МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСИС»**

Институт ИКН

Кафедра инженерной кибернетики

Направление подготовки: 01.03.04 Прикладная математика

Квалификация (степень): бакалавр

Группа: БПМ-20-2

**ОТЧЕТ**

**ПО КУРСОВОЙ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

1. на тему:***Криптосистема proxy re-encryption на базе блокчейн для безопасной передачи конфиденциальных данных в финансовом секторе***

**VIII семестр**

**2023 – 2024 уч. год.**

**Студент** \_\_\_\_\_\_\_\_\_\_/Корнилов М. А./

подпись Фамилия И.О.

**Руководитель КНИР** \_\_\_\_\_\_\_\_\_\_/ к.т.н., доцент Тарханов И. А./

подпись должность, уч. степ. Фамилия И.О.

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Утвердил*:***

**Председатель комиссии** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

подпись Фамилия И.О.

**Москва 2024**

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 4](#_Toc166100030)

[ВВЕДЕНИЕ 5](#_Toc166100031)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ 6](#_Toc166100032)

[1.1 Существующие решения для безопасной передачи конфиденциальных данных 6](#_Toc166100033)

[1.1.1 Существующие методы защиты данных 6](#_Toc166100034)

[1.1.2 Основные принципы криптографии 8](#_Toc166100035)

[1.1.3 Анализ преимуществ и недостатков существующих методов 10](#_Toc166100036)

[1.2 Описание схемы работы proxy re-encryption 11](#_Toc166100037)

[1.2.1 Принцип работы схемы 11](#_Toc166100038)

[1.2.2 Существующие PRE схемы 13](#_Toc166100039)

[1.2.3 Анализ безопасности схемы 13](#_Toc166100040)

[1.2.4 Эффективность криптографической защиты 15](#_Toc166100041)

[1.2.5 Билинейные пары 17](#_Toc166100042)

[1.3 Смарт-контракты в децентрализованных приложениях 17](#_Toc166100043)

[1.4 Выводы из обзора 18](#_Toc166100044)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 19](#_Toc166100045)

[2.1 Содержательная постановка задачи 19](#_Toc166100046)

[2.2 Математическая постановка задачи 20](#_Toc166100047)

[2.3 Алгоритм перешифрования 26](#_Toc166100048)

[3 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ТЕСТИРОВАНИЕ 29](#_Toc166100049)

[3.1 Описание подхода к разработке и тестированию программного обеспечения 29](#_Toc166100050)

[3.2 Используемые программные средства 30](#_Toc166100051)

[3.3 Описание функциональности разработанного ПО 31](#_Toc166100052)

[3.4 Тестирование криптосистемы 32](#_Toc166100053)

[3.4.1 Описание методологии тестирования схемы proxy re-encryption 32](#_Toc166100054)

[3.4.2 Тестирование производительности криптосистем 34](#_Toc166100055)

[3.5 Оценка защищенности системы 35](#_Toc166100056)

[3.6 Анализ полученных результатов 41](#_Toc166100057)

[ЗАКЛЮЧЕНИЕ 42](#_Toc166100058)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44](#_Toc166100059)

[ПРИЛОЖЕНИЕ A 46](#_Toc166100060)

[ПРИЛОЖЕНИЕ Б 55](#_Toc166100061)

[ПРИЛОЖЕНИЕ В 56](#_Toc166100062)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяют следующие сокращения и обозначения:

|  |  |  |
| --- | --- | --- |
| AES | – | advanced encryption standard |
| CA | – | certificate authority |
| CB-PRE | – | certificate based proxy re-encryption |
| CCA | – | chosen-ciphertext attack |
| CPA | – | chosen-plaintext attack |
| EC | – | elliptic curve |
| ECQV | – | elliptic curve Qu Vanstone |
| IOT | – | internet of things |
| PRE | – | proxy re-encryption |
| RSA | – | аббревиатура от фамилий Rivest, Shamir и Adleman |
| SCA | – | side-channel attack |
| SSH | – | secure shell |
| TLS | – | transport layer security |

# ВВЕДЕНИЕ

В современном мире передача конфиденциальной информации является критически важной задачей из-за возрастающего из года в год числа кибератак. Это особенно важно в финансовом секторе, где необходимо соблюдать строгие нормативные требования к защите информации без риска компрометации данных в процессе их передачи. В связи с этим существует потребность в разработке надежных и безопасных методов передачи данных. Схема proxy re-encryption (PRE) является одним из таких методов, который позволяет передавать зашифрованные данные от отправителя к получателю через надежного посредника, который называется прокси-сервером (proxy-server).

