3. Свойства линейных рекуррентных последовательностей;
4. Область применения РСЛОС;

5. Условия максимального периода РСЛОС.

Приложение 1

Вариант	Примитивный многочлен	Триграмма
1	(18,7,0)	101
2	(5,2,0)	000
3	(6,1,0)	111
4	(7,1,0)	100
5	(7,3,0)	010
6	(8,4,3,2,0)	110

7	(9,4,0)	001
8	(10,3,0)	011
9	(11,2,0)	110
10	(12,6,4,1,0)	001
11	(13,4,3,1,0)	010
12	(14,5,3,1,0)	011
13	(15,1,0)	101
14	(16,5,3,2,0)	000
15	(17,3,0)	111

Практическая работа №7 Криптографические хэш-функции
Статистические критерии

Цель работы:

Получить понятие об анализе криптографических псевдослучайных последовательностей (КПСЧ). Ознакомиться с подходами к анализу КПСЧ. Изучить методы статистического тестирования последовательностей.

Теоретические сведения:

Под ***псевдослучайной последовательностью (ПСП)*** понимается такой набор чисел, статистические характеристики которого соответствуют таковым для набора случайных чисел, но в отличие от последнего формируемый по определенному правилу, именуемому алгоритмом генерации ПСП.

К генератору ПСП предъявляются три главных требования:

большой период порождаемых последовательностей;

хорошие статистические характеристики; высокая производительность.

Подходы к анализу последовательностей

Многообразие критериев оценки псевдослучайных последовательностей, используемых при шифровании, весьма велико. Каждый из подходов к анализу шифрующих последовательностей можно отнести к одной из двух групп. Первая группа связана с поиском закономерностей, позволяющих воспроизвести шифрующую последовательность по относительно небольшому отрезку. При этом основные требования сводятся к тому, чтобы в псевдослучайной последовательности отсутствовали

относительно простые межзнаковые зависимости. Вторая группа критериев связана с оценкой статистических свойств последовательности: имеется ли в исследуемой последовательности какой-либо частотный дисбаланс, позволяющий аналитику предположить значение, следующего бита с вероятностью большей, чем при случайном выборе. Эти условия означают, в частности, что равномерно распределены частоты встречаемости в последовательности не только отдельных знаков, образующих последовательность, но и *s*- грамм, то есть наборов из *s* соседних знаков, $s = 1,2,\ldots$

Статистическое тестирование последовательностей

В настоящее время разработано большое количество тестов для анализа случайности последовательностей. Суть тестирования обычно сводится к проверке так называемой «нулевой гипотезы» относительно изучаемой последовательности, согласно которой последовательность длины N получена на основе N испытаний схемы Бернулли с вероятностью появления «1», равной ½. Статистический тест T для двоичных последовательностей длины N можно рассматривать как булеву функцию

$T V: N \to \{1,0\} = \{$"*принять*","*отвергнуть*"$\}$,

которая разделяет множество V_N двоичных последовательностей длины N на множество

$V_{N,0}$ «неслучайных» последовательностей (обычно небольшое) и множество $V_{N,1} = V_N \setminus V_{N,0}$ случайных последовательностей:

$V_{N\,j,} = \{s^N \in V_N : T\,s(^N) = j\}, j \in \{0,1\}$,

где $s^N = (s\ s1,\ 2,\ldots, sn\ )$. Вероятность ρ того, что случайно выбранная последовательность длины N отвергается тестом, выражается равенством

$\rho = V_{N,0} \cdot 2^{-N}$,

как правило, в реальных тестах ρ невелико, $\rho \leq 0.01$.

Для последовательностей большой длины N реализация теста с помощью точного определения значения булевой функции T требует трудоемких вычислений. Поэтому статистический тест обычно реализуют путем задания эффективно вычисляемой тестовой функции (статистики) fT, которая отображает VN в множество действительных чисел.

