Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Приложение для обмена сообщениями с использованием метода сквозного шифрования»

Студент _____ИУ7-73Б_____ _____ _____А. А. Наместник_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Студент _____ИУ7-73Б_____ _____ _____А. Сукочева_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____Н.О. Рогозин_____
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине «Компьютерные сети»

Студенты группы ИУ7-73Б

Наместник Анастасия Андреевна
(Фамилия, имя, отчество)
Сукочева Алис
(Фамилия, имя, отчество)

Тема курсового проекта Приложение для обмена сообщениями с использованием метода сквозного шифрования

Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к _4_ нед., 50% к _7_ нед., 75% к _11_ нед., 100% к _14_ нед.

Задание: Реализовать приложение для обмена сообщениями по не защищенному от прослушивания каналу связи использованием метода сквозного шифрования

Оформление курсовой работы:

Расчетно-пояснительная записка на _25-30_ листах формата А4.
Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы и приложения.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть предоставлена презетация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс, результаты проведенных исследований.

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ Н. О. Рогозин
(Подпись, дата) (И.О.Фамилия)
Студент _____ А. А. Наместник
(Подпись, дата) (И.О.Фамилия)
Студент _____ А. Сукочева
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В современном мире в условиях роста использования информационных технологий для обмена информацией одним из важнейших требований к таким операциям является безопасность данных. Большинство данных передаются по сети, и всегда существует опасность того, что злоумышленник перехватит пакет. Главная задача при осуществлении обмена данными – не позволить получить содержимое пакета в том виде, в котором данные были отправлены. Также для многих пользователей является важным сохранение конфиденциальности данных при передачи их через сервер.

Цель данной работы – реализовать приложение для обмена сообщениями по не защищенному от прослушивания каналу связи с использованием метода сквозного шифрования.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи.

1. Проанализировать существующие решения.
2. Описать метод решение поставленной задачи.
3. Описать требования к системе.
4. Разработать систему для решения поставленной задачи.

1. Аналитическая часть

В данном разделе будет поставлена задача, определено понятие шифрования, проанализированы способы шифрования информации, находящейся в процессе передачи, рассмотрен протокол Диффи-Хеллмана для создания общего ключа шифрования, а также рассмотрены симметричные алгоритмы шифрования.

1.1 Постановка задачи

В рамках поставленной цели необходимо реализовать метод сквозного шифрования при передаче данных по незащищенному каналу в сети с демонстрацией смоделированной ситуации общения двух устройства в интерфейсе командной строки.

1.2 Понятие шифрования

Шифрование - это процесс преобразования открытого текста в нечитаемый вид, чтобы защитить информацию, с применением математических алгоритмов и уникального набора бит, называемого ключом, к которому они применяется наряду с текстом [10].

Дешифрование - это обратный шифрованию процесс с целью получить информацию в первоначальном виде [10].

На рисунке 1.1 представлена концептуальная схема процесса преобразования открытого текста в закодированный и обратно.

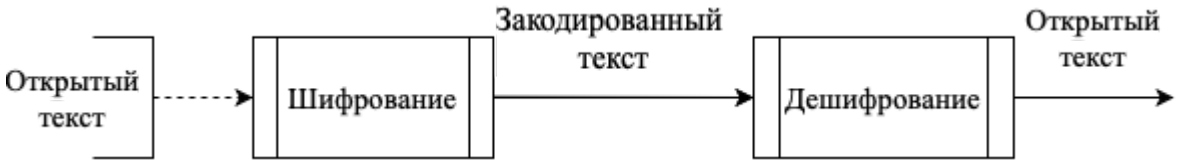

Рис. 1.1 - Двунаправленный процесс преобразования открытого текста

Шифрование позволяет обеспечить такие критерии безопасности информации, как [15]:

1) конфиденциальность;
2) целостность;
3) идентифицируемость.

1.3 Способы шифрования информации в процессе передачи

При передаче сообщения от одного устройства другому используется определенный механизм коммуникации и промежуточный сервер, обрабатывающий транспортировку данных. При этом данные могут передаваться как в зашифрованном, так и в открытом виде [15], что является самым небезопасным способом, так как в случае компрометации канала связи посредством такой атаки, как, например, MITM-атака [2], поток может быть ретранслирован и, в таком случае, все данные окажутся доступны злоумышленнику для изменения или использования в своих целях. Отсутствие криптографической защиты при условии использования в чистом виде наблюдается у некоторых протоколов прикладного уровня, например, Telnet, FTP, IMAP, HTTP, версии 1 и 2c SNMP и других [11].

В настоящее время все больше протоколов, обеспечивающих обмен данными по сети, включают методы защиты, такие как шифрование. Существует два основных способа криптографической защиты при передаче сообщений через промежуточный сервер.

1.3.1 Шифрование транспортного уровня

Транспортное шифрование предполагает наличие общего ключа отправителя и сервера, который используется для дешифрации сообщения на сервере и недоступен другим клиентам [6].

На рисунке 1.2 представлена концептуальная схема шифрования транспортного уровня, где Alice и Bob - клиенты, обменивающиеся данными в

сети через промежуточный сервер, а KAS и KBS - секретные ключи Alice и Bob’а, соответственно.

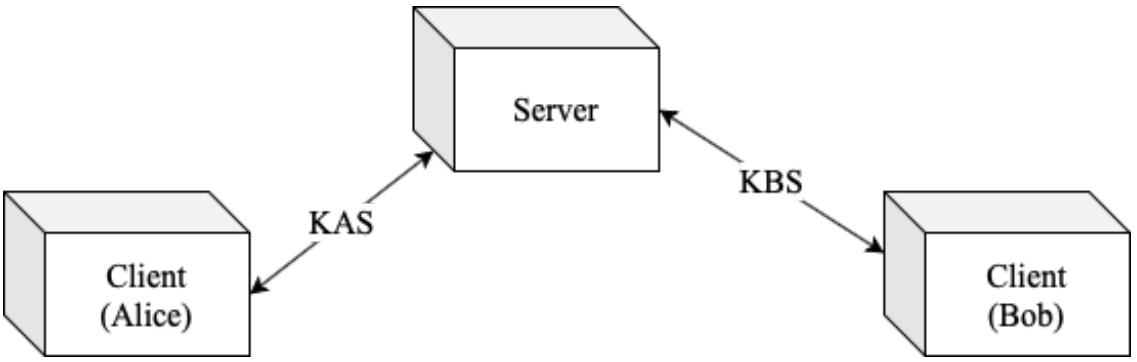

Рис. 1.2 - Концептуальная схема шифрования транспортного уровня

Сторонам, установившим соединение, нет необходимости передавать друг другу ключи, так как дешифрация сообщений происходит в промежуточном звене - на сервере. Процесс передачи данных сводится к следующей последовательности действий.

1. Alice (Bob) зашифровывает сообщение ключом KAS (KBS) и отправляет на сервер в зашифрованном виде.
2. Сервер расшифровывает сообщение ключом KAS (KBS).
3. Сервер зашифровывает сообщение ключом KBS (KAS) и отправляет зашифрованное сообщение Bob’у (Alice).
4. Bob (Alice) расшифровывает сообщение ключом KBS (KAS) и получает данные в исходном виде.

На рисунке 1.3 представлен процесс передачи данных при шифровании транспортного уровня.

