Муниципальное бюджетное общеобразовательное учреждение Гимназия

ПРОЕКТНАЯ РАБОТА ПО ТЕХНОЛОГИИ

Направление Информационная безопасность

Тема: «Постквантовая криптография KEM на решётках»

Выполнил учащийся 10 класса

Брылев Альберт

Руководитель работы

Оборин Дмитрий Евгеньевич

*г. Новый Уренгой 2025*

**Содержание**

1. Выбор, обоснование и актуальность
2. Цели и задачи
3. Анализ современных протоколов
4. Изучение математических методов шифрования
5. Выбор инструментов для разработки проектов
6. Создание протокола
7. Заключение
8. Ссылки и список литературы

**1. Выбор, обоснование и актуальность**

На сегодняшний день криптографические протоколы обеспечивают безопасность передаваемой информации за счёт сложности современных алгоритмов шифрования. Однако, с появлением квантовых компьютеров ситуация может кардинально измениться. В настоящее время данные могут быть перехвачены, но расшифровать их без знания секретного ключа практически невозможно. При этом, как только квантовые компьютеры достигнут мощности в 3–4 тысячи кубитов, они смогут эффективно взламывать существующие криптографические алгоритмы.

К примеру, по расчетам экспертов, RSA-2048 может быть успешно атакован квантовым компьютером, обладающим около 4000 кубитов. Уже сегодня созданы квантовые устройства с 1000 кубитами, что свидетельствует о быстром прогрессе в этой области. Предполагается, что к 2030 году появятся машины с необходимой мощностью для решения подобных задач.

Стоит также отметить недавние достижения компаний, таких как Google, которые разработали технологию компактных квантовых компьютеров, способных разместить 105 кубитов на плате размером 4×4 см. Такой технологический скачок может ускорить развитие квантовых вычислений ещё больше, что в свою очередь подчеркивает необходимость своевременного перехода на постквантовые криптографические протоколы.

Почему нельзя откладывать разработку новых систем шифрования? Если ждать дальнейшего развития квантовых технологий, может наступить момент, когда будут недостаточно времени и ресурсов для создания и внедрения новых протоколов. Стандартизация первых постквантовых протоколов уже заняла около 6 лет (2016–2022), и в условиях стремительно меняющейся технологической среды задержки могут привести к тому, что обмен информацией останется уязвимым до тех пор, пока не будут разработаны и приняты новые стандарты безопасности.

Переход на постквантовую криптографию является не только актуальной, но и крайне необходимой мерой для обеспечения безопасности данных в условиях стремительного развития квантовых вычислительных технологий, особенно в государственных и критически важных секторах.

**2. Цели и задачи**

Цель: разработка, анализ и реализация постквантового криптографического протокола типа KEM (Key Encapsulation Mechanism), основанного на решётках и математической проблеме MLWE (Module Learning With Errors), аналогичной используемой в протоколе Kyber. В рамках работы предполагается:

1. **Исследование и применение математических основ:**  
   Изучение проблемы MLWE, являющейся фундаментальной для создания криптографических схем на решётках. В проекте планируется провести глубокий анализ сложности MLWE и оценить его стойкость к различным типам атак, а также адаптировать математические методы для построения безопасного протокола.
2. **Разработка эффективного механизма инкапсуляции ключей:**  
   Создание протокола KEM, который обеспечивает надёжный обмен ключами, позволяя безопасно передавать криптографические ключи между участниками коммуникации. Особое внимание будет уделено оптимизации алгоритма для достижения высокой производительности при минимальных вычислительных затратах.
3. **Практическая реализация и оценка протокола:**  
   Реализация прототипа предложенного решения с последующим проведением тестирования на устойчивость к атакам, а также сравнительный анализ с существующими постквантовыми решениями, такими как Kyber. Оценка будет проводиться как с точки зрения безопасности, так и эффективности и скоростью работы.

Реализация протокола на основе MLWE позволит не только повысить устойчивость обмена ключами, но и обеспечить практическую возможность перехода на постквантовые методы шифрования в государственных, коммерческих и других критически важных сферах.

**3. Анализ современных протоколов**

**4 Анализ уязвимостей и угроз**

В условиях постоянного развития цифровых технологий и растущей зависимости от современных способов общения, критически важно проанализировать и оценить уязвимости и угрозы безопасности, с которыми сталкиваются существующие методы обмена данными. Этот раздел направлен на разбор ключевых аспектов, включая уязвимости кибератак, социальную инженерию, проблемы управления доступом,

Межплатформенные атаки представляют собой категорию киберугроз, направленных на мессенджеры, которые преодолевают границы между различными платформами и устройствами, создавая угрозу для пользовательской конфиденциальности и безопасности данных. Ниже рассмотрены основные виды межплатформенных атак.

"*Man-in-the-Middle" или "человек посередине"* – это атака на сетевую коммуникацию, при которой злоумышленник встраивается между двумя коммуницирующими сторонами. Этот злоумышленник может просматривать и даже изменять передаваемую информацию без ведома или разрешения обеих сторон.

В типичном сценарии Man-in-the-Middle атаки злоумышленник перехватывает или подменяет коммуникацию между двумя узлами (например, клиентом и сервером). Это может быть достигнуто различными способами, такими как перехват пакетов данных, подделка сертификатов SSL/TLS, атаки на DNS и другие методы.

Примеры сценариев Man-in-the-Middle атак включают перехват паролей, кредитных карт, аутентификационных данных и другой чувствительной информации. Так в 2016 году кампания "WhatsApp Sniffer" осуществляла Man-in-the-Middle атаки, перехватывая данные, передаваемые через протокол синхронизации мессенджера WhatsApp.

