ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

регистрационный номер

_____

название факультета

_____

название кафедры

Разработка калькулятора общего секретного ключа с использованием протокола Диффи-Хеллмана

название работы

Автор:

Гринёва Анастасия Юрьевна

фамилия, имя, отчество

МОУ лицей, 11 класс

наименование учебного заведения, класс

Научный руководитель:

Горбунов Артур Валерьевич

фамилия, имя, отчество

МГТУ им. Н.Э.Баумана, кафедра ФН-12

место работы

Старший преподаватель

звание, должность

_____

подпись научного руководителя

Москва - 2020

АННОТАЦИЯ

Цель работы - создание приложения, выполняющего генерацию общего ключа для отравителя и получателя без обмена данным ключом по открытому каналу связи. Задачи, реализуемые в данной научно – исследовательской работе достаточно актуальны, так как в 21 веке потребность людей в защищенности их личной информации резко возросла, а при помощи утилиты, предложенной в данной работе можно осуществлять безопасную передачу информации по незащищенным каналам связи.

Программный продукт выполнен в виде утилиты, работающей как в оконном режиме, так и полноэкранном. Программный продукт представляет собой приложение, выполняющее некоторые преобразования с большими простыми числами для генерации общего секретного ключа для нескольких пользователей.

В основной части работы приведены существующие протоколы генерации общего ключа, подробно описан протокол Диффи - Хеллмана, проведен краткий анализ данного протокола, а также описаны области применения остальных протоколов. Обозначена проблема, которую можно решить при помощи, предложенной мной утилиты.

В практической части работы приведены некоторые этапы разработки программного продукта и описаны основные компоненты решения. Также описаны основные этапы работы с созданием интерфейса для утилиты, приведена подробная инструкция по использованию программного продукта, описаны аналоги программного продукта, существующие на данный момент, составлены перспективы дальнейшей разработки.

Новизна разработки заключается в том, что в открытом доступе подобных программных продуктов мало. Но даже те, которые существуют, не всегда соответствуют критериям, необходимым для корректной работы пользователей с данным утилитами.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В ходе развития систем обмена и защиты информации возникало множество проблем, связанных с передачей информации.

Одну из этих проблем можно сформулировать так:

Предположим, мы хотим обменяться письмом, содержащим коммерческую тайну, по электронной почте (электронная почта - открытый канал связи, поскольку сообщения на почтовом сервере хранятся в незашифрованном виде и поэтому они доступны системным администраторам этих серверов). В принципе, это не сложно: отправитель берёт практически любой архиватор (7zip, winrar и т.п.) и архивирует своё сообщение, установив на архив пароль (ключ). Такой зашифрованный архив он отправляет получателю, а получатель, зная пароль, его расшифровывает. Как получатель узнает ключ? Для этого его тоже необходимо передать от отправителя к получателю. Но как? Не станем же мы передавать пароль по тому же открытому каналу связи, а если мы передадим пароль от архива в зашифрованном виде, то как передать ключ для расшифровки.

В связи с этим возникает вопрос о том, каким образом можно обеспечить обмен информацией по открытому каналу связи, при этом сохраняя её секретность. Для осуществления такого обмена необходимо реализовать один из алгоритмов симметричного или асимметричного шифрования. При использовании асимметричных протоколов шифрования получатель создаёт два связанных друг с другом ключа: открытый и приватный. Открытый ключ позволяет только шифровать сообщение, а для расшифровки требуется приватный ключ. Открытый ключ получатель передаёт отправителю по любому каналу связи, если злоумышленник его перехватит, то, как и отправитель, сможет только шифровать сообщения, а расшифровывать - нет.

По сравнению с симметричными протоколами шифрования (это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для декодирования информации) ассиметричные более сложные в реализации. Кроме того, асимметричные криптографические системы, как правило, уязвимы к атаке с использование квантовых вычислений. Поэтому на данный момент для шифрования информации часто используются симметричные алгоритмы, а генерация общего ключа отравителя и получателя осуществляется с помощью специальных криптографических протоколов.