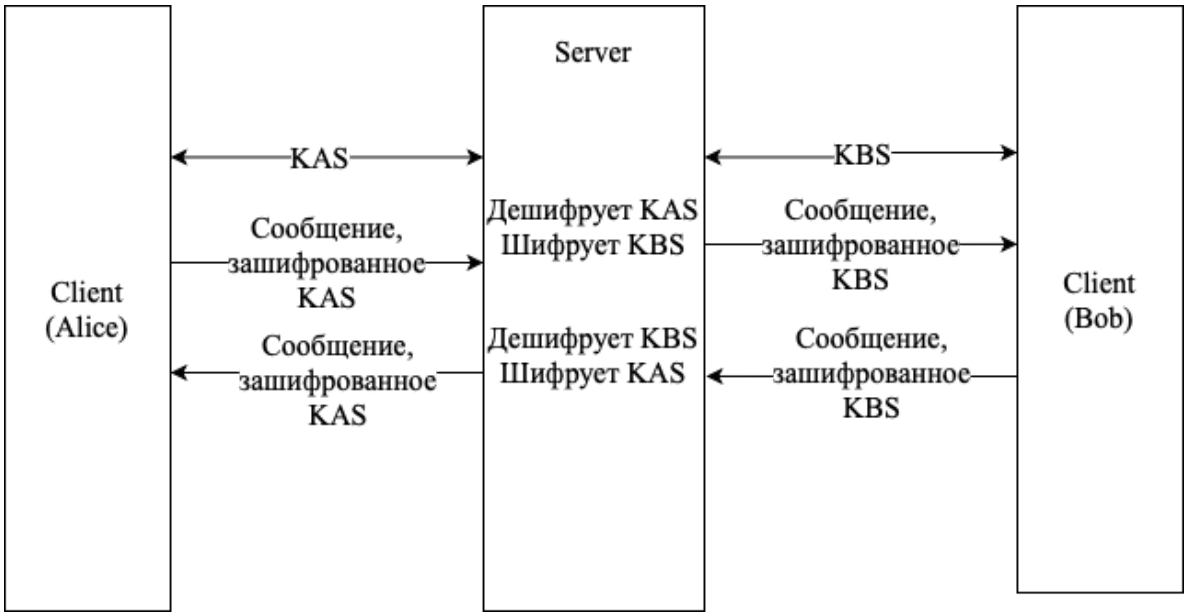

Рис. 1.3 - Процесс передачи данных при шифровании транспортного уровня

Такой подход обеспечивает сохранение конфиденциальности данных при атаках наподобие MITM-атаки, так как теперь пакеты, передающиеся по каналу связи, зашифрованы, однако содержимое сообщения доступно промежуточному серверу. Шифрование транспортного уровня поддерживается таким протоколом, как HTTPS, так как этот протокол использует шифрование, но сервера сайтов, которые его используют, имеют доступ к данным, которые пользователи вводят на этих сайтах [6]. Другим примером такого шифрования является виртуальная частная сеть (VPN). При использовании VPN трафик использует соединение с интернет-провайдером, но при этом трафик будет зашифрован на всем пути до поставщика услуги VPN. При этом злоумышленник увидит лишь то, что было установлено соединение с VPN, а интернет-провайдер может определить, каким именно ресурсом VPN пользуется пользователь. Несмотря на то, что VPN делает трафик недоступным для интернет-провайдера, эта информация будет в полной мере доступна самому поставщику VPN [6].

Достоинствами дешифрации данных на сервере являются возможности использовать дополнительную обработку (например, модерация сообщений),

долгосрочно хранить историю сообщений, контролировать данные, представляющие угрозу общественной безопасности, чтобы оперативно обратиться к правоохранительным органам, и другое. К недостаткам относится угроза конфиденциальности данных, в случае если сервер окажется ненадежным или произойдет взлом сервера в результате атаки [6].

1.3.2 Сквозное шифрование

Сквозное шифрование - это способ шифрования информации при передаче, при котором дешифрация сообщения осуществляется только участниками общения, что обеспечивает недоступность данных в исходном виде как для злоумышленника, перехватывающего пакеты, так и для промежуточного сервера [1]. На рисунке 1.4 представлена концептуальная схема сквозного шифрования, где s - секретный ключ, разделяемый Alice и Bob'ом, E - функция шифрования, m - данные в открытом тексте.

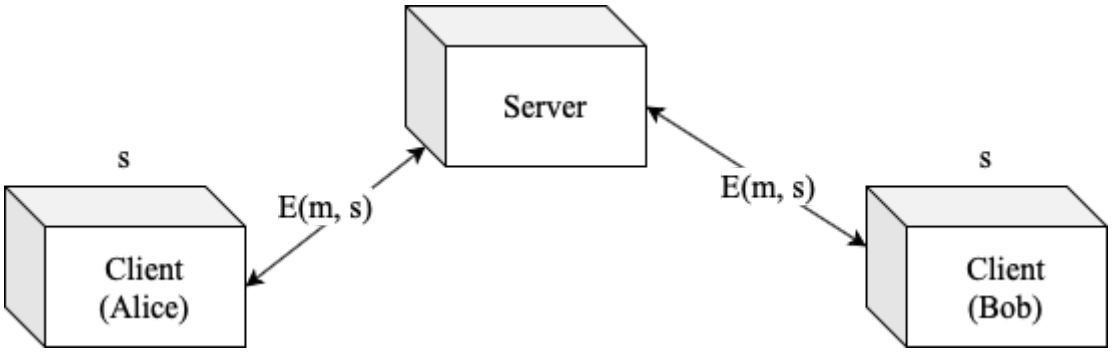

Рис. 1.4 - Концептуальная схема сквозного шифрования

Для обмена ключами могут быть применены симметричный и асимметричный алгоритмы. Для реализации условия дешифрации данных может быть использована схема с предварительным разделением секрета или, например, протокол Диффи-Хеллмана, который используется в мессенджерах WhatsApp [7] и Telegram [4].

1.4 Протокол Диффи-Хеллмана

Протокол Диффи - Хеллмана - криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования [12]. Работа протокола сводится к выполнению следующей последовательности действий каждой стороной:

1. генерирование случайного натурального числа a — закрытый ключ;
2. совместно с удалённой стороной установление открытых параметров p и g (обычно значения p и g генерируются на одной стороне и передаются другой), где p является случайным простым числом;
3. вычисление открытого ключа A, используя преобразование над закрытым ключом: $A = g^a \bmod p$;
4. обмен открытыми ключами с удаленной стороной;
5. вычисление общего секретного ключа K, используя открытый ключ удаленной стороны B и свой закрытый ключ a: $K = B^a \bmod p$

K получается равным с обеих сторон, потому что:

$B^a \bmod p = (g^b \bmod p)^a \bmod p = g^{ab} \bmod p = (g^a \bmod p)^b \bmod p = A^b \bmod p$.

На рисунке 1.5 представлена концептуальная схема обмена ключами по протоколу Диффи-Хеллмана.

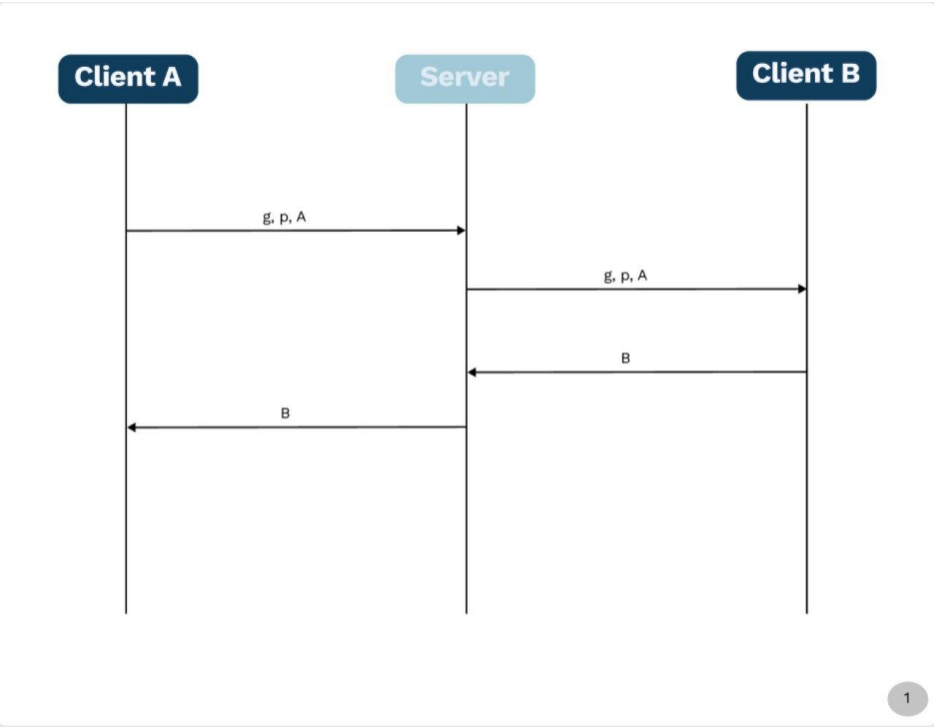

Рис. 1.5 - Концептуальная схема обмена ключами по протоколу Диффи-Хеллмана

На рисунке 1.5 представлена схема реализации обмена ключами по протоколу Диффи-Хеллмана.