*Внедрение вредоносных файлов* – это процесс внедрения вредоносного программного кода или файлов в компьютерную систему или сеть, таких как вирусы, черви, троянские заправки, эксплойты. Этот термин обычно используется в контексте кибератак, когда злоумышленники пытаются проникнуть в систему с целью нанесения ущерба, кражи информации или выполнения других вредоносных действий.

Примером внедрения вредоносных файлов, является атака мае 2017 года. Эта глобальная кибератака затронула компьютерные системы в более чем 150 странах и оказала воздействие на тысячи организаций, включая правительственные учреждения, здравоохранение, транспортные компании и многие другие.

WannaCry был распространен через уязвимость в почтовом клиенте Microsoft Windows, вредоносное программное обеспечение шифровало данные на зараженных компьютерах и требовало выкуп.

*Атаки на ролевую авторизацию* - злоумышленники манипулируют ролями или правами доступа, чтобы получить больше привилегий, чем предусмотрено.

*Перехват сессий* - злоумышленники захватывают активную сессию пользователя для несанкционированного доступа к системе.

Так же еще одной уязвимостью являются открытые исходные коды мессенджеров, хотя и способствуют прозрачности и общественному контролю, также могут стать объектом внимания злоумышленников, которые могут обнаруживать уязвимости и вредоносные возможности в кодовой базе. В этом контексте рассмотрим ключевые характеристики и примеры атак на кодовую базу мессенджеров. Примером является обнаруженная уязвимость в коде Telegram, известной как "Media File Jacking", которая позволяла атакующим изменять медиафайлы до того, как они достигли получателя.

Еще одной не менее значимой уязвимостью является межплатформенная уязвимость из-за различий в реализации на разных платформах могут привести к уязвимостям, которые злоумышленники могут использовать для атак. Примером является атака кампании "Skygofree" на уязвимость в веб-версии Telegram, что дало атакующим возможность вмешательства в личные чаты и прослушивания аудиозаписей.

*Фишинг и социальная инженерия* представляют собой распространенные и опасные методы атак, особенно в сфере мессенджеров, где социальный аспект взаимодействия усиливает вероятность успешных атак. Ниже рассмотрены ключевые черты этих атак, их вариации и примеры в контексте мессенджеров.

Фишинг – метод, при котором злоумышленник выдает себя за доверенное лицо, отправляя фальшивые сообщения с целью мошенничества или выманивания конфиденциальной информации [6]. Фишинг делиться на несколько видов:

Ссылочный фишинг – это отправка ссылок на поддельные веб-сайты, имитирующие официальные страницы мессенджера для получения данных.

Фишинг через вложения – это отправка вредоносных вложений или файлов, маскированных под безвредные документы.

Примером выше сказанного является кампания "WhatsApp Gold", где пользователи получали предложение установить премиальную версию WhatsApp, но на самом деле сталкивались с вредоносным программным обеспечением.

*Социальная инженерия* - техника, в которой атакующий манипулирует человеческим фактором, обманывая или вводя в заблуждение пользователей. Делиться на несколько видов:

Имитация доверенных лиц – это вид атаки при котором атакующие могут представляться друзьями, коллегами или даже службами поддержки мессенджера.

Социальная инженерия в чате – это вид атаки при котором злоумышленники могут манипулировать членами группового чата, притворяясь членами сообщества или администраторами.

Примером социальной инженерии является атака на компанию Target в 2013 году. В этом инциденте хакеры использовали социальную инженерию, чтобы получить доступ к системе компании.

Хакеры взломали систему через стороннего поставщика, поставляющего системы отопления и вентиляции в магазины Target. Они использовали учетные данные подрядчика, чтобы получить доступ к внутренней сети компании.

Перед тем как запустить атаку, хакеры собрали информацию о сотрудниках Target и их обычных рабочих процедурах. Затем они отправили фишинговые электронные письма некоторым сотрудникам, притворяясь сотрудниками поддержки. В электронных письмах содержались вредоносные вложения, которые, как им казалось, требовались для устранения проблем с системой. Когда сотрудники открывали эти вложения, хакеры получали доступ к системам компании.

В результате этой атаки были скомпрометированы миллионы кредитных картных данных клиентов Target. Этот случай подчеркивает, насколько важно обучать сотрудников узнавать и предотвращать атаки, основанные на социальной инженерии, и соблюдать меры безопасности в цифровом пространстве.

**5 Изучение методов шифрования**

В эпоху цифровой коммуникации, где мессенджеры играют ключевую роль в обмене информацией, безопасность передачи данных становится критической задачей. В данном разделе мы проведем глубокий анализ современных методов шифрования в мессенджерах, включая принципы, используемые алгоритмы, протоколы и их влияние на общую безопасность сообщений. Это исследование позволит оценить эффективность шифрования в контексте обеспечения конфиденциальности переписки.

Одним из ключевых элементов обеспечения конфиденциальности и целостности информации является применение эффективных методов шифрования.

Симметричное шифрование, основанное на использовании одного и того же ключа для шифрования и дешифрования данных, предоставляет быстрый и простой способ обеспечения безопасности, особенно при передаче данных внутри локальных сетей или защите файлов. Современные алгоритмы, такие как AES и DES, являются популярными представителями этого метода.

Асимметричное шифрование, в свою очередь, предоставляет пару ключей - открытый и закрытый. Это позволяет безопасно обмениваться данными и использовать цифровые подписи. RSA и ECC являются широко используемыми алгоритмами в этой области.

Хэширование, представленное методом создания хэш-кода данных, обеспечивает надежный способ проверки целостности информации и хранения паролей. SHA-256 и MD5, несмотря на свои особенности, до сих пор применяются для этих целей.

