

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

Методические указания
по проведению практических занятий
по дисциплине «КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»
для студентов специальности 10.05.01 Компьютерная безопасность
специализация «Информационно-аналитическая и техническая экспертиза
компьютерных систем»

Ставрополь
2026

СОДЕРЖАНИЕ

Практическая работа № 1. Решение задач по формальному описанию криптографических протоколов	3
Практическая работа № 2. Определение уязвимостей протоколов.....	5
Практическая работа № 3. Составление формального описания базовых криптографических протоколов.	8
Практическая работа № 4. Проверка чисел на простоту.....	12
Практическая работа № 5. Проверка чисел на простоту. Ветоятностные методы.....	14
Практическая работа №6 Изучение хэш-функций семейств SHA, MD и хэш-функции ГОСТ.....	17
Практическая работа № 7 Решение задач по разделению секрета.....	19
Практическая работа № 8 Использование криптографических примитивов в контексте криптографических протоколов.....	21
Практическая работа № 9 Изучение протоколов аутентификации.....	23
Практическая работа № 10 Изучение протоколов односторонней аутентификации.....	23
Практическая работа № 11 Изучение протоколов двусторонней аутентификации.....	27
Практическая работа № 12 Изучение протоколов электронной подписи.....	29
Практическая работа № 13 Сравнение протоколов ЭП DSA и ГОСТ.....	32
Практическая работа № 14 Изучение схем стираемой электронной подписи.....	35
Практическая работа № 15 Формальное описание протоколов слепой, совместной и подтверждаемой ЭП.....	37
Практическая работа № 16 Изучение и формальное описание протоколов голосования.....	39
Практическая работа № 17 Изучение программных средств, реализующих протоколы электронной коммерции.....	42
Практическая работа № 18 Решение задач по теме «Протоколы разделения секрета».....	44

Практическая работа № 1. Решение задач по формальному описанию криптографических протоколов

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 $A^1, A^2, \dots, A^N$ используют некоторый симметричный криптоалгоритм $y = E(x, k)$ с секретным ключом. Как известно для того, для связи каждой пары абонентов необходимо использовать уникальную пару ключей. 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.

Пусть имеется защищенный справочник сети S . Рассмотрим поле $GF(p)$, p - большое простое число. В $GF(p)$ имеется $(p - 1)$ ненулевых элементов. Пусть α - примитивный элемент и $\alpha^0, \alpha^1, \dots, \alpha^{p-2}$ образуют всю группу $GF(p)$ по умножению. Пусть $x_1, x_2, \dots, x_N$ - случайные числа, которые выбираются каждым абонентом $A_1, A_2, \dots, A_N$ и каждый из них хранит свое число в секрете. Далее каждый абонент из $A_1, A_2, \dots, A_N$ вычисляет $y_i = \alpha^{x_i} \pmod{p}$ и публикует значение y_i в общем справочнике S . Тогда секретный ключ для переписки абонентов сети A_i и A_j формируется следующим образом.

1. A_i берет в общем справочнике число y_j .
2. A_j берет в общем справочнике число y_i .
3. A_i вычисляет $k_{ij} = y_j^{x_i} \pmod{p}$.
4. A_j вычисляет $k_{ji} = y_i^{x_j} \pmod{p}$.
5. $k = k_{ij} = k_{ji}$ - их общий секретный ключ.

В самом деле,

$$k_{ji} = y_i^{x_j} \pmod{p} = (\alpha^{x_i})^{x_j} \pmod{p} = y_j^{x_i} \pmod{p} = k_{ij}$$

Для вычисления этого ключа противнику E надо уметь решать задачу логарифмирования в конечных полях, тогда как в настоящее время не разработано полиномиальных алгоритмов дискретного логарифмирования в произвольном конечном поле. Необходимо учитывать следующие условия, налагаемые на числа p и a .

Число p должно быть сильным простым числом

Число a , вообще говоря, может не являться примитивным элементом $GF(p)$. Достаточно того, чтобы оно образовывало достаточно большую подгруппу группы $GF(p)$ по умножению.

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 вычислить общий сеансовый ключ для пары участников.

Отчет по лабораторной работе представить в виде расчетного файла математического 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 посередине».

Практическая работа № 2. Определение уязвимостей протоколов

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 же УЗ, 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 навязывания ложных данных. Имитовставка представляет собой отрезок информации фиксированной длины, полученный из открытых данных и ключа. 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 расшифровании 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 при создании 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.