1.5 Симметричное шифрование

Симметричное шифрование - это способ шифрования, в котором для шифрования и дешифрования применяется один и тот же криптографический ключ. Алгоритм шифрования выбирается сторонами до начала обмена сообщениями [13].

Процесс шифрования в симметричных криптосистемах можно представить следующим образом:

$$E_k(P) = C$$
$$D_k(C) = P,$$

где E – функция зашифрования, k – ключ, P – открытый текст, C – шифртекст, D – функция расшифрования.

При этом справедливо следующее равенство:

$$D_k(E_k(P)) = P.$$

Симметричные алгоритмы, которые обрабатывают открытый текст побитово (побайтово), называют потоковыми алгоритмами или потоковыми шифрами. Алгоритмы, которые обрабатывают группы битов (блоки) открытого текста, называют блочными алгоритмами или блочными шифрами. В настоящее время используются 128-разрядные блоки, так как длина блока в 64 бита не удовлетворяет современным требованиям эффективности и надежности алгоритмов [13].

1.5.1 Алгоритм DEA

Самым известным и широко распространенным компьютерным алгоритмом шифрования является алгоритм DEA, лежащий в основе DES (Data Encrypt Standard) - стандарта шифрования данных США. Алгоритм DEA был опубликован в 1973 году и в течение почти 20 лет считался криптографически стойким [13].

В процессе шифрования с помощью алгоритма DEA последовательно производятся преобразования (раунды) над 64-битовыми блоками:

$$P, \Phi_1, \Phi_2, ..., \Phi_{16}, P^{-1}, \qquad (1.1)$$

где P – заданная подстановка; $\Phi_i = V_i T$ - преобразование (сеть) Файстеля (H. Feistel), являющееся основой многих симметричных алгоритмов:

$T (L, R) = (R, L)$ – перестановка левой и правой частей;

$V_i = V(L_i, R_i) = (L_i, R_i \oplus F(R_{i-1}, K_i))$;

$L_0R_0; \quad L_i = R_{i-1}, \ R_i = L_{i-1} \oplus F(R_{i-1}, K_i), \ (i = 1,...,16);$

где K_i – ключи, получаемые на основе 56-битового секретного ключа K; F – функция раунда.

Расшифрование производится с помощью преобразований (1.1) на основе ключа K, причем ключи K_i генерируются в обратном порядке.

1.5.2 Алгоритм TDEA

Алгоритм TDEA («тройной» DEA) является одной из составляющих стандарта шифрования данных США 1999 году. В алгоритме TDEA для зашифрования используется три ключа, и трижды применяется алгоритм DEA [13]:

$$C = E_{k3}(D_{k2}(E_{k1}(P)))$$

Расшифрование представляет собой следующее преобразование:

$$P = D_{k1}(E_{k2}(D_{k3}(C)))$$

Длина ключа TDEA оказывается равной 168 бит.

Однако существенным недостатком TDEA является то, что алгоритм оказывается очень медленным в условиях программной реализации. Оригинальный алгоритм DEA был разработан в середине 70-х годов XX в. для реализации в виде микросхемы, а не эффективного программного кода. Алгоритм TDEA, соответственно, оказывается еще более медленным. Кроме того, длина блока (64 бит), используемая в DEA и TDEA, не удовлетворяет современным требованиям эффективности и защищенности, предпочтительнее использование блоков большей длины.

1.5.3 Алгоритм DES

Алгоритм DES - это алгоритм для симметричного шифрования, разработанный фирмой IBM и утверждённый правительством США в 1977 году как официальный стандарт [13].

Процесс шифрования состоит из начальной перестановки, 16 циклов шифрования и конечной перестановки. Исходный текст T (блок 64 бит) преобразуется с помощью начальной перестановки IP. Полученный после начальной перестановки 64-битовый блок IP(T) участвует в 16 циклах преобразования Фейстеля, в которых функция f играет роль шифрования.

Рассмотрим функцию f. Аргументами функции f являются 32-битовый вектор R и 48-битовый ключ k_i, который является результатом преобразования 56-битового исходного ключа шифра k. Для вычисления функции f последовательно используются:

1. функция расширения E;
2. сложение по модулю 2 с ключом k_i;
3. преобразование S, состоящее из 8 преобразований S-блоков S_1,S_2,S_3 , … , S_8;
4. перестановка P.

Недостатком такого алгоритма является его относительно низкая безопасность, так как из-за небольшого числа возможных ключей (всего 2^{56}) появляется возможность их полного перебора на быстродействующей вычислительной технике за реальное время.

1.5.4 Алгоритм AES

Вышеупомянутые проблемы способен решить усовершенствованный стандарт шифрования AES (Advanced Encryption Standard), который стал преемником алгоритма DES. Новый стандарт был принят на основе открытого конкурса, в котором участвовали алгоритмы, предложенные математиками из многих стран мира: США, Канады, Австралии, Бельгии, Германии, Норвегии, Франции, Японии, Южной Кореи, Коста-Рики [13].

AES представляет собой блочный, симметричный алгоритм шифрования с длиной блока 128 бит. Длина ключа может принимать значения 128, 192 или 256 бит (AES-128, AES-192 и AES-256, соответственно). Таким образом, алгоритм защищен от атак методом полного перебора ключей. К достоинствам

алгоритма относятся также высокое быстродействие и умеренные требования к памяти.

1.6 Выводы из аналитического раздела

В данном разделе была поставлена задача, определено понятие шифрования, проанализированы способы шифрования информации, находящейся в процессе передачи. Также был рассмотрен протокол Диффи-Хеллмана для создания общего ключа шифрования в методе сквозного шифрования и симметричные алгоритмы шифрования, которые могут быть реализованы в рамках этого протокола.

2. Конструкторская часть

В данном разделе будет спроектирована система, будут сформулированы требования к системе, а также приведены схемы некоторых методов.

2.1 Проектирование системы

Система состоит из двух компонентов.

1. Сервер - выполняет роль промежуточного звена при обмене сообщениями между клиентами.
2. Клиент - имитирует устройство, обменивающиеся сообщениями с другими устройствами .

Соединение между сервером и клиентом осуществляется по не защищенному от прослушивания каналу связи с помощью протокола транспортного уровня TCP [5].

Обработка соединения с каждым клиентом и дальнейшее его общение с собеседником выполняются асинхронно в отдельном потоке. Для асинхронного получения и отправки сообщений каждым клиентом эти процессы так же выполняются в отдельных потоках.

2.2 Требования к системе

Рассмотрим требования к серверу. Сервер должен предоставлять следующие возможности:

1. подключения n-ого количества клиентов;
2. выбрать клиенту для себя уникальный идентификатор (имя);
3. клиенту ожидать собеседника;
4. клиенту выбрать собеседника по уникальному идентификатору;
5. корректно закрывать соединение при терминировании процесса клиента.

Рассмотрим требования к клиенту. Клиент должен предоставлять следующие возможности:

1. ожидание от пользователя ввода уникального идентификатора;
2. отправка серверу уникального идентификатора пользователя;
3. запрос у пользователя режима работы (ожидание или подключение к собеседнику);
4. отправка серверу идентификатора пользователя для общения;
5. создание общего приватного ключа между двумя собеседниками, пересылая информацию через сервер с использованием протокола Диффи-Хеллмана;
6. ожидание от пользователя ввода сообщения и отправка его на сервер.

2.3 Схемы методов

На рисунках 2.1 - 2.6 представлены схемы работы серверной части системы.

Рис. 2.1 - Начало работы сервера (вход в режим ожидания клиентов)

Рис. 2.2 - Ожидание клиентов по TCP каналу и запуск отдельного потока для каждого

Рис. 2.3 - Работа потока обработки запросов от клиента

Рис. 2.4 - Метод получения уникального идентификатора пользователя (клиента)

Рис. 2.5 - Метод обработки выбора действия клиента

Рис. 2.6 - Метод обработки выбора действия клиента

На рисунках 2.7 - 2.12 представлены схемы работы клиентской части системы.

