Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-01-00283

The reported study was funded by RFBR, project number 20-01-00283

Литература

1. Сарапулов Ф.Н., Фризен В.Э., Швыдкий ЕЛ., Смольянов ИА. Математическое моделирование линейного асинхронного двигателя на основе детализированных схем замещения .// Электротехника. 2018. №4.С.58-63
2. Структурное моделирование электротехнологических систем и механизмов / В.А. Иванушкин, Ф.Н. Сарапулов, В.Н. Кожеуров, Д.В.Исаков. Н-Тагил:НТИ(ф) УГТУ-УПИ, 2007.—393 с.
3. Чапаев В.С., Волков С.В. Исследования линейного асинхронного двигателя с управляющим слоем // Надежность и качество. : Труды международного симпозиума: в 2-х т. / Под ред.Н.К. Юркова. – Пенза: ИИЦ ПензГУ, 2008. – 2 т. – С. 135-136.
4. Численные методы анализа [Текст] : приближение функций, дифференциальные и интегральные уравнения / Б. П. Демидович, И. А. Марон, Э. З. Шувалова ; под ред. Б. П. Демидовича. - Изд. 5-е стер. - Санкт-Петербург и др. : Лань, 2010. - 400 с.

Е.А. Зайченко, ст. преп.; А.А. Барыгин, студент; А.С. Плотников, студент.
(Белорусско-Российский университет, г. Могилев, Беларусь)

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАЩИТЫ РАСПРЕДЕЛЕННОГО ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ СОКЕТОВ PYTHON

В настоящее время актуальной является задача о распределении нагрузки по выполнению вычислений на большое количество отдельных звеньев сети и компонентов компьютеров. В качестве примера распределенного взаимодействия разработано приложение, выполняющее обмен сообщениями между множеством пользователей (чат). Сокеты представляют собой классическое средство межпроцессного взаимодействия IPC (Inter-process communication) и дают возможность обмена информацией между задачами, выполняющимися на различных компьютерах сети, причем независимо от операционных систем. Сокеты можно настроить как для работы в качестве сервера и прослушивания входящих сообщений, так и для подключения к другим приложениям в качестве клиента.

Разработанное распределенное приложение [1] включает в себя следующий функционал:

- регистрация новых пользователей;
- аутентификация и авторизация пользователей;
- создание нового чата;
- присоединение к существующему чату и дальнейшее его использование.

Разработанное распределенное приложение состоит из серверной и клиентской части.

После запуска серверной части распределенного приложения, будет создан сокет для сервера на определенном порте и сервер перейдет в режим ожидания запросов. Сервер может обрабатывать запросы на регистрацию и авторизацию пользователя, запрос на создание нового чата, запрос на присоединение к существующему чату. Когда на сервер поступит один из вышеперечисленных запросов, сервер инициализирует выполнение функции, отвечающей за обработку поступившего запроса. Следует отметить, что для работы каждого

чата создается собственный сокет, а для каждого подключения к сокету – отдельный поток (thread).

Для построения графического интерфейса клиентской части распределенного приложения модуль Tkhinter. После запуска клиентского приложения будет отображено окно авторизации/регистрации пользователей. Далее будет отправлен запрос на сервер, а сервер в свою очередь вернет ответ на клиентское приложение. В случае если сервер по каким-либо причинам недоступен, будет отображено окно с соответствующим сообщением. В случае успеха будет отображено окно подключения к чату. Пользователю будет предложено ввести имя чата и пароль доступа к этому чату. Кроме того, пользователь может инициировать создание нового чата.

После отображения окна самого чата, в котором пользователь уже может отправлять и принимать сообщения, на клиентском приложении запускается новый поток, у которого создается собственный сокет. Этот сокет отвечает за принятие сообщений от потока чата, который запущен на серверной части распределенного приложения. Сообщения, получаемые этим потоком от серверного потока данного чата, отображаются в основном окне чата на клиентском приложении.

Интерфейс разработанного приложения минималистичен (рисунок 1), что отражает его функциональность. Основной задачей является обеспечение надежной и быстрой передачи данных, масштабируемость, производительность и защита данных от несанкционированного доступа.