Обозначим через R^N последовательность из N независимых и одинаково распределенных случайных величин и рассмотрим вероятностное распределение случайной величины fT (R^N), принимающей действительные значения. При заданной величине ρ для значений функции fT задают верхний и нижний пороги $t1$ и $t2$:

$$P f\{ T(R^N) \le t1\} + P f\{ T(R^N) \ge t2\} = \rho.$$

Обычно пороги выбирают так, чтобы

$$P f\{ T(R^N) \le t1\} \approx P f\{ T(R^N) \ge t2\} \approx \frac{\rho}{2}.$$

Множество $VN,0$ «неслучайных» последовательностей мощности $\rho \cdot 2^N$ задается соотношением

$$VN,0 = \{s^N \in VN : f sT(N) \le t1 \text{ либо } f sT(N) \le t2\},$$

обычно функцию fT подбирают таким образом, чтобы вероятностное распределение случайной величины fT (R^N) было достаточно близко е хорошо известному эталонному распределению. Чаще всего таким эталонным распределением является нормальное распределение или распределение χ^2 с некоторым числом степеней свободы. Поскольку для таких вероятностных распределений составлены подробные числовые таблицы, то это упрощает определение порогов $t1$ и $t2$ при заданных величинах ρ и N.

Обычно к нормальному распределению приходят, когда суммируется большое количество независимых и идентично распределенных случайных величин. Распределение χ^2 с d степенями свободы используется, когда суммируются квадраты d независимых и нормально распределенных случайных величин с нулевым средним и единичной дисперсией.

В настоящей работе описано несколько наиболее популярных статистических тестов.

Частотный тест

Статистика $f sч(N)$ имеет вид:

$$(N)\,2\left(N\,N\right) f s_ч = \frac{}{\sqrt{N}} \cdot \left(\sum_{i=1} si - \_\_2\right)$$

количество единиц в случайной последовательности $R^N = (R\ R1,\ 2,\ldots,RN)$ распределено в соответствии с биномиальным распределением, которое при $N \ge 30$ хорошо аппроксимируется нормальным распределением с нулевым средним и единичной

дисперсией. Приемлемыми порогами являются значения $t1 = -t2 = 2.5 \div$

3.0. Автокорреляционный тест

Для последовательности s^N может быть применен автокорреляционный тест. Для этого последовательность s^N нужно преобразовать к виду

$$s\tau N = (s1 \oplus s1{+}\tau, s2 \oplus s2{+}\tau, \ldots, sN - \tau \oplus sN),$$

используя некоторое заданное τ. Данный тест используется для выявления корреляции между битами последовательности s^N на расстоянии τ.

Задания:

В рамках практической части задания необходимо провести статистический анализ последовательности s^N, сгенерированной в ЛР № 5 (РСЛОС). Для s^N необходимо выполнить как частотный, так и автокорреляционный тесты. Тесты могут быть реализованы в любом математическом пакете, либо в MS Excel, по выбору студента.

Контрольные вопросы:

1. Понятие псевдослучайной последовательности;
2. Тестирование ПСП;

3. Свойства псевдослучайных последовательностей;
4. Статистическое тестирование последовательностей;
5. Частотный тест;
6. Автокорреляционный тест.

Практическая работа №8 Применение криптографических атак на хеш-функции.

Цель работы:

Изучить основные методы и механизмы аутентификации.

Теоретические сведения:

Аутентификация - это процесс, в ходе которого на основании пароля, ключа или какой-либо иной информации, пользователь подтверждает, что является именно тем, за кого себя выдает.

Идентификация - это процесс, в ходе которого выясняются права доступа, привилегии, свойства и характеристики пользователя на основании его имени, логина или какой-либо другой информации о нем.