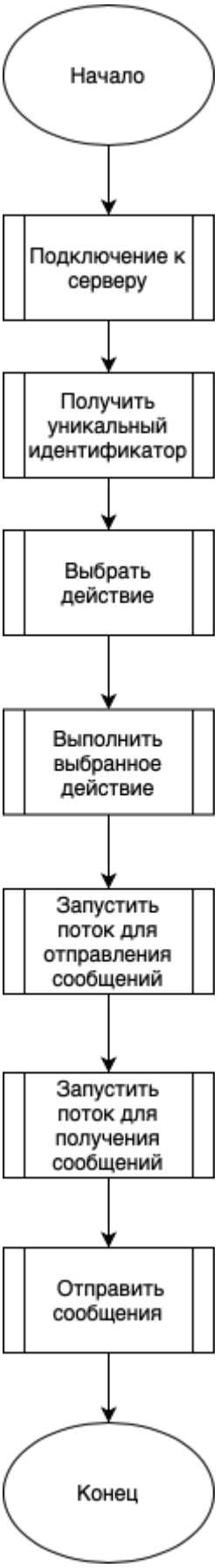

Рис. 2.7 - Методы работы клиента

Рис. 2.8 - Метод получения уникального идентификатора клиента

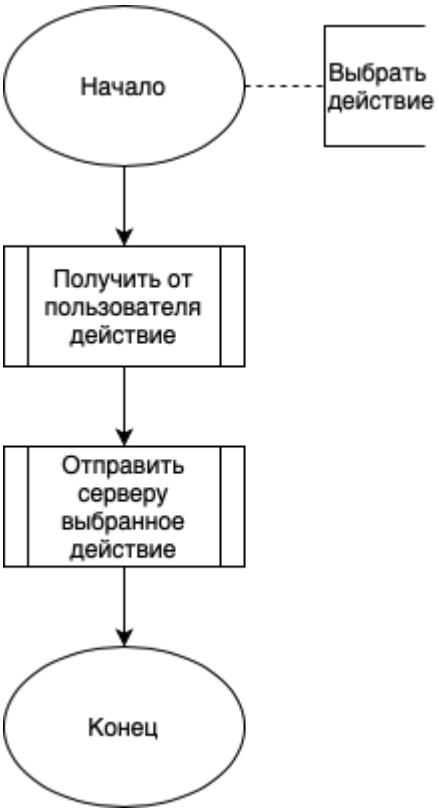

Рис. 2.9 - Метод выбора действия клиента (“ожидание”/”подключение”)

Рис. 2.10 - Реализация протокола Диффи-Хеллмана

Рис. 2.11 - Метод получения сообщения клиентом

Рис. 2.12 - Метод отправления сообщения клиентом

2.4 Выводы из конструкторского раздела

В данном разделе была спроектирована система, были сформулированы требования к системе, а также были приведены схемы некоторых методов.

3. Технологическая часть

В технологической части будут выбраны язык программирования, среда разработки и алгоритм шифрования с указанием причины выбора, приведены структура классов программы и листинги кода, а также продемонстрированы возможности интерфейса.

3.1 Выбор и обоснование используемых технологий

В качестве языка программирования был выбран C# [9]. Данный язык был выбран по следующим причинам.

1. Он использует объектно-ориентированный подход к программированию.
2. В языке присутствует обилие синтаксических возможностей, которые помогают использовать готовые конструкции, вместо того, чтобы переписывать однотипные строки кода.
3. В нем имеется большое количество библиотек и шаблонов, позволяющих не тратить время на изобретение готовых конструкций.
4. Язык является строго типизированным, что позволяет защититься от непроконтролированных ошибок.
5. Он является нативным, что необходимо для асинхронного общения с клиентами.

В качестве среды разработки была выбрана среда Rider [3], так как:

1. Данная среда разработка обладает удобным редактором кода и графическим отладчиком.
2. Rider доступна в бесплатном формате для студентов.
3. Rider поддерживает новейшие версии библиотек C#.
4. Rider – кросс-платформенная среда разработки, что позволяет использовать ее на различных операционных системах.

В качестве алгоритма шифрования данных был выбран алгоритм AES [13] за счет его высокой степени безопасности и высокого быстродействия.

3.2 Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов.

3.2.1 Сервер

На стороне сервера были описаны следующие классы.

- Program - главный класс, содержащий точку входа в программу.
- ClientObject - класс, содержащий информацию о клиенте и предоставляющий методы для взаимодействия с клиентом.
- ServerObject - класс, содержащий информацию о сервере и предоставляющий методы для обработки новых подключений.

На рисунках 3.1-3.2 показан состав классов сервера.