Гибридное шифрование совмещает симметричные и асимметричные методы для оптимизации производительности и безопасности, что находит применение в обеспечении безопасности обмена данными большого объема.

Новаторским направлением в обеспечении безопасности становится квантовое шифрование, использующее принципы квантовой механики. Он обеспечивает абсолютную безопасность передачи данных, противостоя современным вычислительным вызовам.

Все эти методы шифрования являются инструментами, подстраиваемыми под различные требования безопасности и характеристики передаваемых данных. Использование комбинации этих методов создает надежную систему обеспечения безопасности, подготовленную к современным вызовам и угрозам в цифровой среде.

Внедрение эффективных протоколов шифрования в мессенджерах является неотъемлемым элементом обеспечения конфиденциальности и защиты информации от потенциальных угроз.

Протокол TLS/SSL, используемый для шифрования трафика между устройствами пользователя и сервером мессенджера, является стандартным средством обеспечения внутренней защиты от несанкционированного доступа. Он гарантирует конфиденциальность передаваемых данных и предотвращает атаки перехвата в открытых сетях.

Signal Protocol, применяемый в WhatsApp и Signal, обеспечивает конечный шифрованный обмен сообщениями. Этот протокол гарантирует безопасность коммуникации, предоставляя аутентификацию пользователей и защиту от активного перехвата данных.

Протокол Off-the-Record (OTR) предоставляет механизм конфиденциального общения, основываясь на принципе "последующей пересылки" и отказа от использованных ключей. Он обеспечивает защиту от подслушивания и предотвращает возможность дешифрования ранее перехваченных сообщений.

MTProto, разработанный для Telegram, обеспечивает шифрование данных и аутентификацию пользователей, гарантируя конфиденциальность переписки и предотвращая несанкционированный доступ к данным.

PGP (Pretty Good Privacy), используемый в ProtonMail и некоторых мессенджерах, предоставляет асимметричное шифрование и создание цифровых подписей. Этот протокол защищает содержимое сообщений и обеспечивает конфиденциальность электронной почты.

OMEMO (OMEMO Multi-End Message and Object Encryption) - протокол, обеспечивающий конфиденциальное шифрование для множества устройств пользователя, поддерживается в некоторых мессенджерах, таких как Conversations.

В контексте этих вызовов и стремления к поддержанию безопасности данных в мессенджерах, дальнейшее развитие и исследование протоколов шифрования становится неотъемлемой частью современной криптографии. Эффективное применение данных протоколов не только создает надежные механизмы защиты, но и предоставляет решения для растущих угроз, адаптируясь к современным вызовам.

Продолжение развития протоколов шифрования направлено на улучшение их устойчивости к различным атакам, включая квантовые вычисления и современные методы взлома. Исследования в области криптографии должны учитывать новейшие технологии и алгоритмы, обеспечивая не только сегодняшнюю, но и будущую безопасность данных.

Кроме того, важно соблюдать баланс между безопасностью и удобством использования мессенджеров. Протоколы шифрования должны быть не только надежными, но и эффективными, чтобы не создавать излишних сложностей для пользователей. Учитывая динамичность кибербезопасности, постоянное исследование и совершенствование протоколов становится важным элементом обеспечения долгосрочной безопасности в мессенджерах.

Таким образом, внимание к исследованиям и развитию в области протоколов шифрования остается не только актуальным, но и важным фактором в обеспечении безопасности в цифровой среде. Это требует совместных усилий со стороны исследователей, разработчиков и пользователей для создания и поддержания надежных криптографических решений, способных эффективно сопротивляться современным и будущим угрозам кибербезопасности.

**6 Обоснование выбора протоколов шифрование AES и B92**

В 1998 году Национальный институт стандартов и технологий (NIST) объявил конкурс на разработку симметричного алгоритма шифрования под названием Advanced Encryption Standard (AES). Этот конкурс был инициирован с целью замены устаревшего стандарта Digital Encryption Standard (DES), принятого в 1977 году, из-за недостаточной длины ключа и ориентации на аппаратную реализацию.

Проблемы DES, такие как короткая длина ключа (56 бит) и ориентация на аппаратную реализацию, привели к возможности относительно простого взлома методом перебора ключей. В январе 1997 года NIST объявил о намерении выбрать новый алгоритм, и 12 сентября 1997 года начался конкурс. Параметры для кандидатов были открытыми, и NIST предложил не повторять ошибки, связанные с закрытием публикаций об исследованиях алгоритмов-кандидатов.

Основные требования к кандидатам включали блочный метод шифрования с длиной блока 128 бит и размером ключа 128, 192 или 256 бит. Кандидатам также рекомендовалось ориентироваться на 32-разрядные процессоры и использовать операции, реализуемые как программно, так и аппаратно на микрочипах.

В октябре 2000 года был объявлен победитель конкурса - алгоритм Rijndael. После завершения конкурса началась процедура стандартизации, и 26 ноября 2001 года AES был принят как новый стандарт. AES, хотя и основан на алгоритме Rijndael, имеет фиксированный размер ключа (128 бит) и блока данных (16 байт). Он обеспечивает высокую степень защиты, простую структуру и высокую производительность [10].

Шифрование AES

Для шифрования в алгоритме AES применяются следующие процедуры преобразования данных:

ExpandKey — Вычисление раундных ключей для всех раундов.

SubBytes — Подстановка байтов с помощью таблицы подстановок;

ShiftRows — Циклический сдвиг строк в форме на различные величины;

MixColumns — Смешивание данных внутри каждого столбца формы;

AddRoundKey — Сложение ключа раунда с формой.