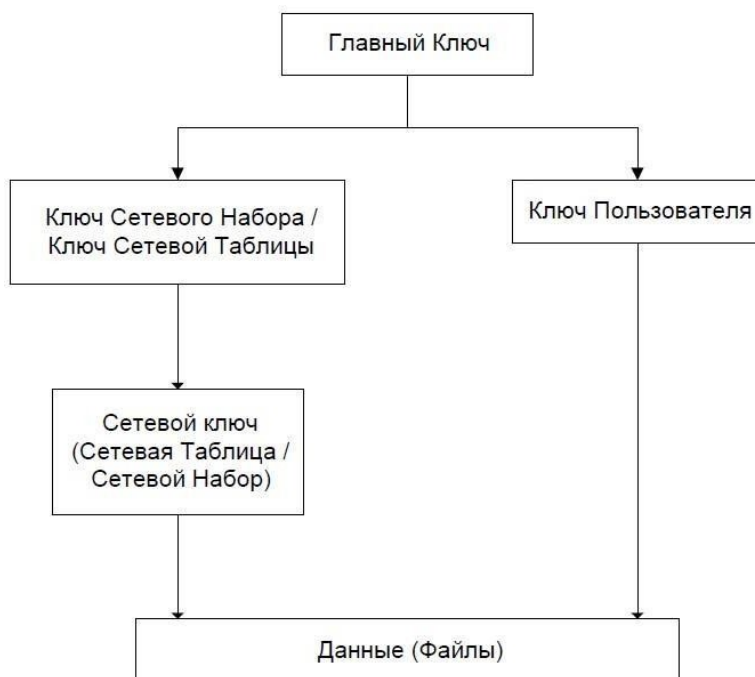


Рисунок 16.1 - Иерархическая структура ключей

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. Сделать выводы о закономерностях изменения размера файла.
12. Изучить возможности гарантированного удаления (уничтожения) выбранных файлов без возможности их восстановления) пакета КРИПТОН®

Шифрование.

13. 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 № 3. Составление формального описания базовых криптографических протоколов

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 сети может 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 при создании 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 главном ключе, 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 узлов криптографической сети. В результате получается множество Ключей Сетевого Набора (NNNNN.KEY) и Сетевых Наборов (NNNNN.SYS) для узлов криптографической 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 тем Сетевым 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 SCSI.

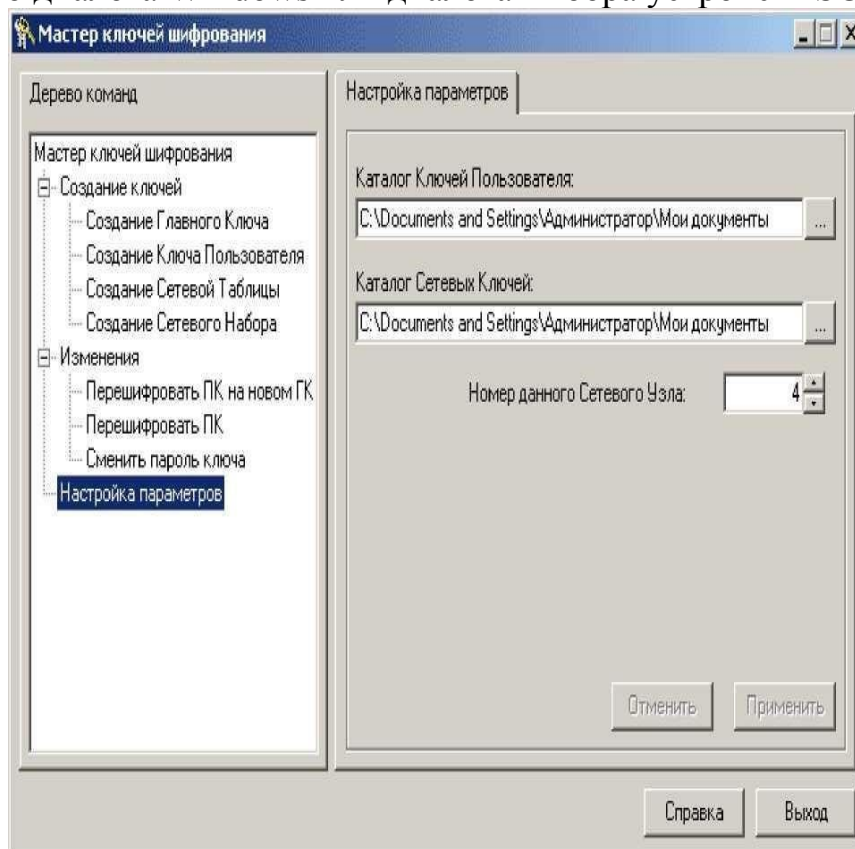


Рисунок 17.1 - Мастер ключей шифрования пакета КРИПТОН® Шифрование
"Каталог Сетевых Ключей".

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
обзор каталогов, по которой вызывается стандартный диалог обзора папок

Windows. Сетевые Ключи хранятся в Сетевых Наборах в файлах
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 следует
нажать 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 и самих СН; - **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 файлами в криптографической сети.
7. Создание сетевых наборов для ограниченного числа узлов криптографической сети.

Практическая работа № 4. Проверка чисел на простоту

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 обмениваться 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 для передачи.

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 удостоверить 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 удостоверить платежные поручения другого, а программист не может удостоверить 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, открытый 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, доставляю-
150

щего 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 № 5. Проверка чисел на простоту. Вероятностные методы

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 всемописанным 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.