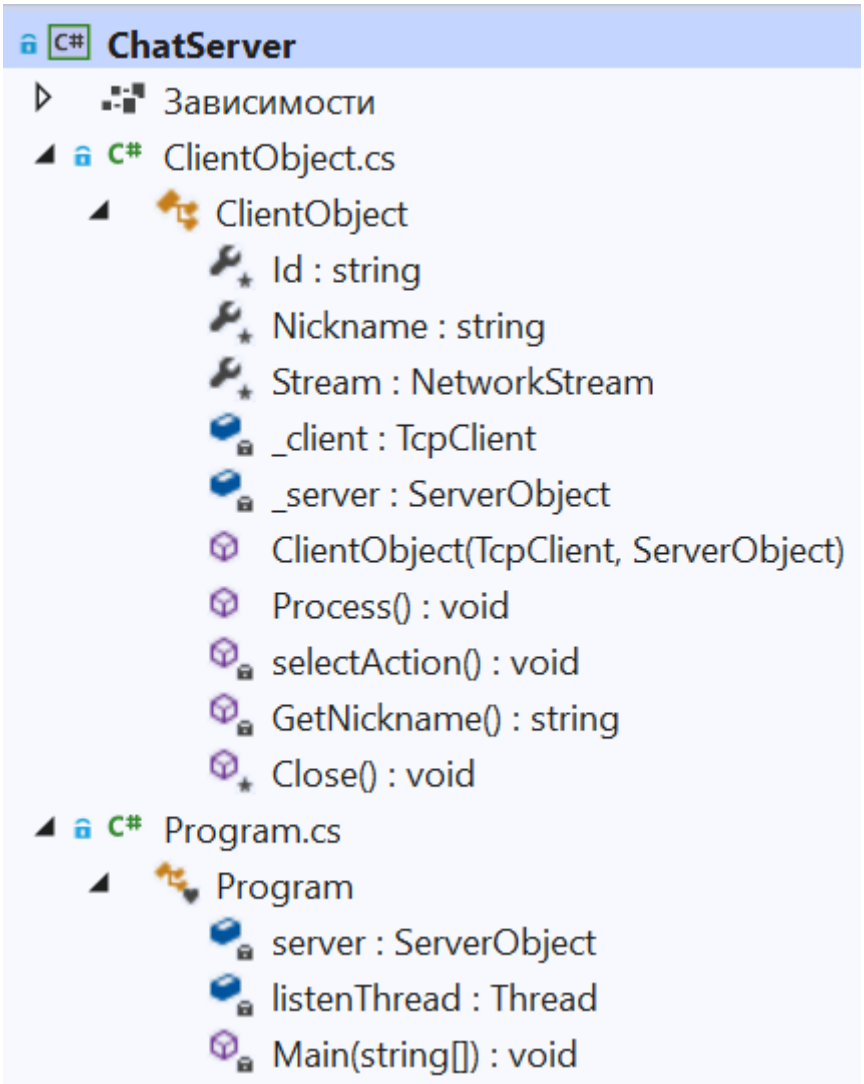

Рис. 3.1 - Состав классов сервера часть 1

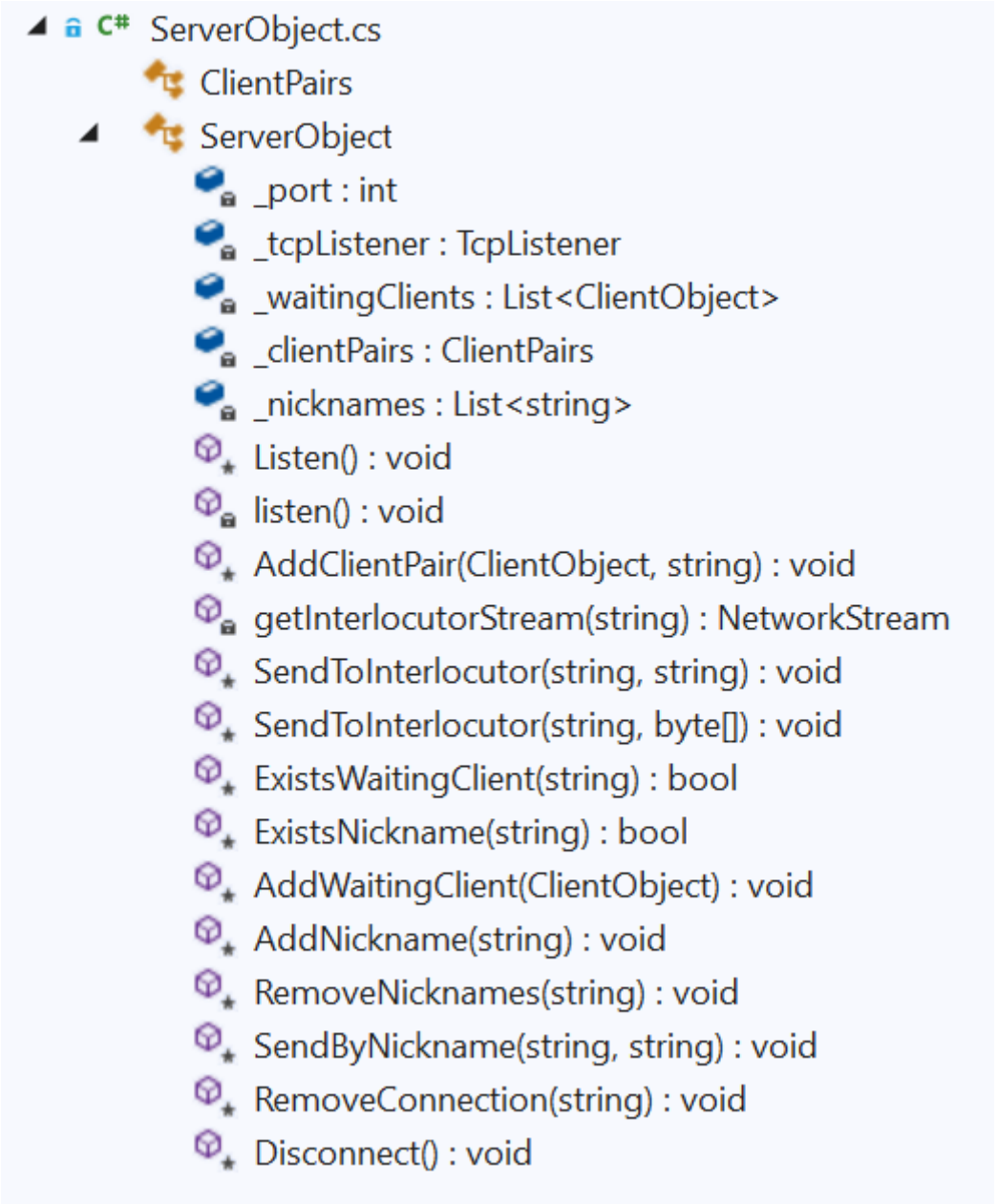

Рис. 3.2 - Состав классов сервера часть 2

На листинге 3.1 продемонстрирован процесс обработки сообщений от клиента.

```
public void Process()
    {
        try
        {
            Stream = _client.GetStream();
            Nickname = GetNickname();
            _server.AddNickname(Nickname);
            selectAction();
```

```
            Console.WriteLine($"{Nickname} connect");
            string message;
            while (true)
            {
                try
                {
                    message = ReadServices.GetMessage(Stream);
                    Console.WriteLine($"{Nickname}: {message}");
                    _server.SendToInterlocutor(Nickname, message);
                }
                catch
                {
                    var msg = $"{Nickname}: disconnect";
                    _server.SendToInterlocutor(Nickname, msg);
                    Console.WriteLine(msg);
                    break;
                }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
        finally
        {
            _server.RemoveConnection(Id);
            _server.RemoveNicknames(Nickname);
            Close();
        }
    }
```

Листинг 3.1 – Процесс обработки сообщений от клиента

3.2.2 Клиент

На стороне клиента были описаны следующие классы.

- Program - главный класс, содержащий точку входа в программу.
- Client - содержит основную логику клиентской части.
- MyMath - содержит дополнительные математические методы.
- DiffieHellman - содержит реализацию протокола Диффи-Хеллмана.
- Encryption - содержит методы для шифрации и дешифрации сообщений.
- Constants - содержит вспомогательные константы.