Данная курсовая работа посвящена разработке алгоритмического и программного обеспечения для создания криптосистемы proxy re-encryption на базе блокчейна для безопасной передачи конфиденциальных данных в реальных отраслях экономики, например, аудите, лицензировании или финансовом секторе.

Работа будет состоять из следующих разделов: введение в схему proxy Re-Encryption, описание CB-PRE схемы и всех ее компонентов, анализ применимости для передачи конфиденциальных данных, реализация схемы на основе выбранного языка программирования, тестирование разработанной системы и обсуждение результатов.

Целью данной курсовой работы является изучение схемы proxy re-encryption, анализ её защищенности и разработка прототипа системы для передачи конфиденциальных данных с использованием выбранного языка программирования. Ожидается, что разработанная система будет эффективно работать на практике и обеспечивать высокий уровень безопасности для передаваемых данных.

# АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ

## Существующие решения для безопасной передачи конфиденциальных данных

### Существующие методы защиты данных

Существует множество методов и технологий защиты данных, начиная от простых методов шифрования и заканчивая сложными системами управления доступом и многофакторной аутентификации. В данной работе мы рассмотрим несколько основных методов, которые широко используются для защиты конфиденциальных данных.

Шифрование данных: один из наиболее распространенных методов защиты данных, который заключается в преобразовании информации таким образом, чтобы только тот, у кого есть ключ, мог ее расшифровать. Существует множество алгоритмов шифрования, таких как AES, RSA, DES и другие [1].

Сетевые протоколы безопасности: используются для защиты данных, передаваемых по сети, от несанкционированного доступа. Примеры таких протоколов включают SSL/TLS, SSH, IPSec и другие [2].

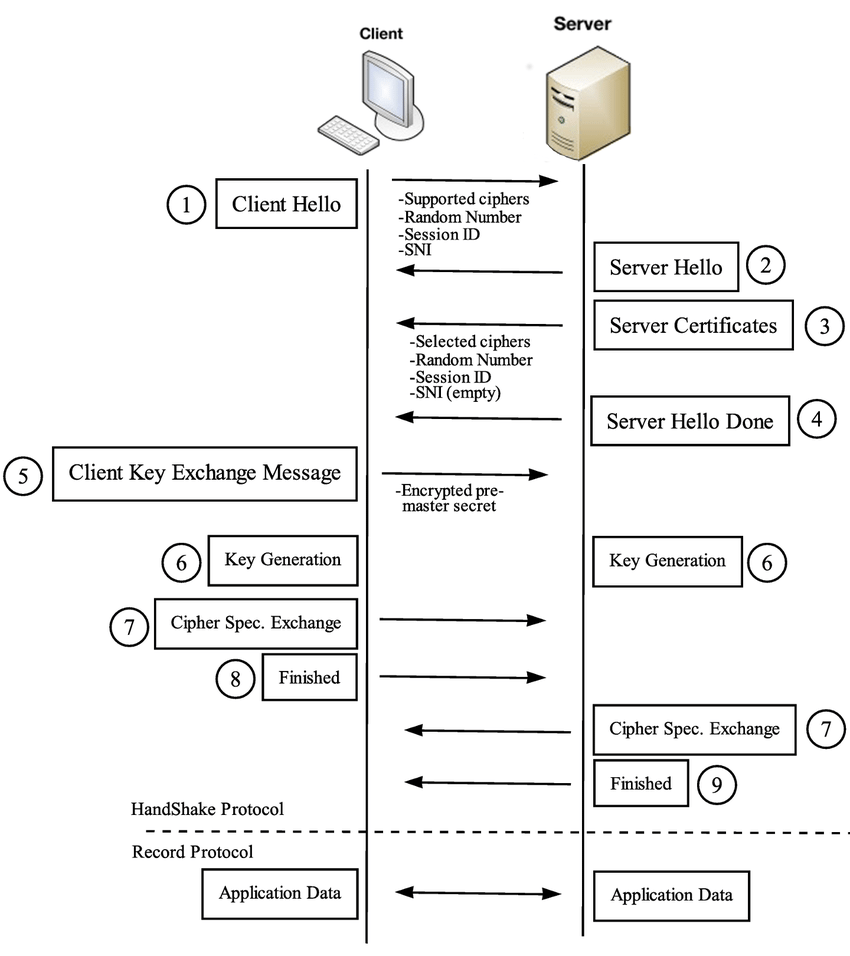


Рисунок 1 – Схема tls соединения

Управление доступом: технологии, позволяющие определять, кто и как может получить доступ к конфиденциальным данным. Обычно включают авторизацию, аутентификацию и управление правами доступа.