Симметричные алгоритмы применяются для шифрования сообщений в мессенджерах, в блокчейне для шифрования данных о транзакциях.

Одним из алгоритмов, используемых в протоколах генерации общего ключа, является алгоритм Диффи – Хеллмана, при помощи которого я реализовала программный продукт «Разработка калькулятора общего секретного ключа с использованием протокола Диффи-Хеллмана».

1. ОСНОВНАЯ ЧАСТЬ

1.1. СПИСОК ТЕРМИНОВ, ИСПОЛЬЗУЕМЫХ В РАБОТЕ

Ути́лита - вспомогательная компьютерная программа в составе общего программного обеспечения для выполнения специализированных типовых задач, связанных с работой оборудования и операционной системы (ОС).

Библиоте́ка - сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО).

Интерфе́йс - граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы.

Реестр Windows - иерархически построенная база данных параметров и настроек в большинстве операционных систем Microsoft Windows.

Простое число - натуральное число, имеющее ровно два различных натуральных делителя: единицу и само себя. Все остальные натуральные числа, кроме единицы, называются составными.

Алгоритм Диффи – Хеллмана - криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи.

1.2. ПОСТАНОВКА ЗАДАЧИ

Целью работы является создание приложения, выполняющего генерацию общего секретного ключа для отравителя и получателя без обмена данным ключом по открытому каналу связи.

Одной из основных задач, при разработке данного приложения было создание понятного интерфейса, удобного в использовании. А также подбор и создание алгоритма, который будет выполняться программой за наименьшее время, при этом не влияя на корректность работы приложения.

Для достижения поставленной цели и решения данных задач были предприняты следующие действия:

- изучение алгоритма Диффи – Хеллмана;

- создание и сравнение нескольких вариантов интерфейса;

- реализация операций с большими числами для создания стойкого к атакам ключа;

- создание генератора больших псевдо - случайных чисел;

- реализация алгоритма проверки числа на простоту;

- реализация процедуры преобразования числа в последовательность текстовых символов;

- реализация работы с реестрами Windows.

1.3. ПРОТОКОЛЫ ГЕНЕРАЦИИ ОБЩЕГО КЛЮЧА

В ходе анализа возможных вариантов реализации программного продукта, я рассмотрела несколько видов криптографических протоколов генерации общего ключа.

СОГЛАШЕНИЕ ОБ ОБОЗНАЧЕНИЯ, ВСТРЕЧАЮЩИХСЯ ПРИ ОПИСАНИИ АЛГОРИТМА ДИФФИ - ХЕЛЛМАНА

— Алиса, Боб, Трент, Злоумышленник… — имена пользователей, встречающихся в протокольных сообщениях. Иногда вместо них используются аббревиатуры A, B, T, M…

— Алиса — Бобу: M. Алиса посылает Бобу сообщение M. Спецификация протокола представляет собой последовательность таких сообщений.

— {M}k — зашифрованный текст, представляющий собой сообщение M, зашифрованное ключом K

— K, Kab, Kat, Ka… — криптографические ключи, где Kxy — ключ, разделенный между пользователями X и Y, а Kx — открытый ключ, принадлежащий пользователю X.

— N, Na… — одноразовые случайные числа. Число Nx генерируется пользователем X.

1.3.1. ПРОТОКОЛ ДИФФИ - ХЕЛЛМАНА

Алгоритм Диффи - Хеллмана позволяет двум или более пользователям обменяться без посредников ключом, который может быть использован затем для симметричного шифрования.

ОПИСАНИЕ ПРОТОКОЛА

- Предположим, существует два абонента: Алиса и Боб.

- Обоим абонентам известны некоторые общедоступные числа g и p.

