МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

Факультет информационных технологий

Кафедра «Инфокогнитивные технологии»

**Лабораторная работа 4**

По дисциплине «Защита информации»

Направление подготовки 09.03.03 «Прикладная информатика»

Профиль «Корпоративные информационные системы»

**Выполнил:**

студент группы 201-361

Погудин Александр

Москва 2023

Цель работы: реализовать простое клиент-серверное приложение, позволяющее аккумулировать короткие анонимные сообщения (систему электронного голосования) согласно следующей схеме:

Изображение выглядит как зарисовка, рисунок, круг, диаграмма

Автоматически созданное описание

Здесь: A – пользователь (избиратель), B – регистратор, C – счетчик, x –сообщение (голос), r – известное только участнику A случайное число, (e, n) – открытый ключ банка. Пренебрегите реализацией правильных механизмов распределения, хранения и сертификации ключей.

**Введение**

**Клиент-серверное приложение** - это приложение, состоящее из двух основных компонентов: сервера и клиента. Сервер представляет собой компьютер, который предоставляет какую-то услугу (например, базу данных, файлы, веб-страницы и т.д.). Клиент, в свою очередь, представляет собой компьютер, который запрашивает эту услугу через сеть (обычно Интернет).

Таким образом, сервер обеспечивает доступ к каким-то ресурсам или услугам, а клиенты используют эти ресурсы или услуги, подключаясь к серверу по определенному протоколу. Взаимодействие между клиентами и сервером обычно происходит по сети в реальном времени.

Примеры клиент-серверных приложений:

- Интернет-магазин, где сервер предоставляет каталог товаров, а клиенты могут добавлять товары в корзину и оформлять заказы.

- Электронная почта, где сервер предоставляет возможность отправлять и получать электронные письма, а клиенты используют почтовые программы для взаимодействия с сервером.

- Онлайн-игры, где сервер обеспечивает игровое поле, а клиенты через Интернет играют в игру и обмениваются данными с сервером.

**Цифровая подпись** - это криптографический механизм проверки подлинности сообщения или документа. Она основана на использовании пары ключей: закрытого и открытого. Когда документ подписывается, то вычисляется криптографический хэш сообщения и подписывается закрытым ключом. Полученная цифровая подпись помещается вместе с самим сообщением и отсылается получателю. Получатель, используя открытый ключ, расшифровывает подпись и сверяет полученное значение с хэшем сообщения. Если значения совпадают, то документ или сообщение считается подлинным и неподделываемым.

Цифровая подпись обычно используется для проверки подлинности электронных документов, электронной почты, транзакции в интернете, программного обеспечения и других электронных ресурсов. Это может быть полезно для обеспечения безопасности и защиты от возможных атак, взломов или попыток фальсификации данных. Кроме того, цифровая подпись может использоваться для установления авторства и прав на интеллектуальную собственность.

**Программа**

Избиратель

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Подключение к серверу на локальном хосте на порту 5000. Затем создает приватный ключ с помощью команды OpenSSL. Пользователь вводит свой голос в файл message.txt. Затем голос шифруется с использованием открытого ключа publickey.pem с помощью команды OpenSSL. Зашфированный голос отправляется на сервер с помощью функции send(). В конце соединение с сервером закрывается с помощью close().

В данной части программы создается сокет, который прослушивает порт 5001 на локальном хосте. После настройки, сокет принимает новые подключения, ждет данные и получает зашифрованный ответ от пользователя, который передает с помощью функции recv(). Затем ответ выводится на экран. После этого, соединение с регистратором закрывается с помощью close().

Затем программа соединяется с портом 5002 и отправляет полученные данные в счетчик. После отправки данных, соединение закрывается с помощью close().

Регистратор

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Создается новый сокет и настраивает его на прослушивание порта 5000 на локальном хосте. С помощью команды OpenSSL создается приватный ключ.

После создания сокет прослушивает новые подключения, ждет данные и получает зашифрованный ответ от пользователя, который передает с помощью функции recv().

Далее зашифрованный голос расшифровывается с использованием приватного ключа privatekey.pem при помощи команды OpenSSL. Затем вычисляется цифровая подпись файла message.dec также при помощи команды OpenSSL и сохраняется в файл signature.bin.

После вычисления цифровой подписи, программа соединяется с портом 5001 и отправляет подтверждение о получении цифровой подписи "Signature received" на следующий сервер (счетчик) с помощью функции send().

В конце программы соединение закрывается с помощью close().

Счетчик

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Создается новый сокет и настраивает его на прослушивание порта 5002 на локальном хосте. После настройки, сокет принимает новые подключения, ждет данные и получает данные от прошлого сервера (регистратора). Данные получаются с помощью функции recv().

Затем с помощью OpenSSL проводится верификация цифровой подписи файла message.txt. Результат выполнения команды сохраняется в переменной result. Если код возврата команды равен 0, то программа выводит сообщение "verification passed", читает содержимое файла message.txt и выводит его на экран вместе с IP-адресом (переменная address) прошлого сервера. Если код возврата не равен 0, то программа выводит сообщение "verification failed".

В конце программы соединение закрывается с помощью close().

**Результат работы программы**

Избиратель

**Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание**

Регистратор

**Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание**

Счетчик

**Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание**

1. Регистратор создает секретный ключ
2. Избиратель создает публичный ключ
3. Избиратель вводит ответ
4. Ответ сохраняется в message.txt
5. Ответ шифруется
6. Регистратор расшифровывает ответ
7. Подписывает его
8. Уведомляет об этом пользователя
9. Пользователь сообщает счетчику
10. Счетчик верифицирует подпись
11. Если верификация пройдена, данные пользователя и его ответ записываются