Порядок выполнения процедур 2 и 3 можно поменять местами в силу определения этих операций.

Процедуры 4 и 5 тоже можно выполнять в разном порядке, но при этом изменяется количество их вызовов, поскольку MixColumns (AddRoundKey (A, B)) = AddRoundKey(MixColumns(A), MixColumns(B)).

Шифрование производится по алгоритму, приведенному на Рисунке 1.

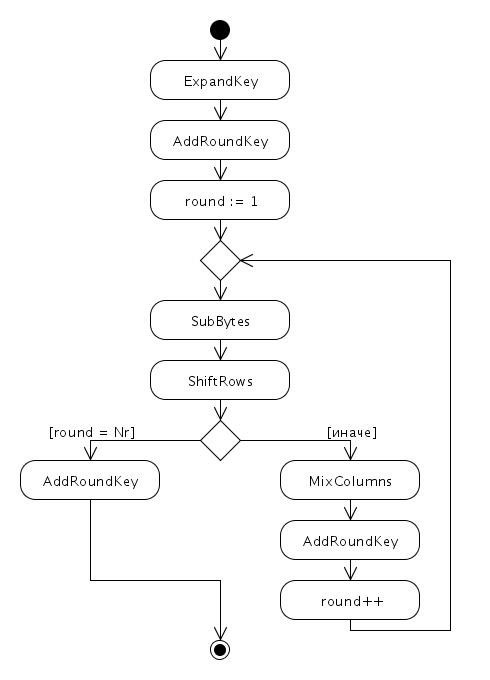


Рисунок 1–Алгоритм шифрования

Преобразование SubBytes заключается в замене каждого байта {xy} формы (где x и y обозначают шестнадцатеричные цифры) на другой в соответствии с Таблицей 3.

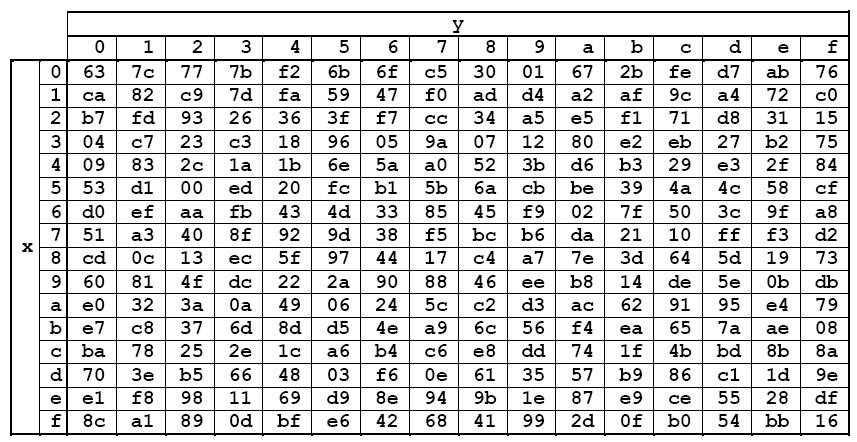


Таблица 3 – Таблица подстановок

Преобразование ShiftRows заключается в циклическом сдвиге влево строк формы. Преобразование схематично представлено на Рисунке 2. Первая строка остается неизменной. Во второй производится сдвиг на 1 байт, то есть первый байт переносится в конец. В третьей ― сдвиг на 2 байта, в четвертой ― на 3.

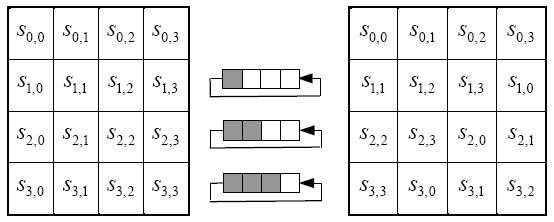


Рисунок 2 – Преобразование ShiftRows

Преобразование MixColumns заключается в умножении квадратной матрицы 4-го порядка на каждый столбец формы показанного на рисунке 3:

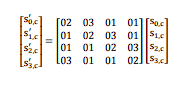


Рисунок 3 – Умножение квадратной матрицы

Умножение производится в поле Галуа GF(28).

Над каждым столбцом операция производится отдельно, как показано на Рисунке 4.

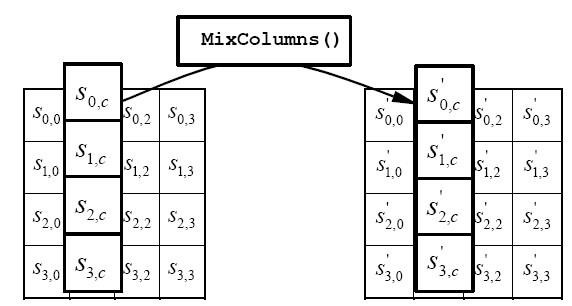


Рисунок 4– Преобразование MixColumns

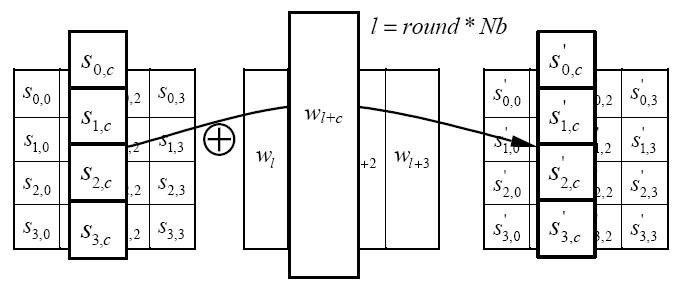
Преобразование AddRoundKey

В преобразовании AddRoundKey 32-битные слова раундного ключа прибавляются к столбцам формы с помощью побитовой операции XOR:

[𝑠0′,𝑐, 𝑠1′,𝑐, 𝑠2′,𝑐, 𝑠3′,𝑐] = [𝑠0,𝑐, 𝑠1,𝑐, 𝑠2,𝑐, 𝑠][wround\*Nb+c]

Здесь wi ― это столбцы ключа.