1 Этап

- Алиса генерирует большое число - a, вычисляет сообщение A и пересылает его Бобу:

$$Alice \rightarrow \{A = g^a \mod p\} \rightarrow Bob.$$

- Боб генерирует большое число - b, вычисляет сообщение B и пересылает его Алисе:

$$Bob \rightarrow \{B = g^b \mod p\} \rightarrow Alice.$$

Предполагается, что злоумышленник может получить оба этих значения, но не модифицировать их.

2 Этап

Алиса на основе имеющегося у нее числа a и полученного по сети B вычисляет значение

$$K_A = B^a \mod p = g^{ab} \mod p.$$

Боб на основе имеющегося у нее числа b и полученного по сети A вычисляет значение

$$K_B = A^b \mod p = g^{ab} \mod p.$$

Как нетрудно видеть, у Алисы и Боба получилось одно и тоже число

$$K = g^{ab} \mod p.$$

Его они и могут использовать в качестве секретного ключа.

Проанализировав некоторые протоколы и протокол Диффи - Хеллмана, я поняла, что протокол Диффи—Хеллмана отличается от большей части протоколов распространения ключей тем, что не использует другие криптографические примитивы (функции шифрования, электроно-цифровой подписи или хеширования), но сам по себе является в некотором смысле криптографическим примитивом для построения более сложных протоколов.

Он обеспечивает генерацию случайного числа в распределённой системе без доверенного центра. Причём ни одна из сторон не может заставить другую сторону использовать старый сессионный ключ.

В связи с этим я решила, что наиболее правильно и корректно будет использовать именно протокол Диффи – Хеллмана для выполнения задачи, решаемой мной в данной работе. Также можно сказать, что использование алгоритма Диффи – Хеллмана и его реализация наиболее экономически выгодна при решении поставленной задачи (так как по сравнению с другими протоколами, в данном протоколе используется упрощенный вариант математического алгоритма, при этом данная особенность не влияет на качество работы реализуемого мной программного продукта).

1.3.2. НЕКОТОРЫЕ ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ НА ОСНОВЕ АЛГОРИТМА ДИФФИ - ХЕЛЛМАНА

протоколы MTI

Протоколы MTI - семейство протоколов распределения ключей, разработанные Т. Мацумото (T. Matsumoto), И. Такасита (Y. Takashita) и Х. Имаи (H. Imai) и названные по именам авторов. Протоколы MTI делятся на три класса протоколов: MTI/A, MTI/B, MTI/C.

протокол Жиро

Протокол - криптографический протокол, позволяющий двум сторонам получить общий секретный ключ, не используя явную сертификацию. Полученный ключ используется для шифрования дальнейшей обмениваемой информации с помощью симметричного алгоритма шифрования.

протокол STS

Протокол STS (Station-To-Station) - протокол формирования общего ключа с взаимной аутентификации сторон. Является модификацией

протокола Диффи-Хеллмана, защищённого от атаки "третий между" за счёт механизма аутентификации.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. ИСПОЛЬЗУЕМЫЕ МЕТОДЫ

Для разработки приложения использовались язык программирования Python3 и среда программирования PyCharm. Данная среда была выбрана в связи с тем, что она обладает достаточным набором инструментов для удобного и быстрого прототипирования программного продукта. (Например, интерпретатором – программа, выполняющая преобразование псевдо – кода в машинный.)

Библиотека Tkinter. Библиотека применялась для создания графического пользовательского интерфейса. Создание визуальной части программного продукта было реализовано с помощью методов и функций данной библиотеки.

Библиотека Winreg. Библиотека использовалась для работы с реестрами Windows.

2.2.1. ДЕМОНСТРАЦИЯ НЕКОТОРЫХ УЧАСТКОВ КОДА

Целью практической части работы являлась разработка программного обеспечения для алгоритма генерации устойчивого к подбору ключа.