Программные средства защиты информации: программные продукты, предназначенные для защиты данных, обнаружения и предотвращения вторжений, мониторинга безопасности и других задач. Примеры таких продуктов включают антивирусы, фаерволы, IDS/IPS и другие [3].

Схемы proxy re-encryption: специальный тип криптографических протоколов, который позволяет безопасно передавать зашифрованные данные между несколькими участниками, используя промежуточный сервер для перешифрования данных.

Токенизация данных: токенизация представляет собой процесс замены конфиденциальных данных уникальными токенами, которые не раскрывают исходные данные. Токены используются для обработки данных, при этом исходные данные сохраняются в безопасности. Этот метод позволяет уменьшить риски утечки конфиденциальной информации и повысить безопасность хранения данных.

Криптографические хеш-функции: Хеш-функциями называют алгоритм, который преобразует входное сообщение произвольной ограниченной длины, представленное в битах или байтах, в выходное значение фиксированного и сравнительно небольшого размера [4]. Они широко используются для обеспечения целостности данных и проверки их целостности. Криптографические хеш-функции обеспечивают невозможность обратного преобразования хэша в исходные данные, что делает их надежными для защиты данных.

Средства мониторинга и обнаружения инцидентов безопасности: Эти инструменты предназначены для постоянного мониторинга и обнаружения потенциальных инцидентов безопасности. Они позволяют своевременно выявлять аномалии, несанкционированный доступ или подозрительное поведение в системе, что помогает предотвратить и реагировать на угрозы безопасности.

Резервное копирование и восстановление данных: Регулярное создание резервных копий данных и разработка стратегии восстановления в случае потери или повреждения данных являются важными аспектами защиты данных. Этот метод обеспечивает возможность быстрого восстановления данных и минимизации потерь в случае сбоев или атак [5].

Физическая безопасность: Обеспечение физической безопасности серверных помещений, хранилищ данных и других физических ресурсов является важным аспектом защиты данных. Меры физической безопасности могут включать контроль доступа, видеонаблюдение, а также защиту от пожара и повреждений.

### Основные принципы криптографии

Криптография – это наука о защите информации, которая изучает методы обеспечения конфиденциальности, целостности и аутентичности данных. Ее основная задача - обеспечить безопасную передачу информации между двумя или более участниками без риска ее прослушивания или изменения [6].

Шифрование – это метод трансформации данных, понятных человеку, в закодированный и нечитаемый вид, известный как шифротекст. Его цель - защитить конфиденциальную информацию от несанкционированного доступа посторонних лиц [7].

Основными принципами криптографии являются:

* симметричное и асимметричное шифрование – это методы шифрования данных, которые обеспечивают защиту информации при ее передаче по сети. В симметричном шифровании для шифрования и расшифрования используется один и тот же ключ. В асимметричном шифровании используются два ключа: открытый и закрытый [1];
* хэширование – это процесс преобразования данных произвольной длины в уникальную фиксированную строку определенной длины. Хэширование используется для проверки целостности данных [1];
* цифровые подписи – это методы обеспечения аутентичности данных. Цифровая подпись позволяет установить, что сообщение было создано определенным отправителем, и что сообщение не было изменено в процессе передачи [8];
* протоколы аутентификации – это методы обеспечения подлинности пользователя. Протоколы аутентификации позволяют установить, что пользователь является тем, за кого себя выдает [9];
* протоколы обмена ключами – это методы обеспечения конфиденциальности данных. Протоколы обмена ключами позволяют двум участникам обменяться секретным ключом для дальнейшего использования в процессе шифрования.

Криптосистема – семейство выбираемых с помощью ключа обратимых преобразований, которые трансформируют защищаемый блок информации в шифрограмму и обратно [10].

В контексте разработки криптосистемы proxy re-encryption для безопасной передачи конфиденциальных данных основными принципами криптографии, которые будут использоваться, являются симметричное и асимметричное шифрование, протоколы обмена ключами и цифровые подписи.

### Анализ преимуществ и недостатков существующих методов

Для выбора наиболее подходящего метода защиты данных для конкретного случая необходимо проанализировать преимущества и недостатки каждого метода.

Симметричное шифрование, например, является быстрым и эффективным методом, однако у него есть недостатки. К примеру, в этом методе используется общий секретный ключ, который должен быть передан между отправителем и получателем, что может привести к его компрометации. Кроме того, управление доступом к данным в таком методе сложно реализовать.

Асимметричное шифрование, с другой стороны, имеет преимущества в том, что нет необходимости передавать общий секретный ключ, так как используются два различных ключа - публичный и приватный. Однако он требует большего времени и вычислительной мощности для работы, а также имеет проблемы с управлением доступом к данным [11].