Над каждым столбцом операция производится отдельно, как показано на Рисунке 5.



Риcунок 5– Преобразование AddRoundKey

Процедура ExpandKey

В алгоритме AES генерируются раундные ключи на основе ключа шифрования с помощью процедуры ExpandKey. Процедура ExpandKey создает Nb \* (Nr + 1) слов: алгоритму требуется начальный ключ размером Nb, плюс каждый из Nr раундов требует ключ из Nb слов.

Здесь использованы следующие функции:

SubWord осуществляет замену каждого байта в слове в соответствии с таблицей подстановок, представленной в Табл. 3.

RotWord осуществляет циклический сдвиг байтов в слове влево 

Риcунок 6– Процедура

RotWord Rcon(i) формирует слово [02i−1, 00, 00, 00].

Дешифрование AES

При дешифровании все преобразования производятся в обратном порядке. Используются следующие обратные преобразования вместо соответствующих шифрующих:

InvSubBytes ― Подстановка байтов с помощью обратной таблицы подстановок;

InvShiftRows ― Циклический сдвиг строк в форме на различные величины;

InvMixColumns ― Смешивание данных внутри каждого столбца формы.

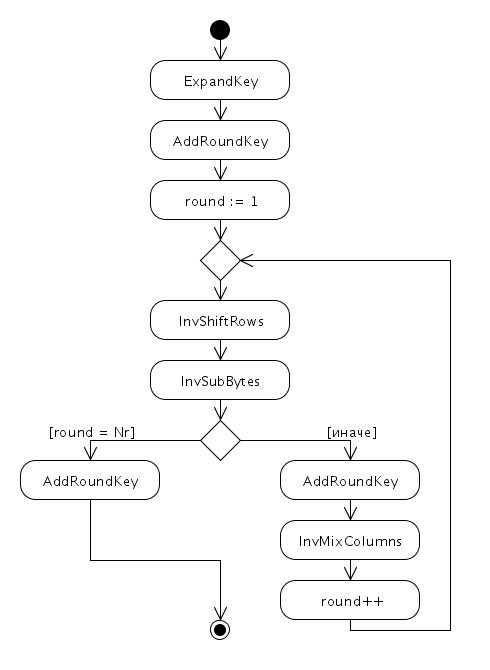


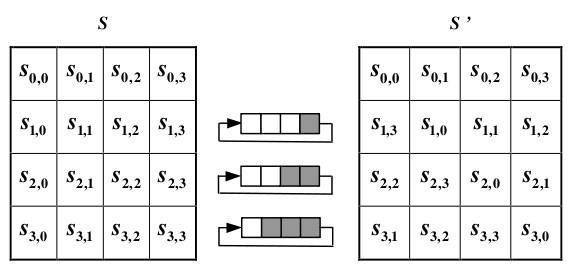
Рисунок 7 – Алгоритм дешифрования

Процедуры ExpandKey и AddRoundKey остаются неизменными. Ключи раунда используются в обратном порядке.

Алгоритм дешифрования представлен на Риcунке 7.

Преобразование InvShiftRows.

Это преобразование обратно преобразованию ShiftRows. Первая строка формы остается неизменной. Вторая строка циклически сдвигается вправо на 1 байт. Третья – на 2, четвертая – на 3. Схематично преобразование показано на Риcунке 8.



Риcунок 8 – Процедура InvShiftRows

Преобразование InvSubBytes

Это преобразование обратно преобразованию SubBytes. Подстановка байтов происходит аналогично с помощью обратной таблицы подстановок, представленной в Таблице 4.

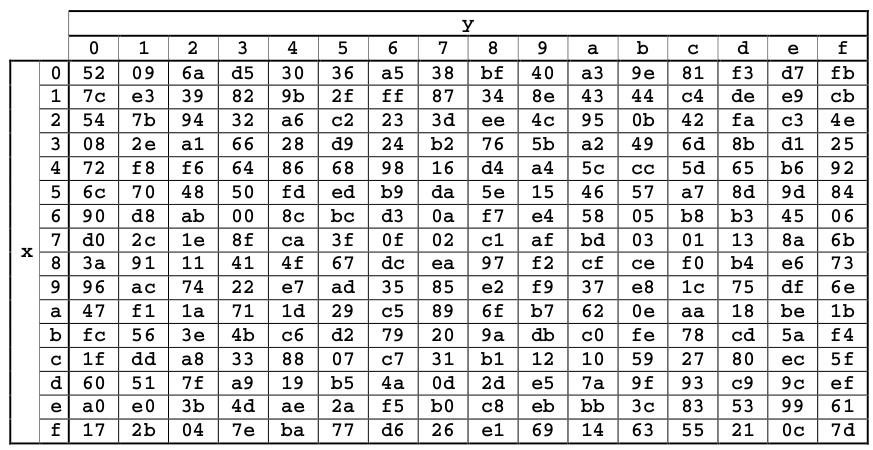


Таблица 4 – Обратная таблица подстановок

Преобразование InvMixColumns

Это преобразование обратно преобразованию MixColumns. InvMixColumns преобразует в форме каждый столбец отдельно.

Протокол B92

Протокол B92, предложенный в 1992 году, опирается на принцип неопределенности в квантовой механике. В этом протоколе информацию передают двухуровневые квантовые системы, известные как кубиты. Ключевой чертой этого метода является использование двух неортогональных состояний. Данный протокол является модификацией протокола BB84. В качестве базиса используются следующие поляризации:

Линейная: горизонтальная; вертикальная.