Представленный ниже участок кода является реализацией алгоритма Диффи – Хеллмана (реализация операций с большими числами для создания стойкого к атакам ключа) и выполняет функцию создания параметров q и g.

```
@classmethod
def calc_q(cls, k: int) -> int:
    q = 2 * (k // 2) + 1
    while True:
        if cls._miller_rabin(q, 5):
            p = 2 * q + 1
            if cls._miller_rabin(p, 5) and cls._miller_rabin(q, 150) and cls._mil
ler_rabin(p, 150):
                break
        q += 2
    return q

@classmethod
def calc_g(cls, q: int) -> int:
    p = 2 * q + 1
    g = random.randint(q // 10, 2 * q)
    while pow(g, q, p) == 1:
        g = random.randint(q // 10, 2 * q)
    return g
```

Для корректной работы алгоритма программного продукта с использованием протокола Диффи – Хеллмана требовалось реализовать алгоритм проверки чисел на простоту. В ходе подбора алгоритмов для решения данной задачи, было реализовано несколько алгоритмов, но при проверке их корректности оказалось, что время, затрачиваемое на работу данного алгоритма слишком велико в масштабах решаемой мной задачи. В связи с этим было решено реализовать алгоритм Миллера – Рабина. Он позволяет осуществлять проверку 256-битного числа на простоту. Представленный ниже участок кода является реализацией алгоритма Миллера – Рабина. Функция miller_rabin вызывается 4 раза: два раза для тестирования q и два раза для тестирования p. Это сделано для ускорения поиска, первый вызов служит, чтобы сразу отсеять явно простые числа p или q. Аргумент k функции miller_rabin отвечает за количество элементарное тестирований по алгоритму Миллера-Рабина, а каждое элементарное тестирование снижает вероятность ошибки примерно в 1/4 раза. Таким образом, если число элементарных тестирований примерно равно числу бит в тестируемом числе, то ошибка практически исключена.

```
@classmethod
    def _miller_rabin(cls, n: int, k: int = 100) -> bool:
        if n == 2 or n == 3:
            return True
        if not n & 1 or n <= 1:
            return False

        def _check(input_a: int, input_s: int, input_d: int, input_n: int) -
> int:
            x = pow(input_a, input_d, input_n)
            if x == 1:
                return True
            for _ in range(input_s - 1):  # O(N)
                if x == input_n - 1:
                    return True
                x = pow(x, 2, input_n)
            return x == input_n - 1

        s = 0
        d = n - 1

        while d % 2 == 0:
            d >>= 1
            s += 1

        for _ in range(k):
            a = random.randint(2, n - 2)
            if not _check(a, s, d, n):
                return False
        return True

    @classmethod
    def next_prime(cls, i: int, k: int = 10) -> int:
        while not cls._miller_rabin(i, k):
            i += 1
        return i
```

В связи с тем, что за один сеанс пользователь может общаться не с одним контрагентом, а с несколькими, возник вопрос о том, где хранить ключи в течении сеанса. Хранить ключи в текстовом файле небезопасно, а предложить пользователю хранить ключи в физическом виде (например, на листе бумаги) нерационально, учитывая длину ключа. Поэтому было принято решение осуществить работу с реестрами, и сохранять ключи туда. Для выполнения

этой задачи была использована библиотека Winreg. Представленный ниже участок кода является реализацией работы с реестрами Windows.

```
import winreg

REG_PATH = r"SOFTWARE\DH_CALC\Data"


def set_reg(name, value):
    try:
        winreg.CreateKey(winreg.HKEY_CURRENT_USER, REG_PATH)
        registry_key = winreg.OpenKey(
            winreg.HKEY_CURRENT_USER,
            REG_PATH,
            0,
            winreg.KEY_WRITE
        )
        winreg.SetValueEx(registry_key, name, 0, winreg.REG_SZ, value)
        winreg.CloseKey(registry_key)
        return True
    except WindowsError:
        return False


def get_reg(name):
    try:
        registry_key = winreg.OpenKey(
            winreg.HKEY_CURRENT_USER,
            REG_PATH,
            0,
            winreg.KEY_READ
        )
        value, regtype = winreg.QueryValueEx(registry_key, name)
        winreg.CloseKey(registry_key)
        return value
    except WindowsError:
        return None
```

Реализация генератора больших псевдо - случайных чисел была осуществлена при помощи участка кода, представленного ниже.

```
k = random.randint(pow(2, 189), pow(2, 191))
```

Также при написании программы была осуществлена реализация процедуры преобразования числа в последовательность текстовых символов (цифровой ключ был преобразован в текстовый).