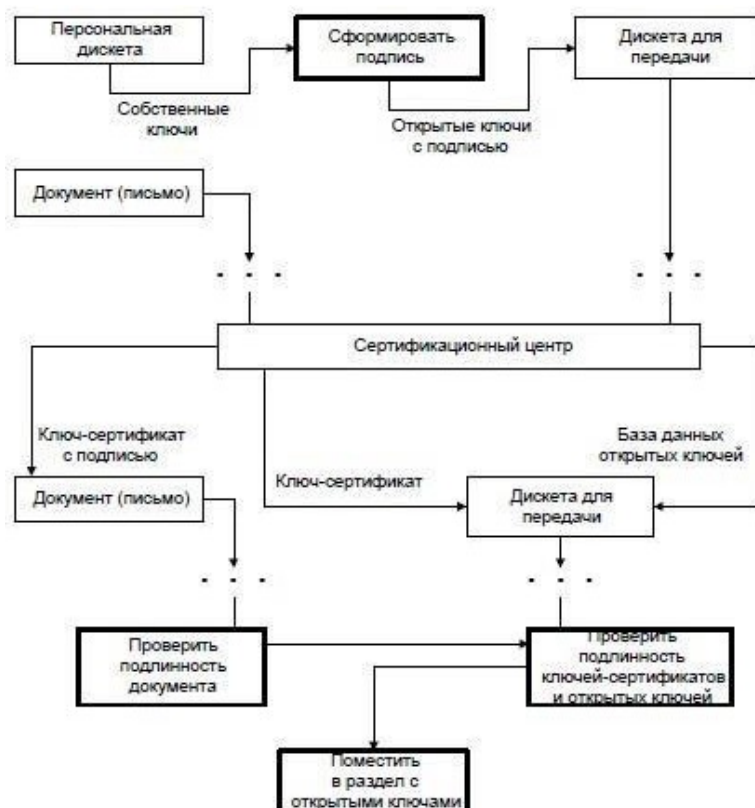


Рисунок 19.1 - Обмен открытыми ключами через сертификационный центр

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 сСЦ (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 6. Изучение хэш-функций семейств SHA, MD и хэш-функции ГОСТ

Цель работы:

Создать криптографическую систему шифрования данных, которая базируется на алгоритме шифрования *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 C_i и D_i , $i = 1, 2, \dots, 16$. Для формирования C_i и D_i применяют операции циклического сдвига влево на один или два бита. Величина сдвига определяется стандартной таблицей. Операции сдвига для C_i и D_i 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

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \quad i = 1, 2, \dots, 16.$$

Функция $f(R_{i-1}, K_i)$ представляет собой некоторую суперпозицию простых преобразований, детальное описание которых будет дано ниже. Отметим, что функция $f(R_{i-1}, K_i)$ является нелинейной. Нелинейность функции $f(R_{i-1}, K_i)$ обеспечивается S -блоками.

Алгоритм вычисления функции шифрования $f(R_{i-1}, K_i)$

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

Вычисляется новый блок размером в 48 бит по

$$\text{формуле } V = E(R_{i-1}) \oplus K_i.$$

Блок V размером в 48 бит разбивается на восемь блоков V_j , $j=1, 2, \dots, 8$, по шесть битов каждый

$$V = V_1 V_2 V_3 V_4 V_5 V_6 V_7 V_8.$$

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

$$V' = S_1(V_1) S_2(V_2) S_3(V_3) S_4(V_4) S_5(V_5) S_6(V_6) S_7(V_7) S_8(V_8)$$

размером в 32 бита.

Алгоритм преобразования блока V_j размером в шесть бит в блок $S_j(V_j)$ размером в четыре бита следующий. Каждая функция S_j 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

$$V_j = b_1 b_2 b_3 b_4 b_5 b_6 = 110010, \quad j = 1, 2, \dots, 8.$$

Тогда имеет место:

биты $b_1 b_6 = 10$ формируют номер столбца (двоичное число 10 равно

десятичному числу 2) таблицы S_j , биты $b_2 b_3 b_4 b_5 = 1001$ формируют номер строки (двоичное число 1001 равно десятичному числу 9) таблицы S_j . Блок $V_j = b_1 b_2$

b3 b4 b5 b6

= 110010 заменяют двоичным значением числа таблицы S_j , которое находится на пересечение строки с номером $b1b6 = 10$ (2) со столбцом с номером $b2 b3 b4 b5 = 1001$ (9). Преобразуя каждое $B_j, j = 1, 2, \dots, 8$, 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 формирования 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 № 7. Решение задач по разделению секрета

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

Выбираем число e , взаимно простое с $\phi(n)$. 4. Вычисляем число d из уравнения $de \equiv 1 \pmod{\phi(n)}$.

Определяются открытые ключи e и n . Определяются закрытые ключи d, p, q и $\phi(n)$.

Алгоритм шифрования

Дан текст сообщения М.
Шифрованный текст С вычисляется по формуле $C = EK(M) = Me \bmod n$.