При входе пользователя в систему первым делом происходит его аутентификация. Если введенные логин и пароль совпадают с хранимыми в системе на сервере, то пользователь успешно входит в систему, иначе ему отказывается в доступе. Здесь уместно контролировать количество попыток, дабы избежать подбора паролей. В зависимости от сложности и надежности системы необходимо выбрать механизм работы с паролями. В самом простом случае можно разрешить пользователю самостоятельно вводить пароль. Но достаточно большое количество пользователей в качестве пароля вводит свой: логин, имя, номер телефона и т.п. Это можно разрешить только в тех системах, где проблема защиты информации пользователя является его собственным делом и не нанесет вреда системе в целом.

Идентификация и аутентификация (ИдиА) - это процесс распознавания и проверки подлинности заявлений о себе пользователей и процессов. ИдиА обычно используется при принятии решения, можно ли разрешить доступ к системным ресурсам пользователю или процессу.

Аутентификация через Интернет имеет ряд проблем. Достаточно легко можно перехватить данные идентификации и аутентификации (или вообще любые данные) и повторить их, чтобы выдать себя за пользователя. При

аутентификации вообще пользователи часто выражают недовольство ею и часто совершают ошибки, что делает возможным получение данных ИдиА с помощью социальной инженерии. Наличие дополнительной ИдиА при использовании Интернета делает необходимым распространение среди пользователей данных для ИдиА, что будет лишь усложнять им работу. Другой проблемой является возможность вклиниться в сеанс пользователя после выполнения им аутентификации.

Существует три основных вида аутентификации - статическая, устойчивая и постоянная. Статическая аутентификация использует пароли и другие технологии, которые могут быть скомпрометированы с помощью повтора этой информации атакующим. Часто эти пароли называются повторно используемыми паролями. Устойчивая аутентификация использует криптографию или другие способы для создания одноразовых паролей, которые используются при проведении сеансов работы. Этот способ может быть скомпрометирован с помощью вставки сообщений атакующим в соединение.

Постоянная аутентификация предохраняет от вставки сообщений атакующим.

Статическая аутентификация

Статическая аутентификация обеспечивает защиту только от атак, в ходе которых атакующий не может видеть, вставить или изменить информацию, передаваемую между аутентифицируемым и аутентифицирующим в ходе аутентификации и последующего сеанса. В этом случае атакующий может только попытаться определить данные для аутентификации пользователя с помощью инициации процесса аутентификации (что может

сделать законный пользователь) и совершения ряда попыток угадать эти данные. Традиционные схемы с использованием паролей обеспечивают такой вид защиты, но сила аутентификации в основном зависит от сложности угадывания паролей и того, насколько хорошо они защищены.

Устойчивая аутентификация

Этот класс аутентификации использует динамические данные аутентификации, меняющиеся с каждым сеансом аутентификации. Атакующий, который может перехватить информацию, передаваемую между аутентифицируемым и аутентифицирующим , может попытаться инициировать новый сеанс аутентификации с аутентифицирующим, и повторить записанные им данные аутентификации в надежде замаскироваться под легального пользователя. Усиленная аутентификация 1 уровня защищает от таких атак, так как данные аутентификации, записанные в ходе предыдущего сеанса аутентификации, не смогут быть использованы для аутентификации в последующих сеансах.

Тем не менее устойчивая аутентификация не защищает от активных атак, в ходе которых атакующий может изменить данные или команды, передаваемые пользователем серверу после аутентификации. Так как сервер связывает на время сеанса данного аутентифицировавшегося пользователя с данным логическим соединением, он полагает, что именно он является источником всех принятых им команд по этому соединению.

Традиционные пароли не смогут обеспечить устойчивую аутентификацию, так как пароль пользователя можно перехватить и использовать в дальнейшем. А одноразовые пароли и электронные подписи могут обеспечить такой уровень защиты.

Постоянная аутентификация

Этот тип аутентификации обеспечивает защиту от атакующих, которые могут перехватить, изменить и вставить информацию в поток данных, передаваемых между аутентифицирующим и аутентифицируемым даже после аутентификации. Такие атаки обычно называются активными атаками, так как подразумевается, что атакующий может активно воздействовать на соединение между пользователем и сервером.