На рисунках 3.3-3.4 показан состав классов клиента

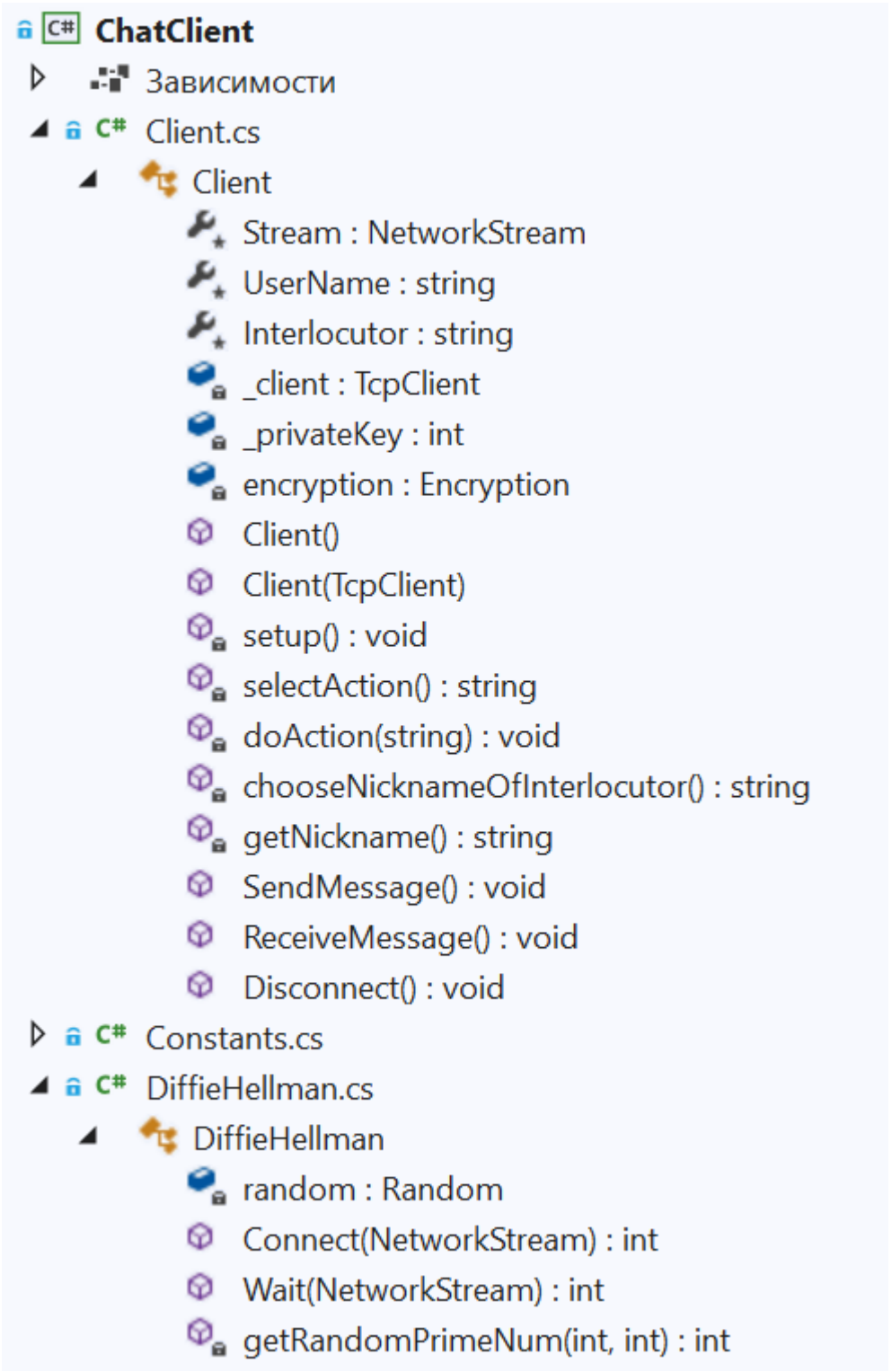

Рис. 3.3 - Состав классов клиента часть 1

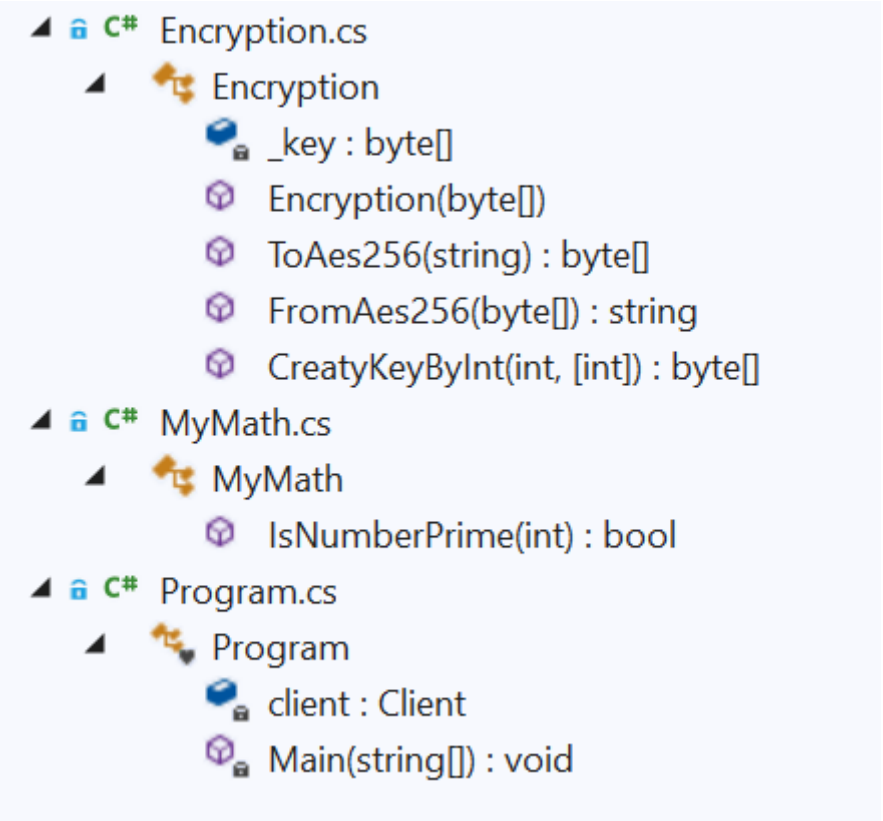

Рис. 3.4 - Состав классов клиента часть 2

На листинге 3.2 продемонстрировано подключения клиента к серверу и начальная настройка.

Листинг 3.2 – Подключения клиента к серверу и начальная настройка.

```
public Client()
 {
     _client = new TcpClient();
     _client.Connect(Constants.Host, Constants.Port);
     setup();
 }
 private void setup()
 {
     Stream = _client.GetStream();
     // Получаем уникальный идентификатор (nickname)
     UserName = getNickname();
     // Выбираем режим: ожидаем подключения или подключаемся к
собеседнику
     var action = selectAction();
     // Создаем общий приватный ключ
     doAction(action);
 }
```

На листинге 3.3 продемонстрировано отправление и получение сообщений на стороне клиента.

Листинг 3.3 – Отправка и получение сообщений на стороне клиента

```
 /// <summary>
/// Отправка сообщений
/// </summary>
public void SendMessage()
{
    Console.WriteLine("Enter the message: ");

    while (true)
    {
        string message = Console.ReadLine();

        var data = encryption.ToAes256(message);
        WriteServices.SendByteArray(Stream, data);
    }
}

/// <summary>
/// Получение сообщений и вывод на экран
/// </summary>
public void ReceiveMessage()
{
    while (true)
    {
        try
        {
            var data = ReadServices.GetByteArray(Stream);
            var message = encryption.FromAes256(data);
            Console.WriteLine($"{Interlocutor}: {message}");
        }
        catch (Exception e)
        {
            Console.WriteLine($"e = {e.Message}");
            Console.WriteLine("Connection interrupted");
            Console.ReadLine();
            Disconnect();
        }
    }
}
```

На листинге 3.4 продемонстрирован класс, реализующий протокол Диффи-Хеллмана.

Листинг 3.4 – Класс, реализующий протокол Диффи-Хеллмана

```
   public static class DiffieHellman
    {
        private static Random random = new Random();

        public static int Connect(NetworkStream stream)
      {
            int g = getRandomPrimeNum(Constants.MinValueG,
Constants.MaxValueG);
            // p is public prime number.
            int p = getRandomPrimeNum(Constants.MinValueP,
Constants.MaxValueP); // Should be simple
            // a is private key for Alice
            int a = random.Next(Constants.MinValuePrivateKey,
Constants.MaxValuePrivateKey);

            // A is public key for Alice
            int A = (int)BigInteger.ModPow(g, a, p);
            var firstPublicData = $"{g} {p} {A}";
            WriteServices.SendString(stream, firstPublicData);

            var B = ReadServices.GetNumber(stream); // This is public key
(for Bob)
            var privateKey = (int)BigInteger.ModPow(B, a, p);

            Console.WriteLine($"privateKey (connect) = {privateKey}");
            return privateKey;
        }

        public static int Wait(NetworkStream stream)
        {
            var tmp = ReadServices.GetMessage(stream);
            var publicData = tmp.Split(" ").Select(x =>
Convert.ToInt32(x)).ToList();

            int g = publicData[0];
            int p = publicData[1];
            int A = publicData[2];

            // b is secret key for Bob
            int b = random.Next(Constants.MinValuePrivateKey,
Constants.MaxValuePrivateKey);
            // B is public key for Bob
            int B = (int)BigInteger.ModPow(g, b, p);
            var PublicKey = B;
            var privateKey = (int)BigInteger.ModPow(A, b, p);
```

```
            WriteServices.SendNumber(stream, B);

            Console.WriteLine($"privateKey (wait)= {privateKey}");
            return privateKey;
        }

        private static int getRandomPrimeNum(int left, int right)
        {
            int num = random.Next(left, right);

            while (!MyMath.IsNumberPrime(num))
                num = random.Next(left, right);

            return num;
        }
    }
```

3.2.3 Сетевой сервис

Также были вынесены общие методы, используемые и сервером, и клиентом в отдельную сборку NetworkServices. В данной сборке описаны следующие классы.

- ReadServices - предоставляет возможность чтение из потока.
- WriteServices - предоставляет возможность записи в поток.

На рисунке 3.5 продемонстрирован состав классов сетевого сервиса.