Алгоритм дешифрования

Дан шифрованный текст С.
Текст сообщения М вычисляется по формуле

$$M = DK(C) = Cd \bmod n \equiv (Me) d \bmod n \equiv M$$

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 формирования цифровой подписи с применением алгоритма RSA.
14. Сформулировать теорему Ферма.
15. Сформулировать теорему Эйлера.

Практическая работа № 8. Использование криптографических примитивов в контексте криптографических протоколов

Цель работы:

Создать программу, которая вычисляет 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

$$L \equiv 448 \bmod 512.$$

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

$M_1, M_2, \dots, M_K$,

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

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

Для записи промежуточных значений в 128 бит при вычислении хэшфункции 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 , $i=1, 2, \dots, K$, в 512 бит в профиль H_i длиной в 128 бит. Напомним, что блок M_i разбит на 16 частей (массивов) по 32 бита $X[0], X[1], \dots, X[k]$, $k=0, 1, \dots, 15$.

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 $448 \bmod 512$. 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: $M_1, M_2, \dots, M_K$, где K есть число таких блоков, $L = K \cdot 512$. В свою очередь каждый блок M_i разбивают на 16 частей (массивов) по 32 бита $X[0], X[1], \dots, X[k]$, $k=0, 1, \dots, 15$.

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. Логiku модуля функциисжатия алгоритма можно представить в виде следующего цикла

$$H_i = H_{i-1} + H$$

Цикл от $t=0$ до $t=79$ шаг

1 Начало цикла

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

$$b = a$$

$$a = H \quad \text{Конец цикла}$$

$$H_{i+1} = H_i + H$$

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 $MD5$ и $SHA-1$ для вычисления профиля начальной информации. Подготовить отчет по работе. В отчете описать алгоритмы $MD5$ 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 $MD5$.
5. Назвать число раундов в алгоритме $MD5$.
6. Назвать число шагов в каждом раунде алгоритма $MD5$.
7. Назвать, какие операции используются в функции сжатия алгоритма $MD5$.
8. Сколько примитивных функций используются в алгоритме $MD5$?
9. Сколько элементов в массиве T , который используется в алгоритме $MD5$?
10. Назвать, какие операции используются в функции сжатия алгоритма $SHA-1$.
11. Перечислить основные постоянные данные, которые используются в алгоритме
12. Назвать число раундов в алгоритме $SHA-1$.
13. Сколько примитивных функций используются в алгоритме $SHA-1$.
14. Длина входного блока в битах для функции сжатия алгоритма $SHA-1$.
15. Длина буфера в битах для функции сжатия алгоритма $SHA-1$.
16. Число шагов в каждом раунде алгоритма $SHA-1$.
17. Число регистров буфера в алгоритме $SHA-1$.
18. Чем инициализируются регистры буфера алгоритма $SHA-1$?
19. Чем инициализируются регистры буфера алгоритма $MD5$?

Практическая работа № 9. Изучение протоколов аутентификации

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

$$x_{n+1} = (ax_n + b) \bmod m;$$

$$x_{n+1} = (ax_n^2 + bx_n + c) \bmod m; \quad x_{n+1} = (ax_n^3 + bx_n^2 + cx_n + d) \bmod m.$$

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 № 10 Изучение протоколов односторонней аутентификации

Цель работы:

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

Теоретические сведения:

Алгоритм формирования схемы Эль-Гамала

Выбираем простое число p .

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

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

Открытым ключом схемы Эль-Гамала 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

$$a \equiv g^k \pmod{p}$$

p . Из

уравнения

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

$(p-1)$ определяем b

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

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 хеш-функция.

Проверка подписи

Дано сообщение M и подпись (a, b) .

Вычисляем

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

Вычисляем

$$g^M \pmod{p}.$$

Если значение $y^a a^b \pmod{p}$

совпало с $g^M \pmod{p}$, 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 $\phi(n)$.

Определяются открытые ключи e и n . 4. Закрытым ключом

вычисляется $C = M^d \pmod{n}$.

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

$$C^e = (M^d)^e \pmod{n} \equiv M,$$

Цифровая подпись Шнорра

ЭЦП Шнорра основана на сложности задачи дискретного логарифмирования.

Поэтому все параметры схемы Шнорра должны удовлетворять условиям существования 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 , $g \in Z_p$, где Z_p – класс вычетов по модулю p , h – хэш-функция, x – случайное число из интервала $[1, q-1]$, 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

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

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

$$\text{и } e'r' = g^s y^e \bmod p, \quad e' = h(r', m).$$

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

$$m_1 \equiv C^{(p+1)/4} \bmod p,$$

$$m_2 \equiv (p - C^{(p+1)/4}) \bmod p, \quad m_3 \equiv C^{(q+1)/4} \bmod q, \quad m_4 \equiv (q - C^{(q+1)/4}) \bmod q.$$

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

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

Возможными сообщениями, которые были отправлены, являются значения

M_1, M_2, M_3 и M_4 ,

которые вычисляются по формулам

$$M_1 = (am_1 + bm_3) \bmod$$

$$n, M_2 = (am_1 + bm_4)$$

$$\bmod n,$$

$$M_3 = (am_2 + bm_3) \bmod n, \quad M_4 = (am_2 + bm_4) \bmod n.$$

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

$m_0, m_1, m_2, \dots, m_{k-1}$

из k бит

$m_i \in \{0,1\}, i = 0, 1, \dots, k-1$.