Основные методы и механизмы аутентификации

Парольная аутентификация

В настоящее время парольная аутентификация является наиболее распространенной, прежде всего, благодаря своему единственному достоинству – простоте использования.

Однако, парольная аутентификация имеет множество недостатков:

1. В отличие от случайно формируемых криптографических ключей (которые, например, может содержать уникальный предмет, используемый для аутентификации), пароли пользователя бывает возможно подобрать из-за достаточно небрежного отношения большинства пользователей к

формированию пароля. Часто встречаются случаи выбора пользователями легко предугадываемых паролей, например:

пароль эквивалентен идентификатору (имени) пользователя (или имени пользователя, записанному в обратном порядке, или легко формируется из имени пользователя и т.д.);

паролем является слово или фраза какого-либо языка; такие пароли могут быть подобраны за ограниченное время путем «словарной атаки» - перебора всех слов согласно словарю, содержащему все слова и общеупотребительные фразы используемого языка;

достаточно часто пользователи применяют короткие пароли, которые взламываются методом «грубой силы», т.е. простым перебором всех возможных вариантов.

Существуют и свободно доступны различные утилиты подбора паролей, в том числе, специализированные для конкретных широко распространенных программных средств.

Пароль может быть получен путем применения насилия к его владельцу.

Пароль может быть подсмотрен или перехвачен при вводе.

Аутентификация с помощью уникального предмета

В большинстве случаев аутентификация с помощью уникального предмета обеспечивает более серьезную защиту, чем парольная аутентификация.

Предметы, используемые для аутентификации, можно условно разделить на следующие две группы:

«Пассивные» предметы, которые содержат аутентификационную информацию (например, некий случайно генерируемый пароль) и передают ее в модуль аутентификации по требованию. При этом, аутентификационная информация может храниться в предмете как в открытом (примеры: магнитные карты, смарт-карты с открытой памятью, электронные таблетки Touch Memory), так и в защищенном виде (смарт-карты с защищенной памятью, USB-токены). В последнем случае требуется ввод PIN-кода для доступа к хранящимся данным, что автоматически превращает предмет в средство двухфакторной аутентификации.

«Активные» предметы, которые обладают достаточными вычислительными ресурсами и способны активно участвовать в процессе аутентификации (примеры: микропроцессорные смарт-карты и USB-токены). Эта возможность особенно интересна при удаленной аутентификации пользователя, поскольку на основе таких предметов можно обеспечить *строгую* аутентификацию. Под этим термином скрывается такой вид аутентификации, при котором секретная информация, позволяющая проверить подлинность пользователя, не передается в открытом виде .

Аутентификация с помощью уникальных предметов обладает и рядом недостатков:

Предмет может быть похищен или отнят у пользователя.

В большинстве случаев требуется специальное оборудование для работы с предметами.

Теоретически возможно изготовление копии или эмулятора предмета.

Биометрическая аутентификация

Биометрическая аутентификация основана на уникальности ряда характеристик человека. Наиболее часто для аутентификации используются следующие характеристики:

Отпечатки пальцев.

Узор радужной оболочки глаза и структура сетчатки глаза.

Черты лица.

Форма кисти руки. 5. Параметры голоса.

Схема кровеносных сосудов лица.

Форма и способ подписи.

В процессе биометрической аутентификации эталонный и предъявленный пользователем образцы сравнивают с некоторой погрешностью, которая определяется и устанавливается заранее. Погрешность подбирается для установления оптимального соотношения двух основных характеристик используемого средства биометрической аутентификации:

1. FAR (False Accept Rate) – коэффициент ложного принятия (т.е. некто успешно прошел аутентификацию под именем легального пользователя). 2. FRR (False Reject Rate) – коэффициент ложного отказа (т.е. легальный пользователь системы не прошел аутентификацию).