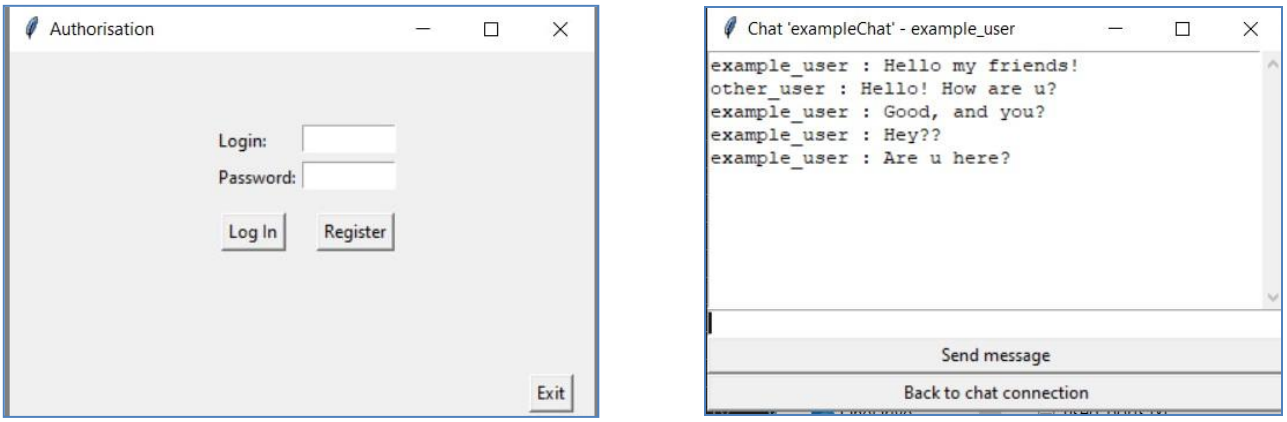

Рисунок 1 – Форма авторизации и вид чата на стороне клиента

В приложении реализован механизм тайм-аута: если все пользователи, подключенные к данному чату, не будут отправлять сообщения в течении некоторого времени, чат будет закрыт, все подключенные к этому чату пользователи будут отключены от него и возвращены к окну подключения к чату.

Особое внимание уделено проблеме безопасности. Передаваемая информация должна быть надежно защищена от несанкционированного доступа. Сведения о клиентах чата должны быть недоступны.

В разработанном распределенном приложении реализовано шифрование паролей пользователей и чатов, а также всех отправляемых сообщений.

Выбрано следующее решение: шифрование и дешифрование производится на клиентской части распределенного приложения и на сервер вся эта информация приходит уже в зашифрованном виде, что предотвращает всевозможные способы перехвата сообщений и паролей, пока они находятся на пути к серверу. База данных о пользователях и чатах хранится в серверной части приложения и реализована с помощью PostgreSQL. Все данные по идентификации пользователей хранятся на сервере в зашифрованном виде, что обеспечивает их защиту.

Разработаны два алгоритма шифрования данных, реализованные в виде библиотек Python.