2.2.2. СОЗДАНИЕ И СРАВНЕНИЕ ВАРИАНТОВ ИНТЕРФЕЙСА

Для создания интерфейса использовалась библиотека Tkinter. Основными требованиями к интерфейсу были удобность и понятность. В ходе разработки интерфейса я рассмотрела несколько вариантов.

В первом варианте интерфейса (рис. 1) я обнаружила определенную нелогичность. Например, пользователю было бы непонятно, как использовать данную программу, какую последовательность действий нужно выполнять. Для того, чтобы решить данный вопрос я добавила окно «Помощь». Также я заметила, что при работе с несколькими контрагентами пользователю нужно где-то хранить пароли, поэтому была добавлено окно «реестр», в котором производится сохранение паролей в реестр Windows. Завершив данный этап разработки интерфейса, я попробовала сменить цвет фона утилиты, но сменив его (рис. 2), поняла, что первоначальный цвет выглядит наиболее гармонично.

Таким образом, я пришла к окончательному варианту интерфейса, который продемонстрирован на рис. 3

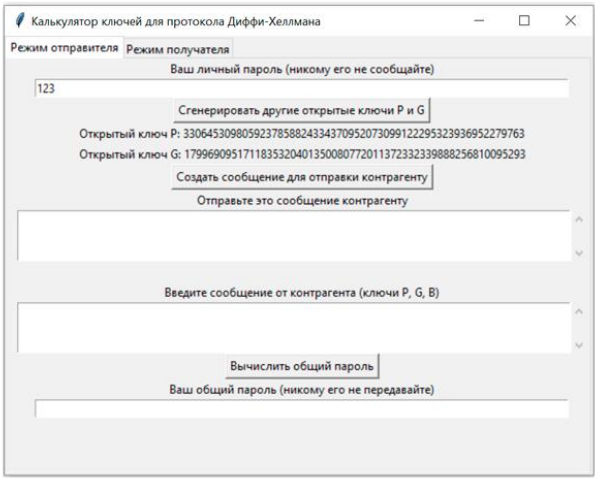

Рис. 1

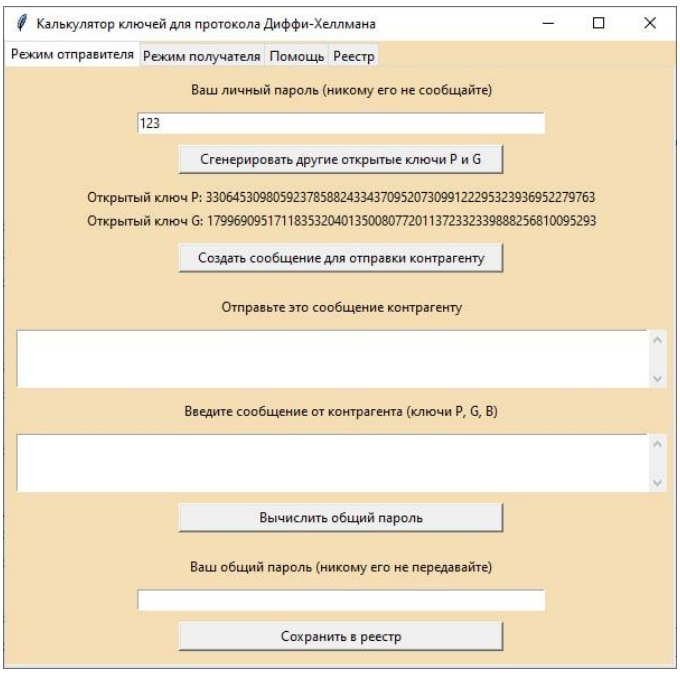
Калькулятор ключей для протокола Диффи-Хеллмана
Режим отправителя
Режим получателя
Помощь
Реестр
Ваш личный пароль (никому его не сообщайте)
123
Сгенерировать другие открытые ключи P и G
Открытый ключ P: 330645309805923785882433437095207309912229532393695227976 3
Открытый ключ G: 179969095171183532040135008077201137233233988825681009529 3
Создать сообщение для отправки контрагенту
Отправьте это сообщение контрагенту
Введите сообщение от контрагента (ключи P, G, B)
Вычислить общий пароль
Ваш общий пароль (никому его не передавайте)
Сохранить в реестр