Обе величины измеряются в процентах и должны быть минимальны. Следует отметить, что величины являются обратнозависимыми, поэтому аутентифицирующий модуль при использовании биометрической аутентификации настраивается индивидуально – в зависимости от используемой биометрической характеристики и требований к качеству защиты ищется некая «золотая середина» между данными коэффициентами. Серьезное средство биометрической аутентификации должно позволять настроить коэффициент FAR до величин порядка 0,01 – 0,001 % при коэффициенте FRR до 3 – 5%.

В зависимости от используемой биометрической характеристики, средства биометрической аутентификации имеют различные достоинства и недостатки. Например, использование отпечатков пальцев наиболее привычно и удобно для пользователей, но,

теоретически, возможно создание «искусственного пальца», успешно проходящего аутентификацию.

Общий же недостаток биометрической аутентификации – необходимость в оборудовании для считывания биометрических характеристик, которое может быть достаточно дорогостоящим.

Контрольные вопросы:

1. Что такое аутентификация?
2. Что такое идентификация?
3. Перечислить основные виды аутентификации и охарактеризовать их.
4. Перечислить основные методы и механизмы аутентификации.

Практическая работа №9 Применение различных функций хеширования, анализ особенностей хешей.

Цель работы:

Изучить протоколы аутентификации. Ответить на контрольныевопросы.

Теоретические сведения:

В случае удаленной аутентификации (скажем, пользователь намерен получить доступ к удаленному почтовому серверу для проверки своей электронной почты) существует проблема передачи аутентификационной информации по недоверенным каналам связи (через Интернет или локальную сеть). Чтобы сохранить в тайне уникальную информацию, при пересылке по таким каналам используется множество протоколов аутентификации. Рассмотрим некоторые из них, наиболее характерные для различных применений.

Доступ по паролю

Простейший протокол аутентификации – доступ по паролю (Password Access Protocol, PAP): вся информация о пользователе (логин и пароль) передается по сети в открытом виде (рис. 1). Полученный сервером пароль сравнивается с эталонным паролем данного пользователя, который хранится на сервере. В целях безопасности на сервере чаще хранятся не пароли в открытом виде, а их хэш-значения.

Рис.1

Данная схема имеет весьма существенный недостаток: любой злоумышленник, способный перехватывать сетевые пакеты, может получить пароль пользователя с помощью простейшего анализатора пакетов типа sniffer. А получив его, злоумышленник легко пройдет аутентификацию под именем владельца пароля.

По сети в процессе аутентификации может передаваться не просто пароль, а результат его преобразования – скажем, тот же хэш пароля. К сожалению, это не устраняет описанный выше недостаток – злоумышленник с тем же успехом может перехватить хэш пароля и применять его впоследствии.

Недостатком данной схемы аутентификации можно считать и то, что любой потенциальный пользователь системы должен предварительно

зарегистрироваться в ней – как минимум ввести свой пароль для последующей аутентификации. А описанные ниже более сложные протоколы аутентификации типа «запрос-ответ» позволяют в принципе расширить систему на неограниченное количество пользователей без их предварительной регистрации.

Запрос-ответ

В семейство протоколов, называемых обычно по процедуре проверки «запросответ», входит несколько протоколов, которые позволяют выполнить аутентификацию пользователя без передачи информации по сети. К протоколам семейства «запрос-ответ» относится, например, один из наиболее распространенных – протокол CHAP (Challenge-Handshake Authentication Protocol).

Процедура проверки включает как минимум четыре шага (рис. 2):пользователь посылает серверу запрос на доступ, включающий его логин;

сервер генерирует случайное число и отправляет его пользователю; пользователь шифрует полученное случайное число симметричным алгоритмом шифрования на своем уникальном ключе, результат зашифрования отправляется серверу;

сервер расшифровывает полученную информацию на том же ключе и сравнивает с

исходным случайным числом. При совпадении чисел пользователь считается успешно аутентифицированным, поскольку признается владельцем уникального секретного ключа.