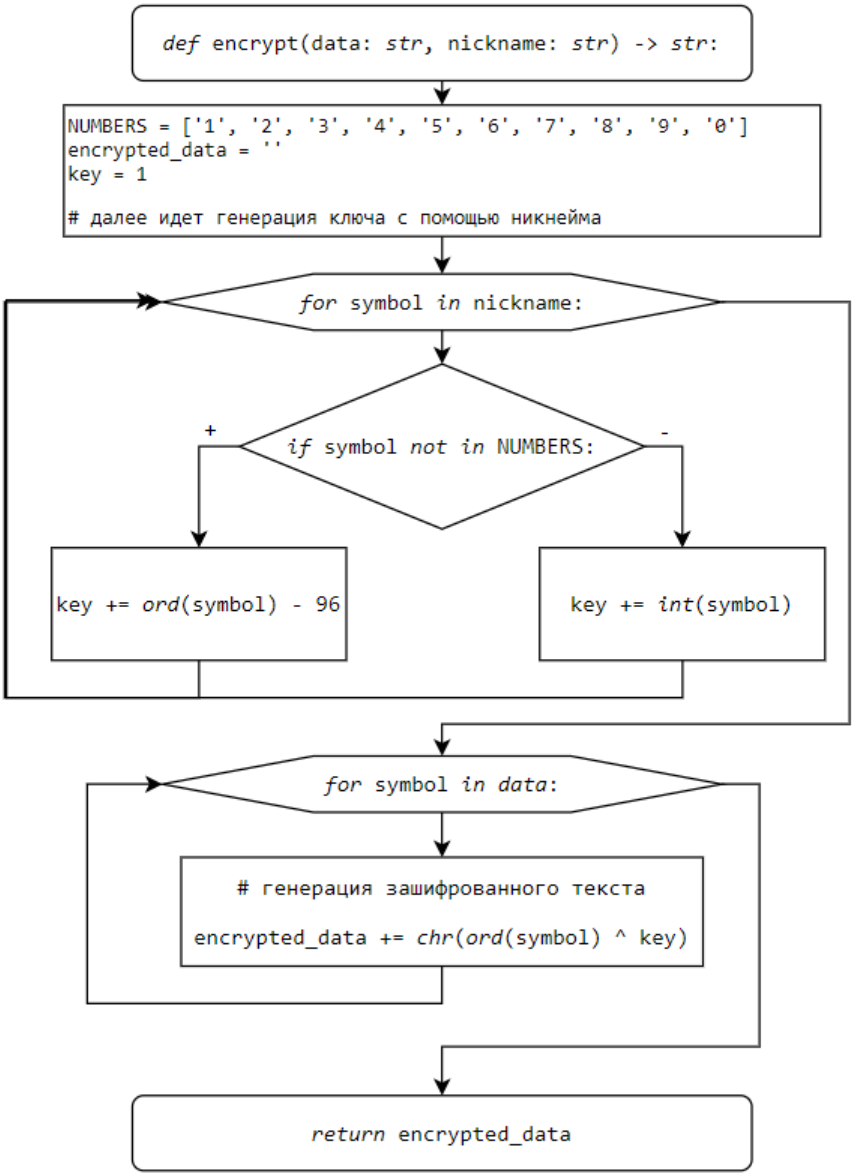

Рисунок 2 – Схема алгоритма шифрования Encript

Первый алгоритм (Encript) использует Pickle, стандартный модуль Python, преобразующий сложные объекты в поток байтов, которые можно передавать по сети. Затем в полученном потоке байт реализована операция побитового сдвига и поэтапное преобразование, причем каждый пароль и каждое сообщение шифруется уникальным образом, при помощи логина пользователя, который отправляет эти сообщения. Применение уникального ключа шифрования для каждого сообщения и пароля существенно усложняет расшифровку этих данных. Схема алгоритма шифрования Encript приведена на рисунке 2.

Второй алгоритм (Encode) использует статичный ключ шифрования, но сам принцип шифрования текста более сложен, что делает этот метод более надежным в плане защиты от попытки взлома. Схема алгоритма шифрования Encode приведена на рисунке 3.

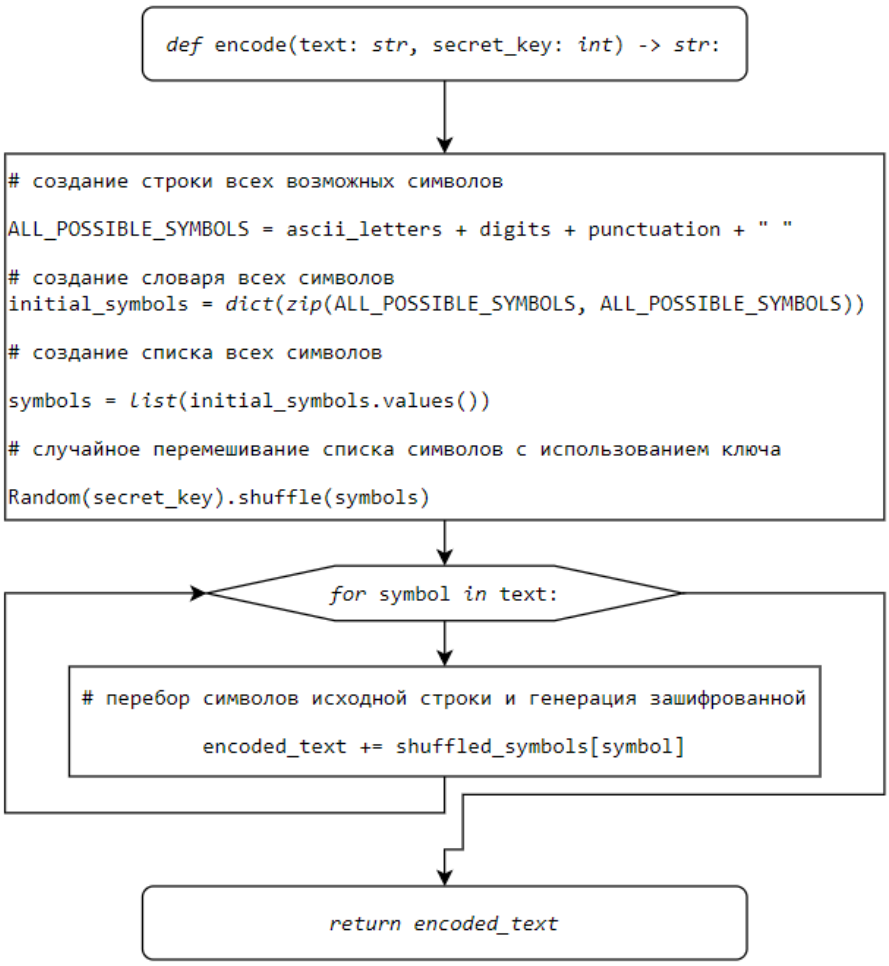

Рисунок 3 – Схема алгоритма шифрования Encode

Оба разработанных алгоритма шифрования протестированы. Для проверки и сравнительного анализа защиты от взлома разработана программа, осуществляющая взлом паролей. В программе реализован метод брутфорс — метод угадывания пароля (или ключа, используемого для шифрования), предполагающий систематический перебор всех возможных комбинаций символов до тех пор, пока не будет найдена правильная комбинация. Как