Рис. 2

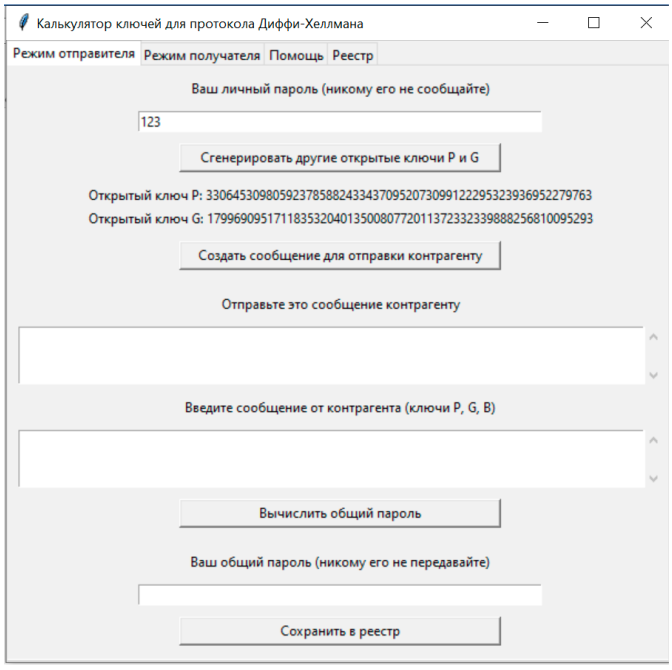
Калькулятор ключей для протокола Диффи-Хеллмана
Режим отправителя
Режим получателя
Помощь
Реестр
Ваш личный пароль (никому его не сообщайте)
123
Сгенерировать другие открытые ключи P и G
Открытый ключ P: 330645309805923785882433437095207309912229532393695227976 3
Открытый ключ G: 179969095171183532040135008077201137233233988825681009529 3
Создать сообщение для отправки контрагенту
Отправьте это сообщение контрагенту
Введите сообщение от контрагента (ключи P, G, B)
Вычислить общий пароль
Ваш общий пароль (никому его не передавайте)
Сохранить в реестр

Рис. 3

2.2.3. ИНСТРУКЦИЯ ПО ИСПОЛЬЗОВАНИЮ УТИЛИТЫ

1) Выберите вкладку в зависимости от того, кем Вы являетесь (отправителем или получателем)

2) Введите в верхнее поле своей вкладки свой личный пароль (там по умолчанию находится трёхзначный пароль)

3) Сгенерируйте заново ключи p и g

Далее Ваши действия зависят от того, отправитель Вы или получатель.

Если Вы отправитель, следуйте следующим инструкциям:

1) Нажмите кнопку "Создать сообщение для отправки контрагенту";

2) Из данного окошка скопируйте сообщение и отправьте его контрагенту (получателю);

3) Получив от контрагента ответное сообщение, вставьте его в окошко ниже;

4) Нажмите кнопку "Вычислить общий пароль".

Если же Вы получатель, то следуйте следующим инструкциям:

1) Получив от контрагента (отправителя) сообщение, вставьте его в соответствующее окошко;

2) Нажмите кнопку "Создать сообщение для контрагента и вычислить общий пароль";

3) Из окошка ниже скопируйте сообщение и отправьте его контрагенту (отправителю).