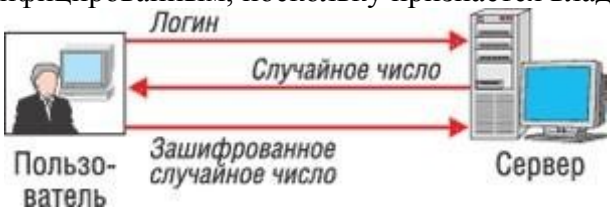

Рис.2

Аутентифицирующей информацией в данном случае служит ключ, на котором выполняется шифрование случайного числа. Как видно из схемы обмена, данный ключ никогда не передается по сети, а лишь участвует в вычислениях, что составляет несомненное преимущество протоколов данногосемейства.

Основной недостаток подобных систем аутентификации – необходимость иметь на локальном компьютере клиентский модуль, выполняющий шифрование. Это означает, что, в отличие от протокола PAP, для удаленного доступа к требуемому серверу годится только ограниченное число компьютеров, оснащенных таким клиентским модулем.

Однако в качестве клиентского компьютера может выступать и смарткарта либо аналогичное «носимое» устройство, обладающее достаточной вычислительной мощностью, например, мобильный телефон. В таком случае теоретически допустима аутентификация и получение доступа к серверу с любого компьютера, оснащенного устройством чтения смарт-карт, смобильного телефона или КПК.

Протоколы типа «запрос-ответ» легко «расширяются» до схемы взаимной аутентификации (рис. 3). В этом случае в запросе на аутентификацию пользователь (шаг 1) посылает свое случайное число (N1). Сервер на шаге 2, помимо своего случайного числа (N2), должен отправить еще и число N1, зашифрованное соответствующим ключом. Тогда перед выполнением шага 3 пользователь расшифровывает его и проверяет: совпадение расшифрованного числа с N1 указывает, что сервер обладает требуемым секретным ключом, т. е.

это именно тот сервер, который нужен пользователю. Такая процедура аутентификации часто называется рукопожатием.

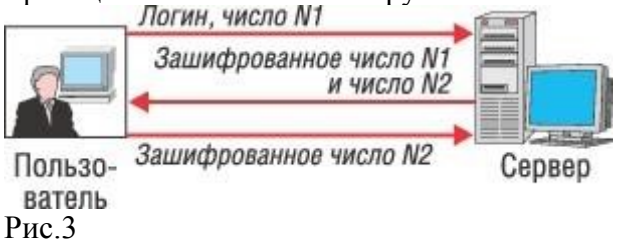

Рис.3

Как видно, аутентификация будет успешна только в том случае, если пользователь предварительно зарегистрировался на данном сервере и какимлибо образом обменялся с ним секретным ключом.

Заметим, что вместо симметричного шифрования в протоколах данного семейства может применяться и асимметричное шифрование, и электронная цифровая подпись. В таких случаях схему аутентификации легко расширить на неограниченное число пользователей, достаточно применить цифровые сертификаты в рамках инфраструктуры открытых ключей.

Протокол Kerberos

Протокол Kerberos, достаточно гибкий и имеющий возможности тонкой настройки под конкретные применения, существует в нескольких версиях. Мы рассмотрим упрощенный механизм аутентификации, реализованный с помощью протокола Kerberos версии 5 (рис. 4):

Рис.4. Схема протокола Kerberos

Прежде всего стоит сказать, что при использовании Kerberos нельзя напрямую получить доступ к какому-либо целевому серверу. Чтобы запустить собственно процедуру аутентификации, необходимо обратиться к специальному серверу аутентификации с запросом, содержащим логин пользователя. Если сервер не находит автора запроса в своей базе данных, запрос отклоняется. В противном случае сервер аутентификации формирует случайный ключ, который будет использоваться для шифрования сеансовсвязи пользователя с еще одним специальным сервером системы: сервером предоставления билетов (TicketGranting Server, TGS). Данный случайный ключ (обозначим его Ku-tgs) сервер аутентификации зашифровывает на ключе пользователя (Kuser) и отправляет последнему. Дополнительная копия ключа Ku-tgs с рядом дополнительных параметров (называемая билетом) также отправляется пользователю зашифрованной на специальном ключе для связи серверов аутентификации и TGS (Ktgs). Пользователь не может расшифровать билет, который необходим для передачи серверу TGS на следующем шаге аутентификации.