Прокси-шифрование является методом, который позволяет повысить безопасность передачи данных между отправителем и получателем, не требуя от них знания ключа шифрования. Кроме того, он обеспечивает гибкое управление доступом к данным. Однако, такой метод также имеет свои недостатки, такие как наличие отдельного централизованного прокси сервера и дополнительная производительность, которая необходима для выполнения решифрования.

Таким образом, каждый метод имеет свои преимущества и недостатки, и выбор конкретного метода зависит от требований к безопасности, производительности и управлению доступом к данным в конкретной ситуации.

## Описание схемы работы proxy re-encryption

### Принцип работы схемы

Схема proxy re-encryption (прокси-перешифрование) является эффективным методом для безопасной передачи конфиденциальных данных между отправителем и получателем с использованием промежуточного прокси-сервера. Она обеспечивает конфиденциальность и защиту данных, позволяя изменять ключ шифрования без необходимости расшифровывать и повторно шифровать данные.

Функциональная схема работы proxy re-encryption представлена на рисунке 2.

A black screen with white text

Description automatically generated

Рисунок 2 – функциональная схема работы proxy re-encryption

Настройка (Setup): В начале работы центра сертификации проводится этап настройки, в ходе которого генерируются необходимые параметры, такие как приватные и публичные ключи, а также другие конфигурационные данные. Конфигурационные данные генерируются один раз при запуске центра сертификации, приватные ключи перегенерируются при необходимости [12].

Сертификация пользователей (CertifiedUserKeyGen): перед использованием схемы, каждый пользователь должен пройти процесс сертификации, в результате которого ему выдаются сертификаты. Эти сертификаты подтверждают идентификацию и аутентичность пользователей, а также содержат необходимую информацию для работы схемы [12].

Шифрование (Encrypt): отправитель использует свой приватный ключ для шифрования данных перед отправкой и создает решифровочный ключ, который используется для перешифрования данных от отправителя к получателю [12]. Ключ регистрируется в смарт-контракте.

Перешифрование (ReEncrypt): прокси-сервер выполняет операцию перешифрования данных с использованием решифровочного ключа, который получен из смарт-контракта. В результате этой операции данные становятся доступными для расшифровки только для получателя, при этом прокси-сервер не расшифровывает исходные данные [12].

Расшифровка (Decrypt): получатель, имея свой приватный ключ, расшифровывает полученные данные. При этом получателю не требуется наличие других ключей для успешной расшифровки [12].

Схема proxy re-encryption позволяет достичь высокого уровня безопасности при передаче конфиденциальных данных, так как она обеспечивает конфиденциальность и контроль доступа, при этом минимизируя потребность в передаче приватных ключей между отправителем и получателем.

### Существующие PRE схемы

Существуют различные способы реализации передачи данных:

* односторонняя: позволяет делегировать доступ к шифрованным данным от одного пользователя к другому, но не обратно [13];
* двусторонняя: поддерживает возможность делегирования доступа между двумя пользователями в обе стороны;
* многоступенчатая: подразумевает возможность передачи доступа через цепочку пользователей;
* симметричная и асимметричная: в симметричном PRE используется один ключ для шифрования и расшифровки, в асимметричном — разные ключи для каждого пользователя [13];
* условная: позволяет решифрование только если выполнены определённые условия.

Также есть разные реализации самой схемы PRE:

* на основе решеток: этот тип схемы использует математическую структуру решёток, что обеспечивает устойчивость к атакам с использованием квантовых компьютеров [14].
* identity-based PRE: этот тип основывается на концепции идентификационной криптографии, где ключи генерируются из уникальных идентификаторов пользователей [15].
* certificate-based PRE: включает использование сертификатов для управления и аутентификации ключей в рамках системы прокси-решифрования [16].

### Анализ безопасности схемы

Анализ безопасности схемы proxy re-encryption включает в себя оценку ее сильных сторон и потенциальных уязвимостей. Ниже представлен анализ безопасности схемы:

1. конфиденциальность. Схема proxy re-encryption обеспечивает высокий уровень конфиденциальности данных. Шифрование данных перед их передачей от отправителя к прокси-серверу и от прокси-сервера к получателю гарантирует, что только авторизованные пользователи смогут получить доступ к расшифрованным данным. Алгоритмы, которые описаны в ReKeyGen и ReEncrypt, обеспечивают безопасное преобразование и перешифрование данных, минимизируя риски утечки информации;
2. целостность. Схема прокси решифрования обеспечивает целостность данных, поскольку шифрование и расшифрование данных выполняются с использованием ключей и алгоритмов, которые гарантируют, что данные не изменяются неправомерно или без уведомления сторон;
3. аутентификация. Сертификационный процесс, описанный в алгоритме CertifiedUserKeyGen, позволяет установить доверие к публичным ключам пользователей, что гарантирует, что данные будут переданы только тем, кто имеет соответствующие авторизационные данные;
4. устойчивость к атакам. Криптосистема разработана с учетом различных типов атак, таких как перехват, подмена и подслушивание данных. Использование криптографических алгоритмов, включая эллиптические кривые [17], повышает уровень безопасности и усложняет возможность проведения успешных атак;
5. надежность ключей. Безопасность схемы прокси решифрования зависит от надежности и безопасности используемых ключей. Важно обеспечить защиту приватных ключей пользователей и прокси-серверов, чтобы предотвратить их компрометацию. Это может включать использование сильных алгоритмов генерации ключей, хранение ключей в безопасной среде и управление доступом к ключам с помощью соответствующих политик и процедур;
6. мониторинг и обнаружение атак. Важным аспектом обеспечения безопасности является мониторинг и обнаружение атак. Регулярный мониторинг сетевого трафика, анализ логов и использование систем обнаружения вторжений (IDS) [18] и систем защиты от вторжений (IPS) помогут обнаружить аномальную активность и потенциальные попытки нарушения безопасности системы;
7. регулярные проверки на уязвимости. Для поддержания высокого уровня безопасности необходимо проводить регулярные проверки на уязвимости схемы proxy re-encryption. Это может включать проведение пенетрационного тестирования, аудита кода, анализа уязвимостей и других методов оценки безопасности. Выявление и устранение потенциальных уязвимостей поможет повысить стойкость системы к возможным атакам [19].

Однако, как и любая криптографическая схема, схема proxy re-encryption имеет свои потенциальные уязвимости. Например, уязвимости могут возникнуть в результате компрометации приватных ключей пользователей или прокси-серверов, а также из-за ошибок в реализации алгоритмов. Поэтому важно применять известные практики безопасности, такие как защита приватных ключей и регулярные проверки на уязвимости системы.

### Эффективность криптографической защиты

Оценка эффективности защиты информации с помощью криптографии включает в себя выявление всех возможных нападений, которым может подвергнуться криптографическая система в конкретных условиях её использования, и установление её способности сопротивляться таким атакам. Под прочностью системы мы будем подразумевать её криптографическую надежность, то есть её способность выдерживать нападения криптоаналитиков.

Разработка методов определения криптографической надежности представляет собой сложную задачу, требующую комплекс инструментальных средств, которые позволят оценивать действия злоумышленника в попытках взлома системы с использованием разнообразных техник криптоанализа. Детальное описание программного комплекса, предназначенного для анализа устойчивости асимметричных криптосистем с применением математических методов, выходит за рамки данной статьи, но его можно найти в других источниках.

Для точного определения эффективности криптосистемы целесообразно проверять её на устойчивость не ко всем без исключения атакам, а лишь к тем, которые наиболее вероятны и представляют собой наибольшую опасность [20]. Набор таких атак определяется в зависимости от типа криптосистемы и специфики её применения.

Процесс оценки эффективности криптографической защиты можно представить в виде схемы в соответствии с рисунком 3 [21].

A diagram of a diagram

Description automatically generated

Рисунок 3 – Процесс оценки эффективности системы

Этап 1. Какая криптосистема является объектом атаки?

Этап 2. Кто будет атаковать эту криптосистему?

Этап 3. Какие методы криптоанализа с наибольшей вероятностью будут использованы при осуществлении попыток взлома криптосистемы?

Этап 4. Способна ли криптосистема противостоять таким атакам?

Этап 5. Является ли использование исследуемой криптосистемы экономически выгодном в данном контексте

Этапы 1-3 представляют моделирование угроз для информационных ресурсов, которые защищает исследуемая криптосистема.

Этап 4 – это анализ устойчивости системы в атаках определенного типа, которые были определены на предыдущих этапах. Цель заключается в оценке риска нарушения безопасности.

Этап 5 – использование различных подходов оценки экономической эффективности использования выбранной криптосистемы.

### Билинейные пары

Для осуществления перешифрования данных от одного пользователя другому используется алгоритм билинейных пар: отображения между двумя группами и в мультипликативную группу , все три с первичным порядком . Пара описывается как функция [22].

Основные свойства билинейных пар:

* билинейность;
* невырожденность.

Эти свойства делают их эффективным инструментов криптографии.

## Смарт-контракты в децентрализованных приложениях

Смарт-контракт (smart contract, умный контракт) — это компьютерный протокол, предназначенный для автоматизации исполнения условий и соглашений в рамках децентрализованных систем, таких как блокчейн [23].