Алгоритм схемы подписи Рабина

Формирование подписи

Генерируется случайная последовательность R длиной из t бит

$R = (r_0, r_1, \dots, r_{t-1}), r_i \in \{0,1\}, i = 0, 1, \dots, t-1$.

Сообщение M объединяется с

последовательностью $RM||R = (m_0, m_1, m_2, \dots, m_{k-1}, r_0, r_1, \dots, r_{t-1})$.

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

Проверяются условия

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

Если условия не выполняются, то перейти на шаг 1.

Вычисляются значения

$z_p \equiv a^{(p+1)/4} \pmod{p}, z_q \equiv a^{(q+1)/4} \pmod{q}$.

По китайской теореме об остатках вычисляется значение

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

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

Проверка подписи

Вычисляется значение

$a' a' \equiv Z^2 \pmod{n}$.

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

Задания:

Реализовать ЭЦП на базе алгоритма Эль-Гамала и алгоритма RSA . При формировании ЭЦП на базе алгоритма RSA использовать результаты лабораторной работы

№ 2. Предусмотреть режимы формирования параметров криптосистемы Эль-Гамала. Программу оформить, как интегрируемую среду с удобным интерфейсом формирования ЭЦП и ее проверки. Подготовить отчет, в который включить алгоритмы формирования

системы Эль-Гамала, 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 Эль-Гамала.
2. Перечислить секретные параметры системы Эль-Гамала.
3. Перечислить открытые параметры системы Эль-Гамала.
4. На какой достаточно трудной задаче из теории чисел

- базируется криптографическая система Эль-Гамала.
5. Описать схему формирования ЭЦП с использованием алгоритма ЭльГамала.
 6. Описать схему проверки ЭЦП с использованием алгоритма ЭльГамала.
 7. Описать схему формирования цифровой подписи с применением алгоритма RSA.
 8. Описать схему проверки цифровой подписи с применением 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-Гамала действительно дает верные цифровые подписи.

Практическая работа № 11. Изучение протоколов двусторонней аутентификации

Цель работы:

Ознакомиться 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

$$g^a \bmod p.$$

Абонент B посылает абоненту A сообщение

$$g^b \bmod p.$$

Абонент A вычисляет общий ключ

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

Абонент B вычисляет общий ключ

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

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 дешифрования D_A и D_B , которые обладают свойствами

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

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

$g^a \bmod p$. Абонент B

- вычисляет общий ключ

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

- используя закрытую функцию D_B , создает подпись (сообщение) $D_B(g^a \bmod p, g^b \bmod p)$;
- используя ключ k , шифрует подпись $E_K(D_B(g^a \bmod p, g^b \bmod p))$, где E_K общая функция шифрования;

- отправляет абоненту A сообщение $g^b \bmod p, E_K(D_B(g^a \bmod p, g^b \bmod p))$ Абонент A

- вычисляет общий ключ

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

- используя закрытую функцию D_A , создает подпись (сообщение)

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

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

- отправляет абоненту B сообщение

$E_K(D_A(g^a \bmod p, g^b \bmod p))$

- проверяет подпись B , вычисляя

$E_B(D_K(E_K(D_B(g^a \bmod p, g^b \bmod p)))) = (g^a \bmod p, g^b \bmod p)$, где D_K - общая функция дешифрования.

Абонент B проверяет подпись A , вычисляя

$E_A(D_K(E_K(D_A(g^a \bmod p, g^b \bmod p)))) = (g^a \bmod p, g^b \bmod p)$.

Задания:

Реализовать два простейших протокола. Протокол Диффи-Хеллмана 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 № 12. Изучение протоколов электронной подписи

Цель работы:

Создать криптографическую систему шифрования данных, которая базируется на алгоритме шифрования 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

$X_0, X_1, \dots, X_7$

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

$X_i, i = 1, 2, \dots, 7$, то есть

$X_0 = (w_{32}, \dots, w_1),$

$X_1 = (w_{64}, \dots, w_{33}),$

...

$X_7 = (w_{256}, \dots, w_{225}).$

Очередной блок начальных данных T_i длиной в 64 бит

$T_i(a_1, a_2, \dots, a_{32}, b_1, b_2, \dots, b_{32})$

засылается в накопители N_1 и N_2 по следующей схеме

$N_1 = (a_{32}, \dots, a_1), N_2 = (b_{32}, \dots, b_1).$

Содержимое N_1 суммируем по модулю 2^{32} с содержимым X_0 , результат засылается в N_1 .

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

N_2 . Содержимое N_1 пересылается в N_2 , а результат 7 шага записывается в N_1 .