Круговая: правая круговая; левая круговая.

Исходя из принципа неопределенности Гейзенберга, невозможно с высокой достоверностью различить два неортогональных состояния кванта при измерении. Следовательно, определение значения бита становится проблематичным, и любые попытки измерить состояние кубита могут непредсказуемо его изменить. Протокол B92 представляет собой более простую в реализации схему из-за использования всего двух состояний для кодирования, но обеспечение должной надежности при таком протоколе представляет сложную задачу. Существует риск, что протокол может оказаться небезопасным. Ниже будет разобран алгоритм работы B92.

Протокол работы описывается следующим образом: Алиса передает последовательность импульсов с фемтосекундными интервалами и частотой повторения 80 МГц. Каждый импульс, в идеале, содержит один поляризованный фотон, причем нули кодируются в H-поляризованные фотоны, а единицы в V-поляризованные фотоны (красные стрелки на рисунке 9). Однако в половине случаев биты выбираются случайным образом и кодируются в диагональную поляризацию (D и A) (синие стрелки на рисунке 9). Приемник Боб измеряет поляризацию с использованием стандартной установки, способной различать H и V в базисе HV (обозначается как "+"), но иногда меняет базис на AD (обозначается как "X") .

После передачи определенного количества битов, Боб объявляет публично, какой базис использовал для каждого бита. Алиса затем сообщает, в каких случаях они использовали одинаковые базисы. Биты, где использовали разные базисы, отбрасываются, и оставшийся ключ уменьшается в два раза, оставаясь случайным и совпадающим у Алисы и Боба.

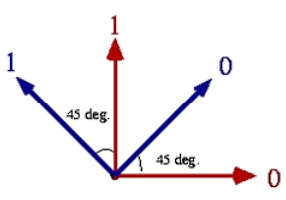


Рисунок 9 – Последовательность импульсов

Далее проводится проверка на подслушивание, сравнивая часть ключа (например, 10%). Если выявлено подслушивание, весь ключ отбрасывается, и процедура повторяется .

Таким образом, ключевым вопросом остается, как создавать одиночные фотоны. Одним из методов является использование света, излучаемого отдельными атомами, молекулами, центрами окраски алмаза или квантовыми точками. Однако этот подход требует значительных усилий и не обеспечивает "фотоны по требованию". Использование таких источников света все еще остается вероятностным, учитывая эффективность выбросов и сбора. На практике до сих пор применяются слабые когерентные состояния, характеризующиеся средним числом фотонов в импульсе около 0,2-0,3.

**7 Выбор инструментов для разработки проекта**

Для выбора языка программирования в рамках реализации программного комплекса рассматривались Python, Java, C++ и R, каждый из которых обладает своими уникальными возможностями. Однако, было принято решение разрабатывать программный продукт на языке Python, исходя из следующих соображений:

Простота синтаксиса: Python известен своим понятным и простым синтаксисом, что упрощает создание программного кода, особенно для команды разработчиков.

Обширная поддержка сред разработки: Язык обладает разнообразием сред разработки, обеспечивая удобную среду для написания и отладки кода, что способствует повышению эффективности работы.

Читаемость и структурированность кода: Python предлагает читаемый и легко структурируемый код, что облегчает его поддержку, модификацию и добавление новых функций.

Портабельность и кроссплатформенность: Python обеспечивает возможность запуска программы на различных платформах без необходимости изменения кода, а также гарантирует работоспособность на различных операционных системах.

Широкий выбор библиотек: Язык имеет обширное сообщество и множество библиотек, предоставляющих готовые решения и инструменты для различных задач, что ускоряет разработку и упрощает реализацию функциональности.

Поддержка сообщества разработчиков: Python имеет активное сообщество разработчиков, предоставляющее поддержку, ресурсы и обширный опыт, что значительно облегчает процесс разработки и решение возможных проблем.

Лидерство в рейтингах популярности: Согласно индексу TIOBE, отслеживающему рейтинг популярности языков программирования, Python в настоящее время занимает лидирующую позицию, что подчеркивает его широкое распространение и признание в сообществе разработчиков.

Начало формы

На графике, представленном на рисунке 10, показана динамика изменения рейтинга языков программирования.



Рисунок 10 – Рейтинг языков программирования

В качестве среды разработки использовался PyCharm 2023.3 PyCharm является многофункциональной программой для разработки программного обеспечения.

Это решение было принято по ряду следующих причин:

PyCharm является одной из наиболее популярных и мощных сред разработки для языка программирования Python, также она обладает широкими возможностями по работе с Python, включая подсветку синтаксиса, автозавершение кода, быстрый переход к определению и другие функции, которые значительно упрощают процесс написания и отладки кода на Python;

PyCharm обеспечивает интеграцию с популярными библиотеками машинного обучения, такими как NumPy, pandas, scikit-learn и TensorFlow, что позволяет удобно использовать эти библиотеки для обработки данных, создания моделей машинного обучения и оценки их производительности;

PyCharm предоставляет мощный и удобный редактор кода, который помогает разработчикам повысить производительность и качество своего кода, также редактор поддерживает функцию автодополнения кода, что значительно ускоряет процесс написания кода и предлагает функцию быстрого перехода к определению, что упрощает навигацию по проекту;

PyCharm имеет интеграцию с популярными системами контроля версий, такими как Git, что позволяет команде разработчиков легко отслеживать изменения, совместно работать над проектом и управлять версиями кода;

PyCharm предлагает множество полезных функций и инструментов, таких как отладчик, профилировщик, автоматическое форматирование кода, управление зависимостями и другие, кроме того, существует широкое сообщество разработчиков, которое поддерживает PyCharm, предоставляет ресурсы и делится опытом, что облегчает процесс разработки и решение возникающих проблем.