Следующее действие пользователя – запрос к TGS, содержащий логин пользователя, имя сервера, к которому требуется получить доступ, и тот самый билет для TGS. Кроме того, в запросе всегда присутствует метка текущего времени, зашифрованная на ключе Ku-tgs. Метка времени нужна для предотвращения атак, выполняемых повтором перехваченных предыдущих запросов к серверу. Подчеркнем, что системное время всех компьютеров, участвующих в аутентификации по протоколу Kerberos, должно быть строго синхронизировано.

В случае успешной проверки билета сервер TGS генерирует еще один случайный ключ для шифрования сеансов связи между пользователем, желающим получить доступ, и целевым сервером (Ku-serv). Этот ключ шифруется на ключе Kuser и отправляется пользователю. Кроме того, аналогично шагу 2, копия ключа Ku-serv и необходимые целевому серверу параметры аутентификации (билет для доступа к целевому серверу) посылаются пользователю еще и в зашифрованном виде (на ключе для связи TGS и целевого сервера – Kserv).

Теперь пользователь должен послать целевому серверу полученный на предыдущем шаге билет, а также метку времени, зашифрованную на ключе Kuserv. После успешной проверки билета пользователь наконец-то считается аутентифицированным и может обмениваться информацией с целевым сервером. Ключ Ku-serv, уникальный для данного сеанса связи, часто применяется и для шифрования пересылаемых в этом сеансе данных.

В любой системе, может быть, несколько целевых серверов. Если пользователю необходим доступ к нескольким из них, он снова формирует запросы к

серверу TGS – столько раз, сколько серверов нужно ему для работы.

Сервер TGS генерирует для каждого из таких запросов новый случайный ключ Ku-serv, т. е. все сессии связи с различными целевыми серверами защищены при помощи разных ключей.

Процедура аутентификации по протоколу Kerberos выглядит достаточно сложной. Однако не стоит забывать, что все запросы и зашифровывание их нужными ключами автоматически выполняет ПО, установленное на локальном компьютере пользователя. Вместе с тем необходимость установки достаточно сложного клиентского ПО – несомненный недостаток данного протокола.

Впрочем, сегодня поддержка Kerberos встроена в наиболее распространенные ОС семейства Windows, начиная с Windows 2000, что нивелирует данный недостаток.

Второй недостаток – необходимость в нескольких специальных серверах (доступ к целевому серверу обеспечивают как минимум еще два, сервер аутентификации и TGS). Однако в системах с небольшим числом пользователей все три сервера (аутентификации, TGS и целевой) могут физически совмещаться на одном компьютере.

Вместе с тем следует подчеркнуть, что сервер аутентификации и TGS должны быть надежно защищены от несанкционированного доступа злоумышленников. Теоретически злоумышленник, получивший доступ к TGS или серверу аутентификации, способен вмешаться в процесс генерации случайных ключей или получить ключи всех пользователей и, следовательно,инициировать сеансы связи с любым целевым сервером от имени любого легального пользователя.

Выбор того или иного протокола зависит в первую очередь от важности информации, доступ к которой предоставляется по результатам аутентификации. Другой критерий – удобство использования. И здесь, как и везде, полезно соблюдать разумный баланс. Иногда, если нет особых требований к секретности процедуры аутентификации, вместо надежного (но непростого в реализации) протокола типа Kerberos лучше использовать

«элементарный» парольный протокол PAP, простота и удобство применения которого часто перевешивают все его недостатки.

Контрольные вопросы:

1. Охарактеризовать протокол аутентификации «доступ по паролю».
2. Охарактеризовать протокол аутентификации «запрос-ответ».
3. Охарактеризовать протокол аутентификации Kerberos.
4. Электронная цифровая подпись. Схема электронной подписи RSA.