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 X_0):

1-й раунд шифрования из 8 шагов;

$X_0, X_1, \dots, X_7$ 2-й раунд шифрования из 8

шагов; $X_0, X_1, \dots, X_7$ 3-й раунд шифрования

из 8 шагов; $X_0, X_1, \dots, X_7$

4-й раунд шифрования из 8 шагов

$X_7, X_6, \dots, X_0$

Результат последнего 32 шага шифрования, т.е. результат 8 шага алгоритма, сохраняется в накопителе N_2 , а в N_1 сохраняется результат 31 шага.

Результат шифрования текущего блока T_i содержится в накопителях N_1 с 1-го по 32 разряд и N_2 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

$X_0, X_1, \dots, X_7$

записывается сгенерированный ключ длиной в 256 бит по 32 бита в каждый $X_i, i = 1, 2, \dots, 7$, то есть

$$X_0 = (w_{32}, \dots, w_1),$$

$$X_1 = (w_{64}, \dots, w_{33}),$$

...

$$X_7 = (w_{256}, \dots, w_{225}).$$

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

$$S(s_1, s_2, \dots, s_{32}, s_{33}, s_{34}, \dots, s_{64})$$

и засылается в накопители N_1 и N_2 по следующей схеме

$$N_1 = (s_{32}, \dots, s_1)$$

$$N_2 = (s_{64}, \dots, s_{33}).$$

Содержимое N_1 и N_2 шифруется в режиме простой замены.

Содержимое накопителей N_1 и N_2 , в которых находится результат шифрования, переписываются соответственно в накопители N_3 и N_4 .

Содержимое накопителя N_4 суммируется по модулю $(2^{32}-1)$ с накопителем N_6 , в котором хранится стандартная константа

$$C_2 = 000000001000000001000000001000000001.$$

Результат суммирования записывается в накопитель N_4 .

Содержимое накопителя N_3 суммируется по модулю 2^{32} с накопителем N_5 , в котором хранится стандартная константа $C_1 = 100000000100000000100000000100000000$.

Результат суммирования записывается в накопителя N_3 . Содержимое накопителя N_3 переписывается в N_1 .

Содержимое накопителя N_4 переписывается в N_2 .

Содержимое N_1 и N_2 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

$X_0, X_1, \dots, X_7$

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

$X_i, i = 1, 2, \dots, 7$, то есть

$X_0 = (w_{32}, \dots, w_1),$

$X_1 = (w_{64}, \dots, w_{33}),$

...

$X_7 = (w_{256}, \dots, w_{225}).$

Генератором последовательности случайных чисел формируется

двоичная последовательность длиной в 64 бита (синхропосылка)

$S(s_1, s_2, \dots, s_{32}, s_{33}, s_{34}, \dots, s_{64})$

и засылается в накопители N_1 и N_2 по следующей схеме

$N_1 = (s_{32}, \dots, s_1),$

$N_2 = (s_{64}, \dots, s_{33}).$

Содержимое N_1 и N_2 шифруется в режиме простой замены. Результат четвертого шага образует текущий блок гаммы G .

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

C по следующему правилу

$C(c_1, c_2, \dots, c_{32}, c_{33}, c_{34}, \dots, c_{64})$

в режиме гаммирования с обратной связью.

7. Зашифрованный блок C является исходным состоянием накопителей N_1 и N_2 для выработки следующего блока гаммы. Последовательность битов

$c_1, c_2, \dots, c_{32}, c_{33}, c_{34}, \dots, c_{64}$

засылается в накопители N_1 и N_2 по следующей схеме

$N_1 = (c_{32}, \dots, c_1),$

$N_2 = (c_{64}, \dots, c_{33}).$

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

$X_0, X_1, \dots, X_7$

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

$X_i, i = 1, 2, \dots, 7$, то есть

$X_0 = (w_{32}, \dots, w_1),$

$X_1 = (w_{64}, \dots, w_{33}),$

...

$X_7 = (w_{256}, \dots, w_{225}).$

Очередной блок начальных данных T_i длиной в 64 бит

$T_i(a_1, a_2, \dots, a_{32}, b_1, b_2, \dots, b_{32})$

засылается в накопители N_1 и N_2 по следующей схеме

$N_1 = (a_{32}, \dots, a_1), N_2 = (b_{32}, \dots, b_1).$

Содержимое N_1 и N_2 подвергается преобразованию, которое соответствует первым шестнадцати циклам алгоритма в режиме простой замены.

Полученное после шестнадцати циклов содержимое накопителей в N_1 и N_2 суммируются по модулю 2 со вторым блоком открытых данных.

Результат суммирования заносится в накопители N_1 и N_2 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 завершения 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 Кергоффа 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 № 13. Сравнение протоколов ЭП DSA и ГОСТ

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 любых 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,

$$4290 = 3 \cdot 2 \cdot 5 \cdot 11 \cdot 13,$$

В общем случае каноническим разложением целого числа a называется представление в виде $a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$,

где p_i , $i = 1, 2, \dots, k$, – простые числа. Например, $133100 = 2^2 \cdot 5^2 \cdot 11^3$.