2.3. АНАЛОГИ

Полных аналогов программы найдено не было. На данный момент создается малое количество подобных приложений, а те, которые создаются, не находятся в открытом доступе. Однако, учитывая то, что аналогов мало,

они все же существуют, хотя и выполняют ту же функцию, что и утилита, созданная мной (при этом, данные приложения имеют тот же функционал, что и утилита, но являются более сложными в реализации). Также стоит отметить, что, выполняя данный проект я стремилась создать утилиту именно по протоколу Диффи – Хеллмана, и аналогов такой программы, работающей по данному протоколу, не было найдено.

Примеры:

- PuTTY Key Generator

«PuTTY Key Generator – это генератор для создания пар открытых и закрытых ключей.» Первоначально данный генератор был написан для операционной системы Microsoft Windows, теперь он официально доступен для нескольких операционных систем, включая macOS, Linux.

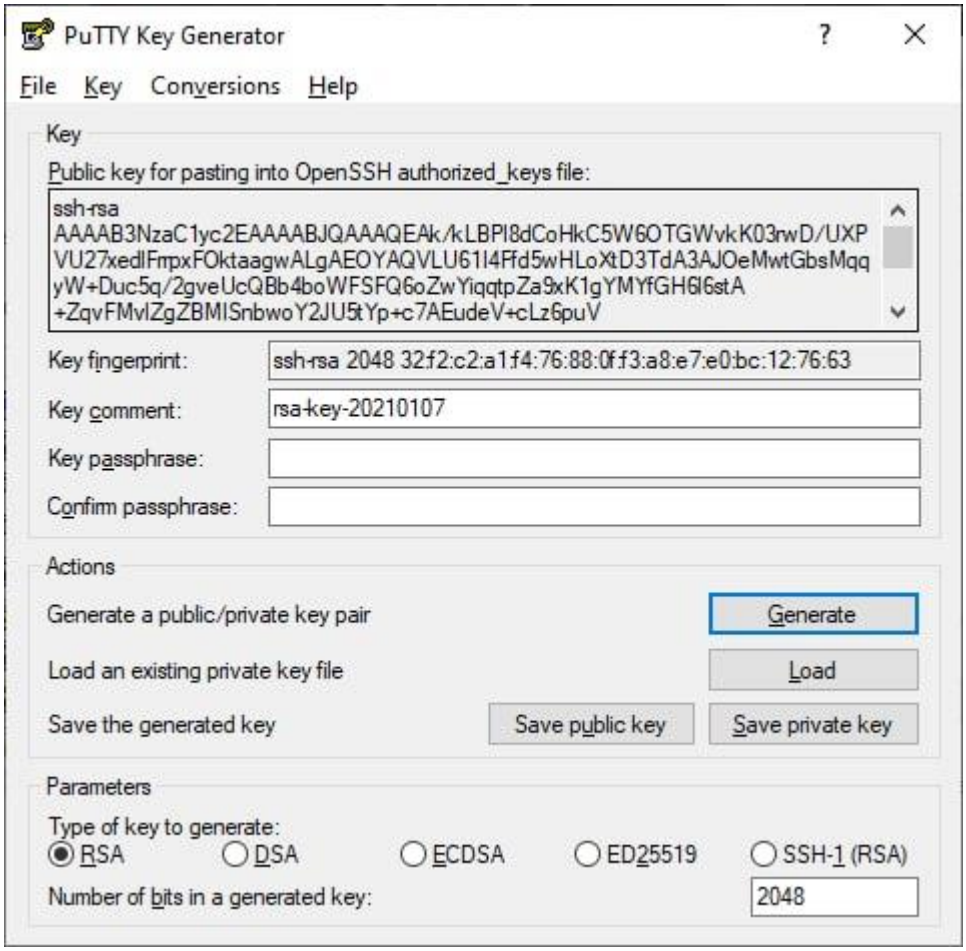

Рис. 4

- KeyGen

«KeyGen - это программа, которая помогает защитить учетные записи и зашифровать документы, генерируя ключи.»