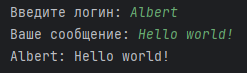
Для корректной работы нужно сначала импортировать библиотеки. Ниже представлено описание основных из них:

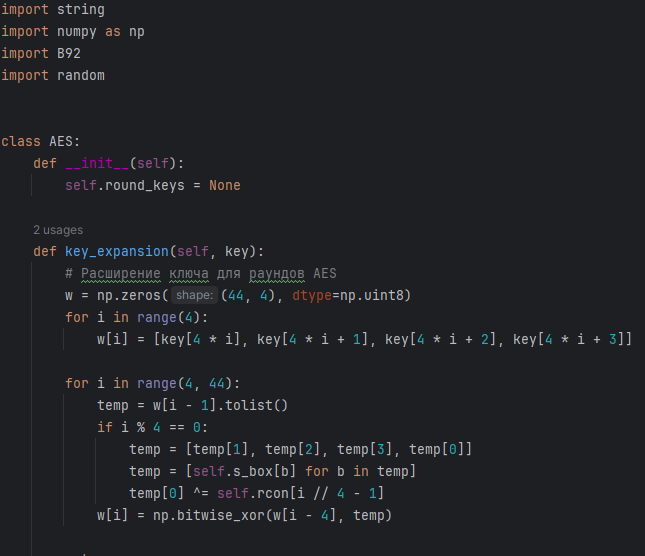
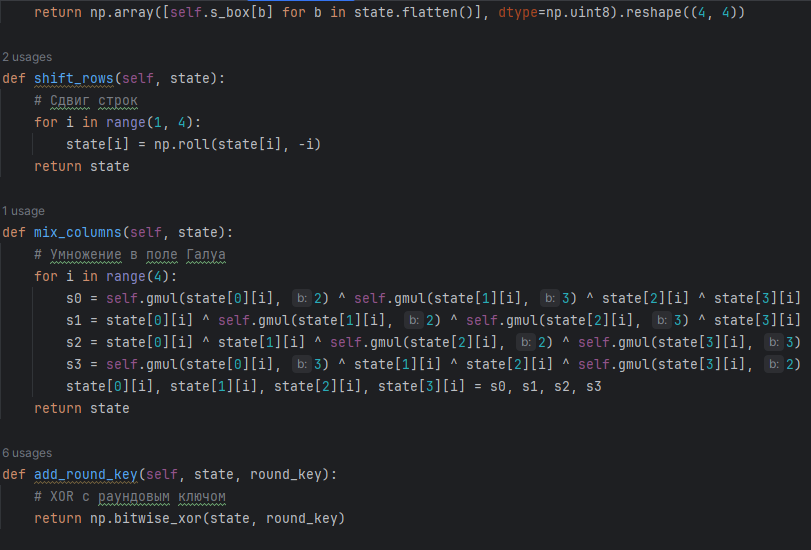
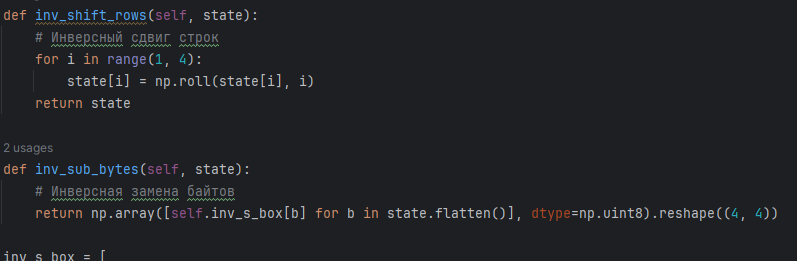
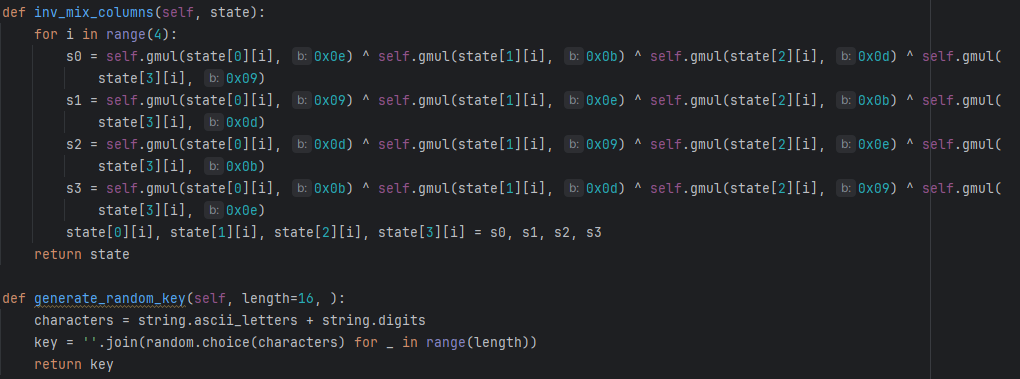
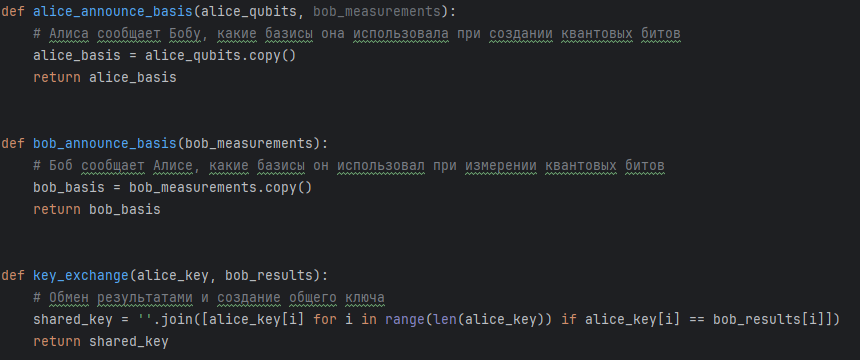
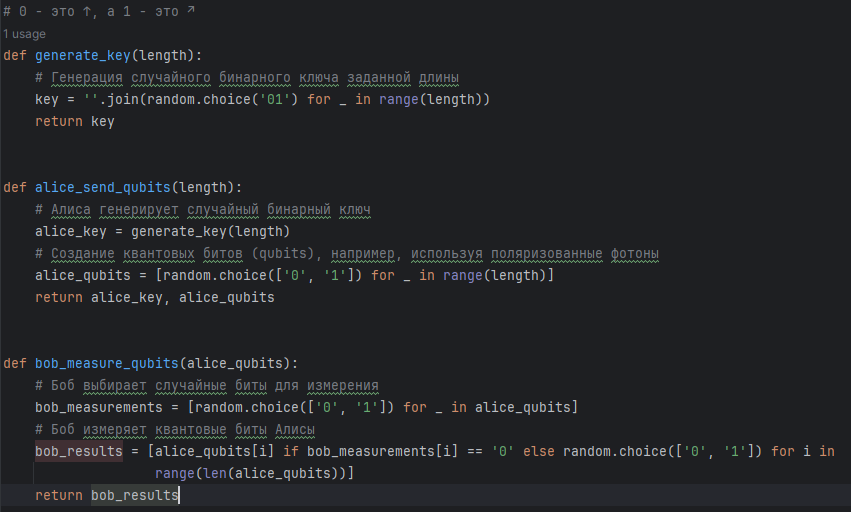
random – это встроенная библиотека в языке программирования Python, предоставляющая функциональность для работы с случайными числами. Эта библиотека используется для генерации случайных данных, чисел, перемешивания последовательностей и других операций, связанных с случайностью.