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

$$a_1, a_2, \dots, a_k$$

называется любое целое число d , такое, что

$$d|a_1, d|a_2, \dots, d|a_k.$$

Определение. Наибольшим общим делителем (НОД) целых чисел $a_1, a_2, \dots, a_k$ 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

$$d \leq p,$$

~

то число p – простое.

Определение. Числа $a_1, a_2, \dots, a_k$ называются взаимно простыми, если наибольший общий делитель этих чисел равен 1.

Определение. Числа $a_1, a_2, \dots, a_k$ называются попарно взаимно простыми, если $(a_i, a_j) = 1$, $i \neq j$, $1 \leq i \leq k$, $1 \leq j \leq k$.

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

$$= bq_0 + r_1;$$

Делим число b на число r_1 , имеем

$$= r_1q_1 + r_2;$$

Делим число r_1 на число r_2 , запишем

$$r_1 = r_2q_2 + r_3;$$

Делим число r_2 на число r_3 , получаем

$$r_2 = r_3q_3 +$$

$$r_4; r_{t-1} = r_tq_t$$

$$+ r_{t+1}.$$

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

Кратко алгоритм Евклида 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

$$r, 0 < r \leq a.$$

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

$$1173 = 323 \cdot 3 + 204;$$

Делим число 323 на число 204, получаем $323 = 204 \cdot 1 +$

$$119;$$
 Делим число 204 на число 119, получаем

$$204 = 119 \cdot 1 + 85;$$

Делим число 119 на число 85,

$$\text{получаем } 119 = 85 \cdot 1 + 34;$$

Делим число 85 на число 34,

$$\text{получаем } 85 = 34 \cdot 2 + 17;$$

Делим число 34 на число 17,

$$\text{получаем } 34 = 17 \cdot 2 + 0;$$

Итак, в итоге

$$\text{получаем } (1173, 323)$$

$$= 17.$$

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.

Практическая работа № 14. Изучение схем стираемой электронной подписи

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 x и 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 словами наибольший общий делитель двух чисел можно представить в виде линейной комбинации этих чисел с целыми коэффициентами.

Схема расширенного алгоритма Евклида

Определить

$$a_0 = 1, a_1 = 0, b_0 = 0,$$

$$b_1 = 1, \alpha = a, \beta = b.$$

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

$$a = qb + r.$$

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

Определяем

$$a = b, b = r, t = a_0, a_0 = a_1,$$

$$a_1 = t - a_1q, t = b_0, b_0 = b_1; b_1 = t - b_1q;$$

Возвращаемся на шаг 2. 6. Определяем $x = x_0, y = y_0, d = \alpha x + \beta y$.

Пример. Дано $a = 1769, b = 551$. Используя расширенный алгоритм Евклида, найти целые числа x и y такие, что

$$d = ax + by,$$

где d – наибольший общий делитель чисел a и b .

I этап последовательности вычислений

$$a_0 = 1, a_1 = 0, b_0 = 0, b_1 = 1, \alpha = 1769, \beta = 551.$$

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

$$a = 551, b = 116, t = a_0 = 1, a_0 = a_1 = 0, a_1 = t - a_1q = 1 - 0 \cdot 3 = 1$$

$$t = b_0 = 0, b_0 = b_1 = 1, b_1 = t - b_1q = -3;$$

В результате текущего шага получили следующие промежуточные значения параметров

$$a = 551, b = 116, a_0 = 0, a_1 = 1, b_0 = 1, b_1 = -3.$$

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

II этап последовательности вычислений

Значение параметров

$$a = 551, b = 116, a_0 = 0,$$

$$a_1 = 1, b_0 = 1, b_1 = -3.$$

Частное от деления $q = a/b = 551/116 = 4, -3$, а остаток от деления $r = 87$.

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

Определяем

$$a = 116, b = 87, t = a_0 = 0, a_0 = a_1 = 1, a_1 = t - a_1q = 0 - 1 \cdot 4 = -4, t = b_0 = 1, b_0 = b_1 =$$

$$b_1 = t - b_1q = 1 - (-3) \cdot 4 = 13.$$

В результате текущего шага получили следующие промежуточные значения параметров

$$a = 116, b = 87, a_0 = 1, a_1 = -4, b_0 = -3, b_1 = 13.$$

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

Значение параметров

$$a = 116, b = 87, a_0 = 1, a_1 = -4, b_0 = -3, b_1 = 13.$$

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

$$a = 87, b = 29, t = a_0 = 1, a_0 = a_1 = -4, a_1 = t - a_1q = 1 - (-4) \cdot 1 = 5; t = b_0 = -3, b_0 = b_1 = 13; b_1 = t - b_1q = -3 - (13) \cdot 1 = -16.$$

В результате текущего шага получили следующие промежуточные значения параметров

$$a = 87, b = 29, a_0 = -4, a_1 = 5, b_0 = 13, b_1 = -16.$$

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

IV этап последовательности вычислений

Значение параметров

$$a = 87, b = 29, a_0 = -4, a_1 = 5, b_0 = 13, b_1 = -16.$$

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

$$a = 87, b = 29, t = a_0 = -4, a_0 = a_1 = 5, a_1 = t - a_1q = -4 - 5 \cdot 3 = -19, t = b_0 = 13, b_0 = b_1 = -16,$$

$$b_1 = t - b_1q = 13 - (-16) \cdot 3 = 61.$$

В результате текущего шага получили следующие промежуточные значения параметров

$$a = 87, b = 29, a_0 = 5, a_1 = -19, b_0 = -16, b_1 = 61.$$

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

$$d = \alpha x + \beta y, \text{ где}$$

$$x = x_0 = 5, y = y_0 = -16, \alpha = 1769, \beta = 551.$$

Подставляя значение параметров, получаем

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

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 № 15. Формальное описание протоколов слепой, совместной и подтверждаемой ЭП

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.