По сути, это компьютерная программа, которая может отслеживать и обеспечивать исполнение обязательств при сделках. Смарт-контракты исполняют следующие основные принципы:

* автоматизация;
* безопасность;
* надежность;
* прозрачность.

Децентрализованное приложение (dApp) – это приложение, которое работающее на децентрализованной вычислительной системе, обычно на блокчейне, и имеющих открытый исходный код. Ни один узел сети не имеет полного контроля над dApp [24].

Разработка децентрализованных приложений часто включает в себя создание смарт-контрактов, которые определяют условия обработки и исполнения данных, автоматически проверяя исполнение договора и упрощая процесс взаимодействия с приложениями. Все изменения, которые связаны с приложением, можно легко отслеживать благодаря прозрачности смарт-контрактов.

## Выводы из обзора

В данной работе в целях повышения безопасности, гибкости управления доступа к данным, удобства передачи данных между участниками сети, а также простоты реализации, была выбрана двухсторонняя ассиметричная PRE схема. Также в криптосистему PRE на базе блокчейна можно интегрировать смарт-контракт, в котором будут храниться ключи решифрования, что снизит риски компрометации всех ключей одновременно и повысит аудитируемость системы.

Для сравнение разрабатываемой криптосистемы с классической (наиболее распространенной асимметричной RSA) будет использоваться метод оценки защищенности, описанный в п. 1.2.4.

# СПЕЦИАЛЬНАЯ ЧАСТЬ

## Содержательная постановка задачи

**Цель работы**

Разработка и анализ оптимизированной схемы прокси-решифрования с использованием смарт-контрактов, обеспечивающих повышение эффективности и безопасности доступа к зашифрованным данным.

**Предметная область**

Информационная безопасность, криптография.

В качестве входных данных выступают:

1. клиент, сервер(получатель), прокси-сервер, центр сертификации;
2. конфиденциальная информация;
3. адрес блокчейна;
4. адрес смарт-контракта, который зарегистрирован на блокчейне.

В качестве результата должна быть разработана и проведена оценка защищенности схемы прокси-решифрования, интегрированная с смарт-контрактами, обеспечивающая безопасную передачу и доступ к конфиденциальным данным.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. создание алгоритма прокси решифрования на python;
2. разработка смарт-контракта на solidity;
3. объединение прокси-решифрования и смарт-контракта в единую систему;
4. генерация центром сертификации приватных ключей;
5. проверка работоспособности системы;
6. оценка защищенности PRE и RSA;
7. сравнение результатов оценки защищенности;
8. тесты производительности разработанной схемы.

## Математическая постановка задачи

Чтобы определить комплекс угроз для криптосистемы, разработаем математическую модель, которая оценивает безопасность защиты информации. Модель базируется на таких исходных данных:

* возможно, что один хакер будет использовать множество разнообразных методов атаки, также как один и тот же тип атаки может быть осуществлён различными хакерами;
* разнообразные атаки могут быть направлены против одной криптосистемы, и одинаковые атаки могут нарушить безопасность разных криптосистем;
* атакующий, скорее всего, предпочтёт тот метод атаки, который при фиксированном уровне ресурсов даст ему наилучший итог или который окажется наиболее экономичным среди различных опций, ведущих к схожему результату.

Пусть – множество параметрических моделей атак, где – множество значений j-го параметра модели атаки. Тип атаки определяется в соответствии с критериями классификации [25]:

* - по доступу к открытому и зашифрованному тексту;
* - по контролю над процессом шифрования;
* - по объему необходимых ресурсов;
* - по степени применимости к различным шифрам;
* - по используемым средствам;
* - по последствиям атаки;
* - по возможности распарллеливания.

Каждая модель преставляет собой вектор , где

Мощность множества моделей атак

Пусть – множество параметрических моделей злоумышленников, где ) – множество значений j-го параметра модели злоумышленника. Тип злоумышленника определяется в соответствии с критериями классификации [25]:

* - по технической оснащенности;
* - по конечной цели;
* - по доступу к шифрующим средствам
* - по уровню подготовки;
* - по первичной информации о средстве шифрования;
* - по возможности кооперации.

Каждая модель преставляет собой вектор , где

Мощность множества моделей злоумышленников

Пусть – множество параметрических моделей криптосистем, где ) – множество значений j-го параметра модели криптосистемы. Тип криптосистемы определяется в соответствии с критериями классификации [25]:

* - по доступности информации о криптоалгоритме;
* - по числу ключей;
* - по стойкости криптоалгоритма;
* - по используемым средствам шифрования;
* - по наличию сертификата;

Каждая модель представляет собой вектор , где

Мощность множества моделей криптосистем

Риск от атаки вычисляется на основе двух факторов – вероятности происшествия и тяжести возможных последствий

Функцию для оценки уровня риска обозначим через: . Она связана с атакой , когда она применена злоумышленником  для взлома криптосистемы .