sqlite3 – это встроенный модуль в языке программирования Python, предоставляющий интерфейс для взаимодействия с базой данных SQLite. SQLite – это легковесная встроенная база данных, которая не требует отдельного сервера и хранится в виде одного файла на диске. Модуль sqlite3 позволяет создавать, подключаться к базам данных SQLite, выполнять SQL-запросы и манипулировать данными. Начало формы

**8 Обзор программы**

Сначала пользователь вводит свой логин, затем он вводит сообщение. После нажатия на Enter формируется ключ протокола b92. При этом квантовый ключ шифруется алгоритмом AES, а сообщение с помощью операции xor шифруется ключом, полученным в ходе работы протокола b92. При получении нового сообщения на экран выводиться логин отправителя и текст сообщения.



Таким образом, был реализован квантовый криптографический метод шифрования с использованием алгоритмов B92 и AES.

**9 Заключение**

В результате выполнения проекта :

1. Принцип работы и структура алгоритмов,…..добавить

2. История создания алгоритма AES и протокола,

3. Методы проверки алгоритма на соответствие условиям конкурса AES,

4. Методы программной реализации алгоритма AES и B92,

5. Основы работы с фрейворками PyQT, Flask.

Помимо этого, был получен ценный практический навык работы с языком программирования Python. Проведена разработка программы и тестирование ее. Представленная программа может применяться в любой сфере деятельности, где необходимо произвести шифрование текста или текстового файла с использованием модели криптосистемы на основе алгоритма AES и B92. Проект отвечает всем целям и задачам, заявленным в начале работы.

**10 Список источников и литературы**

https://ru.wikipedia.org/wiki/AES\_(%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82\_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)

<https://ru.wikipedia.org/wiki/B92#:~:text=%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%20B92%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%20%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D0%B5,%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B4%D0%B2%D1%83%D1%85%20%D0%BD%D0%B5%D0%BE%D1%80%D1%82%D0%BE%D0%B3%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D1%8B%D1%85%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B9>

<https://numpy.org/>

<https://en.wikipedia.org/wiki/NumPy>

<https://pythonworld.ru/numpy>

<https://flask.palletsprojects.com/en/3.0.x/>

https://docs-python.ru/packages/veb-frejmvork-flask-python/klass-request/

https://python-socketio.readthedocs.io/en/stable/

https://socket.io/docs/v4/

https://docs.python.org/3/library/threading.html

https://docs-python.ru/standart-library/modul-threading-python/klass-timer-modulja-threading/

https://superfastpython.com/timer-thread-in-python/

https://docs.python.org/3/library/sqlite3.html

https://habr.com/ru/articles/754400/

https://pythonru.com/biblioteki/vvedenie-v-sqlite-python

https://metanit.com/python/database/1.1.php

https://docs.python.org/3/library/re.html

https://tproger.ru/translations/regular-expression-python

https://habr.com/ru/articles/349860/

https://docs.python.org/3/library/sys.html

https://pythonworld.ru/moduli/modul-sys.html

<https://docs.python.org/3/library/tkinter.messagebox.html>

<https://younglinux.info/tkinter/dialogbox>

<https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html>

<https://habr.com/ru/articles/212235/>

<https://habr.com/ru/articles/112733/>

<https://vc.ru/dev/656195-kak-rabotaet-aes-advanced-encryption-standard-obyasnenie-dlya-gumanitariev-tipa-menya>

<https://bigenc.ru/c/b92-protokol-kvantovoi-kriptografii-e066fa>