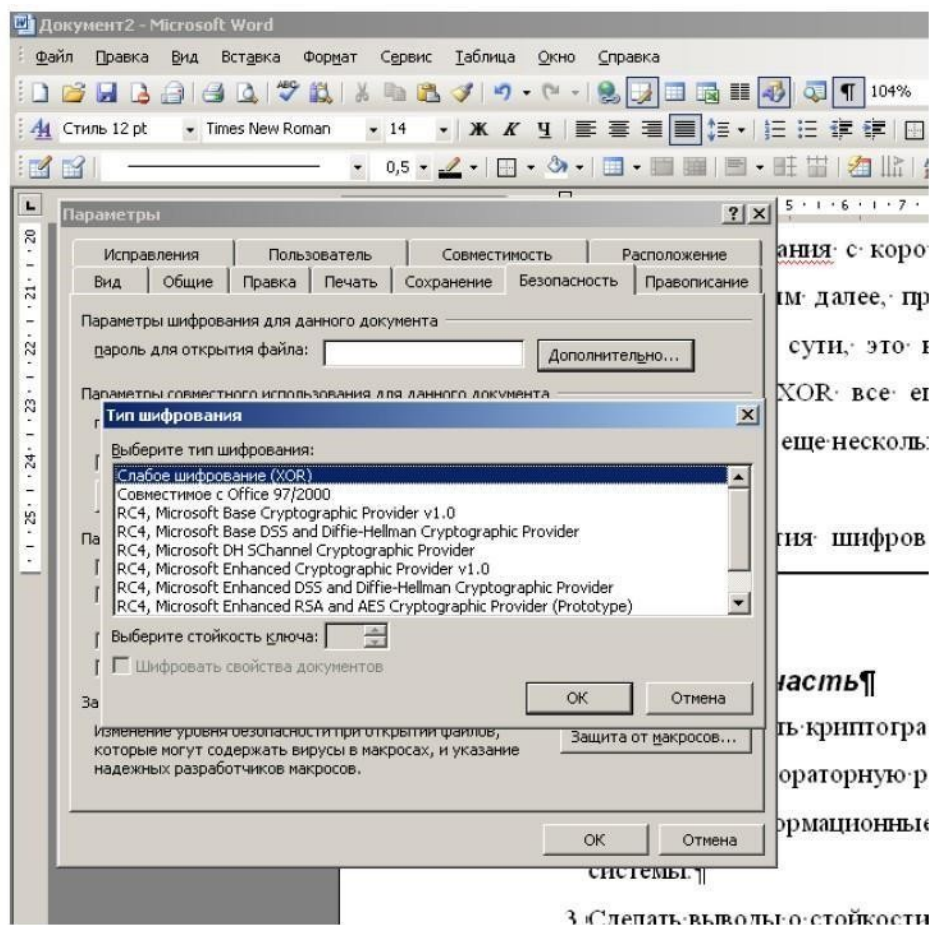


Рисунок 9.1 - Диалог «Тип шифрования» в Microsoft Word

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, Y' - 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 Ъ Щ Ф Я Я Е Ц Ы И Я Щ В Я Э Ч Я Я И Е Ы
Л Ш Е Щ Ъ П З Я Е Ы Г В М К Ч Ъ Я Ц В М Е Т Э Э
З Т Л В М Т Л Я Ъ 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 № 16. Изучение и формальное описание протоколов голосования

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 $m \ll l$), 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 блокшифртекста 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.



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 256-битовыми величинами) определяется следующим образом:

Генерируются 4 ключа шифрования K_j , $j = 1 \dots 4$, 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-

106

тельность S_4, S_3, S_2, S_1 длиной 256 бит запоминается во временной переменной S .

Значение H_i 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:

H_n - 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:

$H = f(Z \parallel M', f(L, f(M', H_n)))$.

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;
Changed1bit - для файлов, с измененным одним битом; - Changed 1byte - 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'e) 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 четырехбит 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 № 17. Изучение программных средств, реализующих протоколы электронной коммерции

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

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.

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.

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.

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:

наименование фирмы);
секретный ключ (имя файла секретного
ключа); собственно код ЭЦП.

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

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

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 (*.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 хранения 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 № 18 Решение задач по теме «Протоколы разделения секрета»

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 Lab13. 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 своим ключом.

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.