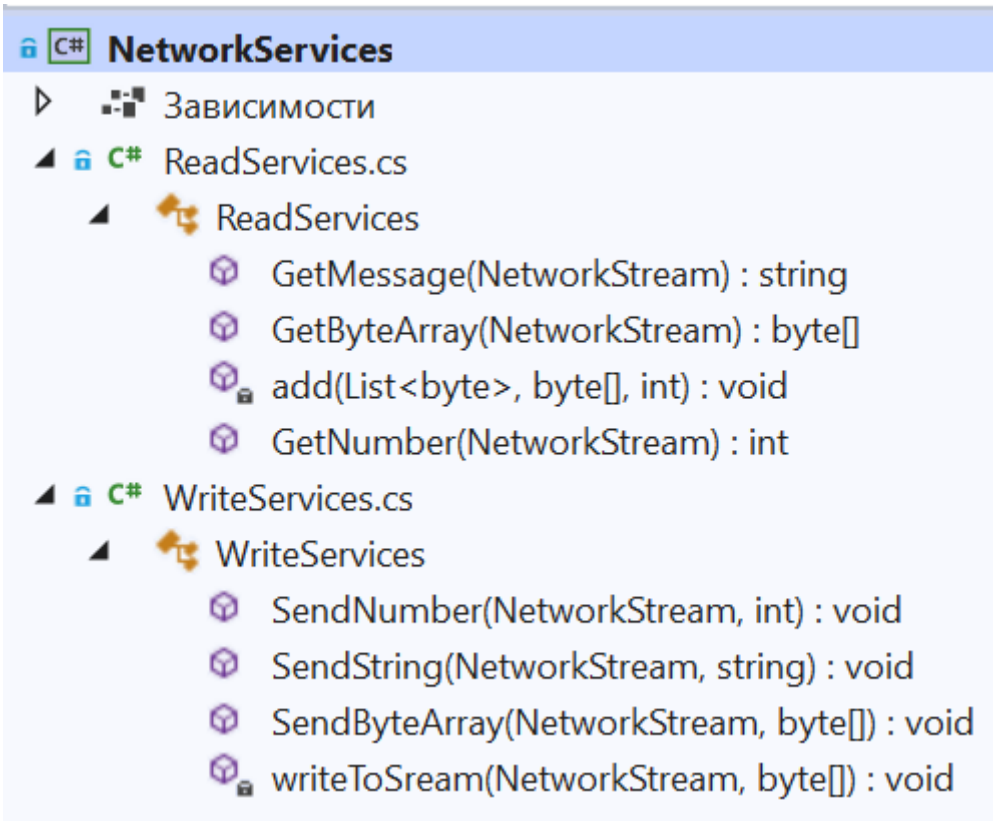

Рис. 3.5 - Состав классов сетевого сервиса

3.3 Интерфейс программы

При запуске пользователь вводит уникальное имя. Если данное имя занято, программа просит ввести его заново, пока что не будет введено уникальное имя, которое еще не используется на сервере рис. .

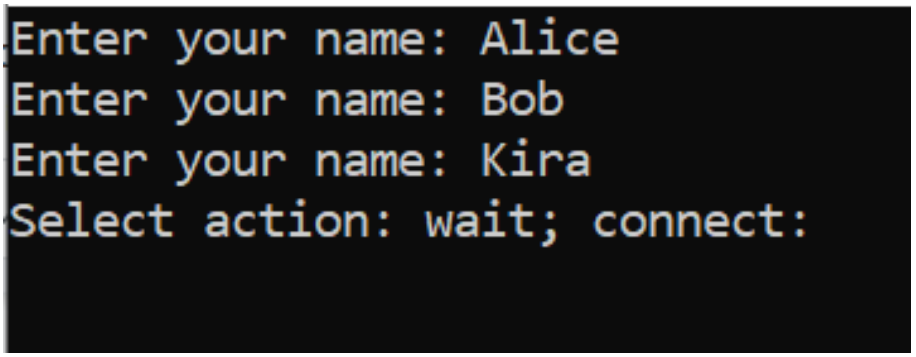

Рис. 3. - Попытка ввода не уникальное имя

Далее пользователь вводит действие, которой он хочет совершить: wait или connect. В зависимости от выбранного действия ему предоставляется

различный дальнейший функционал. При выборе wait пользователь ожидает собеседника рис. . При выборе Connect пользователь вводит имя собеседника рис. . Также показано на рис. , что пользователь не может начать общение с самим собой. Далее происходит общения собеседников.

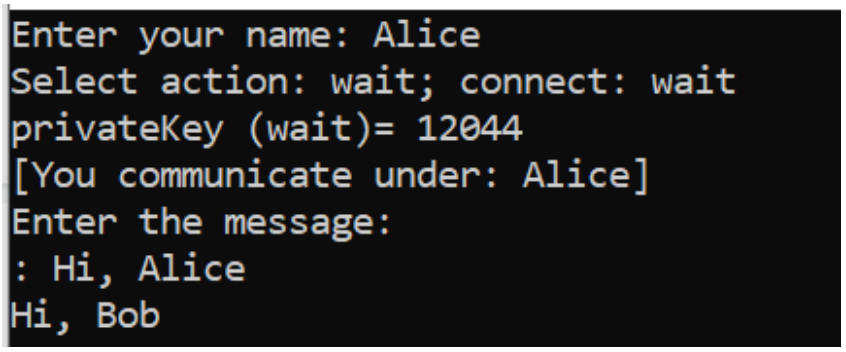

Рис. 3. - Алгоритм действий первого клиента

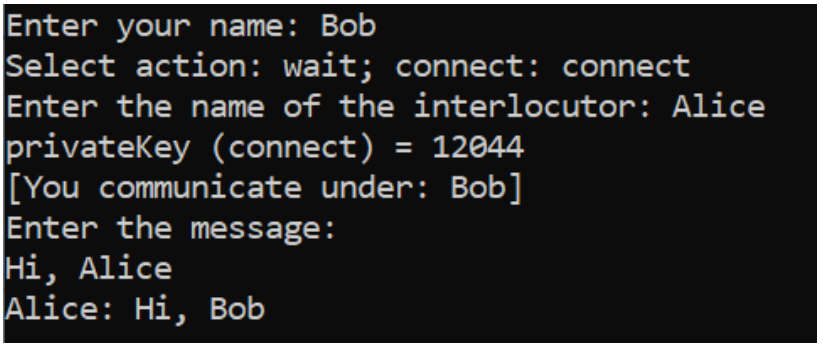

Рис. 3. - Алгоритм действий второго клиента

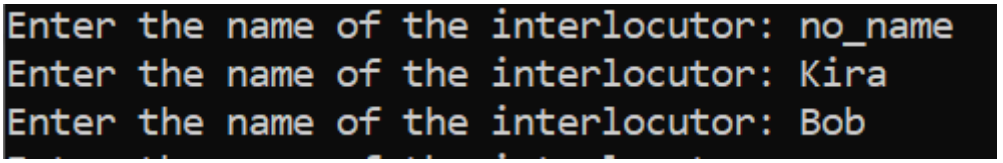

Рис. 3. - Ввод имени собеседника

3.4 Выводы из технологического раздела

В данном разделе был выбран язык программирования C#, среда разработки Rider и симметричный алгоритм шифрования данных AES. Также были описаны структура и состав реализованных классов, представлены листинги кода и продемонстрированы возможности интерфейса.

4. Экспериментальная часть

В данном разделе будет произведен анализ трафика.

4.1 Анализ трафика

Для анализа воспользуемся программой-анализатором трафика wireshark [8].

Запустим сервер и двух клиентов: Алису и Боба. Алиса будет ожидать соединения. Боб подключится к Алисе. После установка соединения Алиса отправит Бобу некоторую информацию. На рисунке 4.1 показан алгоритм действий Алисы. На рисунке 4.2 показан алгоритм действий Боба. На рисунке 4.3 показан вывод информации на сервере.