Пусть – функция влияния, которая оценивает степень ущерба от применения атаки к криптосистеме .

Пусть – вероятность того, что злоумышленник предпримет атаку .

Таким образом, функция риска будет выглядеть как

Для определения функции рассмотрим семейство функций.

, , где , - множество неотрицательных действительных чисел. Функция задает уровень взаимного влияния параметра криптосистемы и параметра атаки :

* , если атака c параметром  не применима к криптосистеме со значением параметра ;
* , если значение параметра криптосистемы снижает вероятность успешного применения атаки со значением параметра ;
* , если значение параметра криптосистемы не влияет на применимость атаки с параметром ;
* , если значение параметра криптосистемы указывает на то, что атака с параметром применима для взлома.

Уровень взаимного влияния параметром криптосистемы и атаки определяется на основе экспертных оценок.

Пусть – нормированная функция

Значит ущерб от применения атаки к криптосистеме вычисляется по следующей формуле:

где атака и криптосистема заданы параметрами: и (.

Для определения функции рассмотрим семейство функций , , . Функция задает уровень взаимного влияния параметра злоумышленника и параметра атаки :

* *,* если злоумышленник со значением параметра  гарантированно не будет использовать атаку со значением параметра ;
* *,* если значение параметра злоумышленника  снижает вероятность использования атаки со значением параметра ;
* *,* если значение параметра злоумышленника не влияет на вероятность использования атаки со значением параметра ;
* *,* если злоумышленник со значением параметра с большой вероятностью будет использовать атаку со значением параметра .

Уровень взаимного влияния параметров злоумышленника и атаки также определяется экспертами.

Пусть  – нормированная функция

Тогда вероятность того, что злоумышленник предпримет атаку

где атака и злоумышленник заданы параметрами и соответственно.

Таким образом общая формула для определения уровня риска имеет вид:

Криптосистема подвержена атаке в условиях, когда ей угрожает злоумышленник , если , где  – заданное пороговое значение. Допустимый уровень риска настраиваемый параметр, который задается с учетом критериев:

* критичность защищаемых данных;
* ресурсы, доступные специалисту для осуществления проверки защищенности системы.

В общем контексте криптосистема может включать в себя несколько подсистем, таких как генератор ключей и симметричный шифратор, каждая из которых подвержена своему собственному спектру возможных атак. Криптосистема может быть также объектом нападения различных злоумышленников.

В описанной математической модели сделаны следующие допущения:

* не учитывается зависимость параметров атаки от сочетания параметров криптосистемы;
* не учитывается возможность совместных действий со стороны взломщиков различных типов.

Учёт этих изменений в модели сделает её более сложной. Остаётся открытым вопрос, сказываются ли эти изменения на точности предсказаний модели по поводу угроз безопасности, этот аспект требует дополнительного анализа.

## Алгоритм перешифрования

* SetUp(l). При заданном определенном параметре безопасности l будут выполнены следующие шаги для получения общедоступных параметров params и главного секретного ключа msk:

1. сначала центр сертификации выбирает l-разрядное простое число q. Далее генерируется EC порядка q и определяется соответствующая точка генератора P. Обозначим через G группу точек EC;
2. выбирается случайное значение и вычисляется ;
3. определяются четыре различные хэш-функции;
4. общедоступные параметры params теперь , секретный ключ ;

* CertifiedUserKeyGen(). Этот алгоритм основан на механизме сертификации эллиптической кривой Кванстоуна (ECQV) и состоит из следующих трех этапов:

1. генерирует случайное значение и вычисляется , далее кортеж отправляется в CA;
2. по прибытию CA проверяется идентификатор . Далее он выбирает случайное значение и вычисляет и сертификат получен. Вспомогательная информация для вычисления приватного ключа для задействованного клиента вычисляется по формуле

Кортеж отправляется обратно;

1. клиент сначала получает свой закрытый ключ . Его публичный ключ эквивалентен . Если , то он принимает пару ключей .

* Encrypt(:

Метадата генерируется для сообщения M,

Дальше производятся следующие вычисления

На выходе алгоритма получаем .