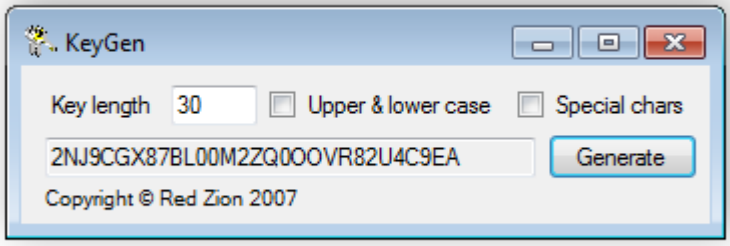

Рис.5

Таблица для анализа преимуществ и недостатков разработанного программного продукта

	Быстродействие (от 0 до 10)	Использование алгоритма Диффи - Хеллмана	Генерация нескольких ключей в одном сеансе	Понятность интерфейса (от 0 до 10)
PuTTY Key Generator	9	-	+	6
KeyGen	10	-	-	10
Утилита по протоколу Диффи - Хеллмана	8	+	+	7

2.4. ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕЙ РАЗРАБОТКИ

Так как использование алгоритмов шифрования и передачи зашифрованной информации актуально на данный момент, то я считаю, что стоит совершенствовать данное приложение.

Планируется доработка текущей версии программного продукта: усовершенствование интерфейса, алгоритма работы программы, реализация наиболее сложных протоколов, основанных на алгоритме Диффи - Хеллмана, но при этом более сложных (например, протокола MTI).

Стоит отметить, что алгоритм Диффи – Хеллмана уязвим к атаке «третий между» и к «перебору» секретных ключей. Именно для устранения данной проблемы планируется усовершенствовать программу.

Также утилита не полностью приспособлена к использованию более, чем двумя контрагентами. В ближайшем времени планируется устранить данный недостаток.

На данный момент в процессе разработки находится функция, которая обеспечит кроссплатформенность утилиты.

ЗАКЛЮЧЕНИЕ

В данном проекте в рамках разработки калькулятора общего секретного ключа с использованием протокола Диффи-Хеллмана, была проведена работа в следующих направлениях:

- рассмотрены задачи, необходимые для реализации программного продукта задачи
- подтверждена актуальность данной задачи
- произведен выбор алгоритма, необходимого для реализации утилиты
- разработан программный продукт, отвечающий поставленным задачам

В основной части работы был проведен анализ некоторых возможных методов реализации поставленной задачи. Приведено подробное описание используемого алгоритма.

В практической части работы был создан программный продукт, позволяющий генерировать общий ключ для нескольких пользователей,

описаны методы создания утилиты. Кроме того, был разработан удобный как для специализированных сотрудников, так и для простых потребителей интерфейс, совмещенный с программой.

Выполненная разработка имеет существенное практическое значение: она может применяться в абсолютно любых сферах хранения и передачи информации (начиная обменом данными в социальных сетях, заканчивая передачей данных о транзакциях).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мао В. Современная криптография: теория и практика : Пер. с англ. — М.: 2005. — 768 с.
2. Запечников С. В., Казарин О. В., Тарасов А. А. Криптографические методы защиты информации : учебник для вузов. — Москва : Издательство Юрайт, 2020. — 309 с.
3. Федюшина Е. О., Балашов М. А. Уязвимость и защита алгоритма Диффи - Хеллмана // Безопасность информационного пространства : материалы XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых, Екатеринбург, 2-4 декабря 2013 г. — Екатеринбург : Изд-во Урал. ун-та, 2014. — С. 247-253.

4. Войтович А.М. Сравнительный анализ реализаций протокола обмена ключами Диффи – Хеллмана. — бакалаврская работа. — Тольятти: Тольяттинский государственный университет, 2016. — 50 с.