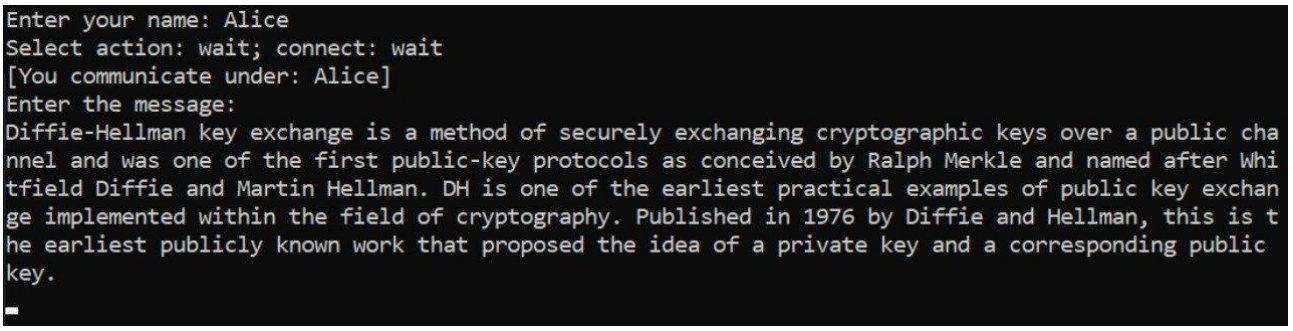

Рис. 4.1 - Алгоритм действий Алисы

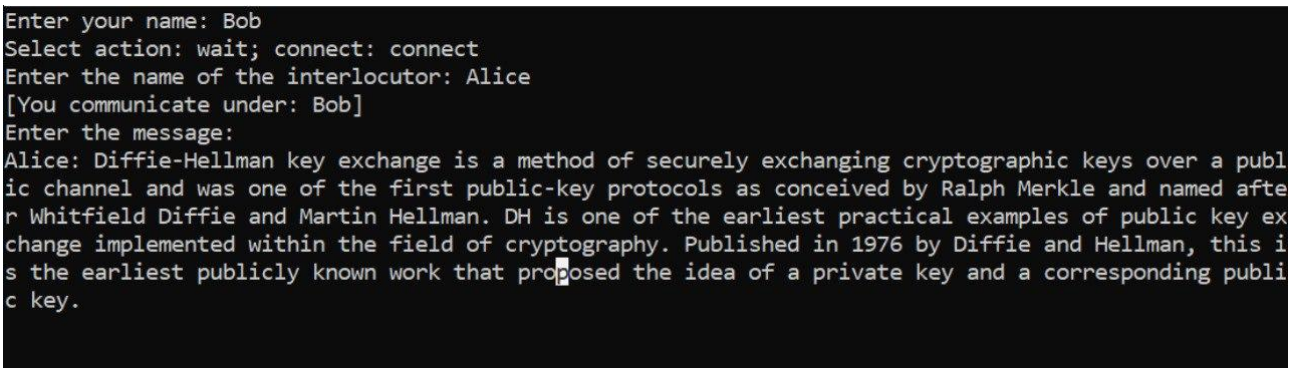

Рис. 4.2 - Алгоритм действий Боба

```
The server is running. Waiting for connections...
Alice connect
Bob connect
Pair is created Alice --- Bob
```

Рис. 4.3 - Вывод информации на сервере

На рисунке 4.4 показаны данные, которые отправила Алиса Бобу. Как можно увидеть, данные передаются в зашифрованном виде, что не позволяет серверу получить искомую информацию, т.к. он не владеет общим ключом Алисы и Боба.

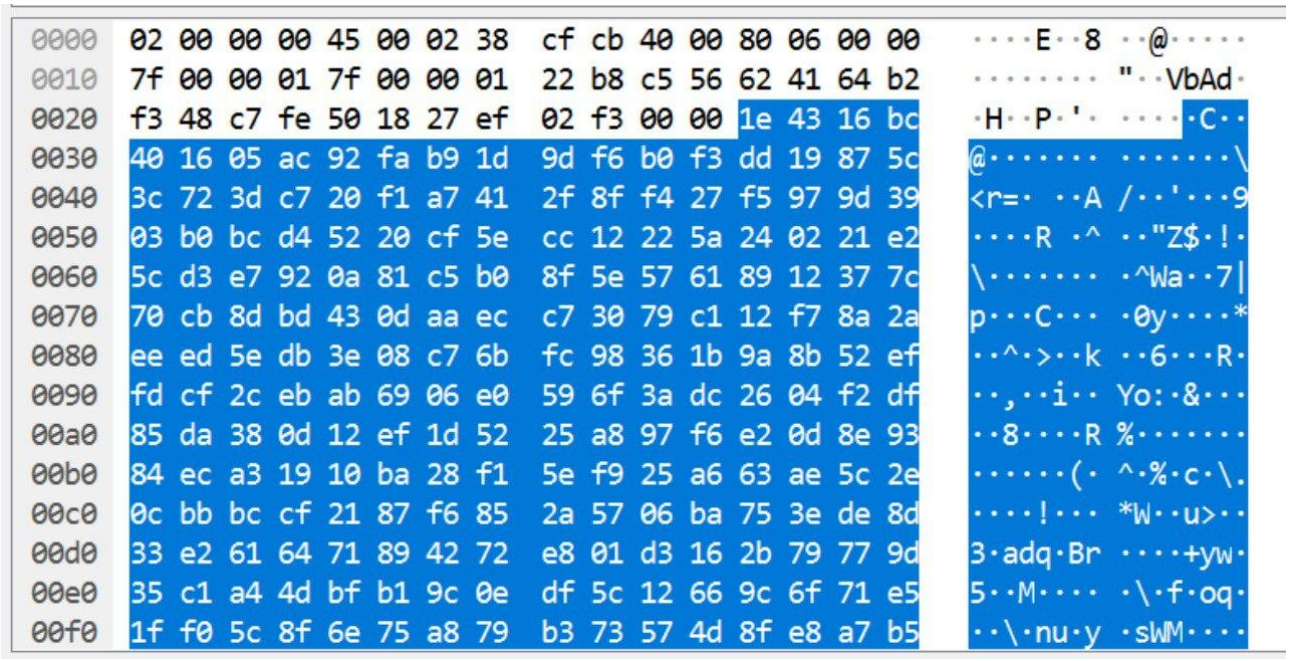

Рис. 4.4 - Анализ трафика

4.2 Выводы из экспериментального раздела

В данном разделе был произведен анализ трафика. Было показано, что сервер не имеет возможности читать сообщения, которыми обмениваются клиенты, потому что он не обладает общим приватным ключом.

Заключение

В ходе выполнения данного проекта были рассмотрены существующие способы шифрования данных, находящихся в процессе передачи, и протокол обмена криптографическими ключами между общающимися сторонами. На основании анализа симметричных алгоритмов шифрования, используемых в этом протоколе, был выбран наиболее подходящий алгоритм. Также была спроектирована система, моделирующая передачу данных с применением сквозного шифрования, которая использует интерфейс командной строки.

Существует перспектива развития работы: реализация общения между тремя и более клиентами с использованием протокола Диффи-Хеллмана для неограниченного количества участников.

Для реализации приложения для обмена сообщениями по не защищенному от прослушивания каналу связи с использованием метода сквозного шифрования были решены следующие задачи.

1. Проанализированы существующие решения.
2. Описан метод решение поставленной задачи.
3. Описаны требования к системе.
4. Разработана система для решения поставленной задачи.

Список использованной литературы

1. End-to-end encryption [Электронный ресурс] Режим доступа: https://ssd.eff.org/en/glossary/end-end-encryption, (дата обращения 01.12.2021)
2. MITM-атака (атака «человек посередине») [Электронный ресурс] Режим доступа: https://encyclopedia.kaspersky.ru/glossary/man-in-the-middle-attack/, (дата обращения 01.12.2021)
3. Rider [Электронный ресурс] Режим доступа: https://www.jetbrains.com/ru-ru/rider/, (дата обращения 05.12.2021)
4. Telegram Web [Электронный ресурс] Режим доступа: https://web.telegram.org/z/, (дата обращения 11.12.2021)
5. Transmission Control Protocol [Электронный ресурс] Режим доступа: https://datatracker.ietf.org/doc/html/rfc793, (дата обращения 11.12.2021)
6. What Should I Know About Encryption? [Электронный ресурс] Режим доступа: https://ssd.eff.org/module/what-should-i-know-about-encryption, (дата обращения 10.12.2021)
7. WhatsApp [Электронный ресурс] Режим доступа: https://www.whatsapp.com/?lang=ru, (дата обращения 11.12.2021)
8. Wireshark [Электронный ресурс] Режим доступа: https://www.wireshark.org, (дата обращения 18.12.2021)
9. Документация по C# [Электронный ресурс] Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/, (дата обращения 05.12.2021)
10. Мэйволд, Э. Безопасность сетей [Электронный ресурс] Режим доступа: https://www.studentlibrary.ru/ru/book/ISBN5957000469.html, (дата обращения 06.12.2021)

11. Небезопасность использования простых текстовых протоколов [Электронный ресурс] Режим доступа: https://netberg.ru/supports/article/text_protocol/, (дата обращения 06.12.2021)

12. Протокол Диффи-Хеллмана [Электронный ресурс] Режим доступа: https://cryptography.ru/ref/протокол_диффи-хеллмана/, (дата обращения 10.12.2021)

13. Симметричная криптография [Электронный ресурс] Режим доступа: http://mf.grsu.by/UchProc/livak/b_protect/zok_2.htm (дата обращения 15.12.2021)

14. Сквозное шифрование: что это и зачем оно нужно вам [Электронный ресурс] Режим доступа: https://www.kaspersky.ru/blog/what-is-end-to-end-encryption/29075/, (дата обращения 10.12.2021)

15. Шнайер, Брюс Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С [Электронный ресурс] Режим доступа: https://studfile.net/preview/951064/, (дата обращения 01.12.2021)