известно, при применении данного метода можно взломать любой пароль, но количество времени на взлом увеличивается экспоненциально количеству символов [2]. Поэтому рекомендуется использовать в паролях не менее 6 символов. На диаграмме (рисунок 4) показано среднее время, требуемое на подбор пароля для разного количества символов. На основании проведенных тестов можно сделать вывод, что второй алгоритм требует больше времени на подбор пароля, и, следовательно, обеспечивает более надежную защиту данных от взлома.

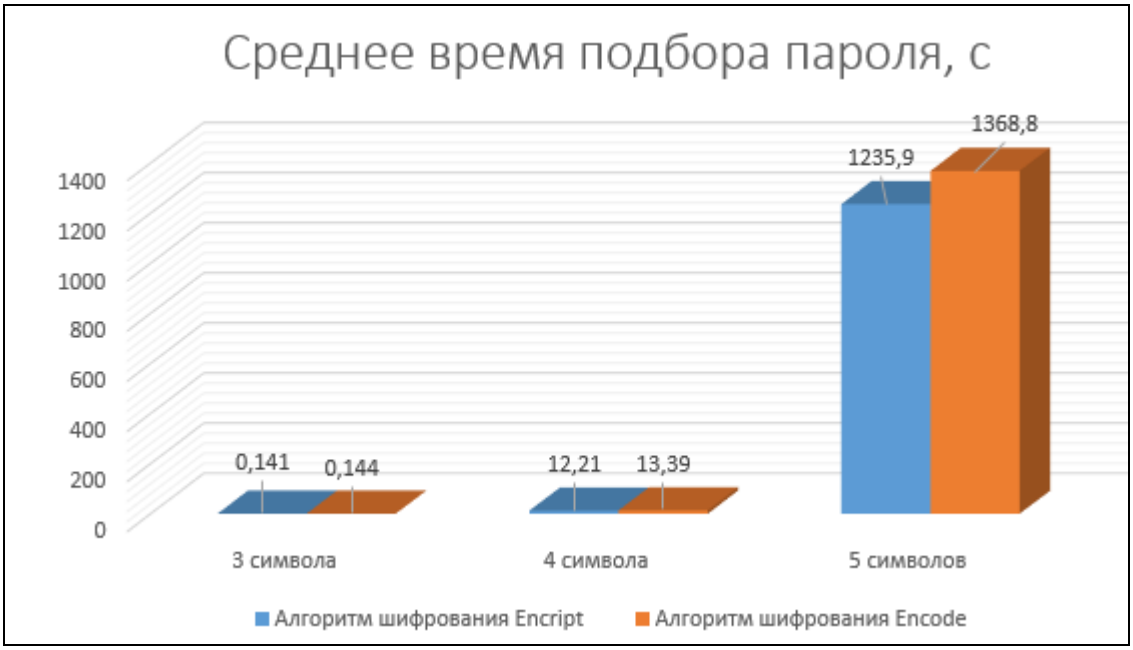

Рисунок 4 – Сравнение времени подбора пароля

Предложенные алгоритмы шифрования при обмене сообщениями показали высокую устойчивость к ошибкам и удовлетворительную надежность, и могут быть использованы для защиты данных в распределенном приложениях, например, в веб-чатах, чатах групп студентов или сотрудников предприятий.

Литература

1. Разработка интернет-чата с использованием сокетов Python. // А. С. Барыгин, А. С. Плотников (Научный рук. Зайченко Е. А.) // 56 Студенческая научно-техническая конференция. – Могилев: Белорусско-Российский университет. – 2020. – С. 25. – [Электронный ресурс]. – Режим доступа: URI: http://e.biblio.bru.by/handle/1212121212/12993– Дата доступа: 07.09.2021.
2. Жук, А.П Защита информации: учеб. пособие / А.П. Жук и др. - 3-е изд., - Москва: РИОР: ИНФРА-М, 2021. – 400 с — [Электронный ресурс]. — Режим доступа. — URL: https://znanium.com/catalog/product/1210523 (дата обращения: 09.09.2021)