* ReKey. получено из C. Публичный ключ вычисляется как . Это приводит к вычислению ключа решифрования как:
* ReEncrypt. Повторное шифрование изменяет зашифрованные текст от в зашифрованные текст :

Если подставить , то получаем, что

В результате работы функции получаем

* Decrypt1. Для расшифровки оригинального сообщения и проверки его подлинности требуются следующие вычисления:

Проверка:

* Decrypt2На этом этапе получатель извлекает сообщение из с помощью следующих операций:

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ТЕСТИРОВАНИЕ

## Описание подхода к разработке и тестированию программного обеспечения

* выбор инструментов разработки:

1. для разработки и запуска в тестовой среде смарт контрактов, которые будут использоваться в схеме Proxy re-encryption, будет использована онлайн-среда Remix Ethereum. Этот инструмент обладает мощными возможностями разработки, отладки и развертывания смарт контрактов на блокчейне Ethereum;
2. для создания локальной сети блокчейна Ethereum, в которой будут тестироваться и разрабатываться смарт контракты, будет использоваться инструмент Ethereum Ganache, который предоставляет среду для развертывания локального блокчейна Ethereum, что позволяет тестировать и отлаживать смарт контракты в изолированной среде без необходимости подключения к основной блокчейн-сети;

* разработка смарт контрактов:

1. среда Remix Ethereum хорошо подходит для создания и разработки смарт контракты на языке Solidity. В ней необходимо реализовать алгоритмы и протоколы, описанные в предыдущих разделах работы, внутри смарт контрактов;
2. Remix Ethereum предоставляет мощные инструменты для проверки синтаксиса, отладки и тестирования смарт контрактов;

* запуск смарт контрактов:

1. после разработки и отладки смарт контрактов в среде Remix Ethereum, мы можем использовать Ganache для развертывания смарт контрактов на локальной сети блокчейна Ethereum;
2. Ganache предоставляет нам локальный блокчейн с эмулированными аккаунтами и средствами для развертывания контрактов. Мы можем использовать его для проверки работоспособности наших смарт контрактов в изолированной среде;

* тестирование и отладка:

1. после запуска смарт контрактов на локальной сети блокчейна с помощью Ganache, необходимо провести тестирование и отладку наших контрактов;
2. Remix Ethereum для отправки транзакций и вызова функций наших смарт контрактов, а также для проверки возвращаемых результатов;
3. описанный подход позволит разработать и тестировать смарт контракты для схемы Proxy re-encryption с помощью Remix Ethereum и локального блокчейна Ethereum с использованием Ganache.

## Используемые программные средства

Для реализации использованы языки программирования Python 3 и Solidity.

Среда программирования – PyCharm IDE, Remix IDE. Для поднятия локального блокчейна используется Ganache.

Были использованы следующие библиотеки, которые не входят в стандартные библиотеки python, и фреймворки:

* сharm-crypto;
* web3;
* cryptography.

Charm-crypto – этот фреймворк предназначен для ускоренной разработки прототипов сложных криптосистем. Разработка велась на Python с целью упрощения кода и сокращения времени разработки, при этом платформа поддерживает повторное использование компонентов. Charm применяет гибридный подход, где вычислительно интенсивные математические операции выполнены в нативных модулях C, в то время как сами криптосистемы реализованы на высокоуровневом и легко читаемом языке. Фреймворк также включает новые компоненты, которые упрощают разработку новых криптографических схем и протоколов.

Web3 – эта библиотека Python используется для взаимодействия с Ethereum и часто применяется в децентрализованных приложениях (dapps) для помощи в отправке транзакций, управлении смарт-контрактами, чтении данных блокчейна и многом другом.

Cryptography – это библиотека, предоставляющая разработчикам на Python криптографические инструменты и примитивы. Она стремится быть «стандартной криптографической библиотекой», предлагая разнообразные криптографические решения для разработчиков.

Для построения полноценной схеме было поднято 4 виртуальные машины на платформе Yandex Cloud, каждая со своим “белым” ip-адресом.

## Описание функциональности разработанного ПО

Проект состоит из 7 файлов расширения (.py):

1. ca.py. Центр сертификации, который генерирует секретный ключ, ключи участников сети и публичные параметры;
2. sender.py. Участник сети, которые отправляет данные;
3. receiver.py. Получатель зашифрованных данных;
4. proxy.py. Прокси сервер, которые выполняет функции решифрования данных и их передачи получателю;
5. smart\_contract.py. Описание ABI смарт-контракта, поддерживаемых им методов, функции для взаимодействия с ним;
6. constants.py. Общие константы (адрес смарт-контракта, адрес прокси и так далее);
7. util.py. Вспомогательные функции для работы с данными.

Так же в проекте есть 1 файл с расширение (.sol) – исходный код смарт-контракта. Смарт-контракт используется для хранения решифровочных ключей, которые по запросу получается прокси-сервер.

## Тестирование криптосистемы

### Описание методологии тестирования схемы proxy re-encryption

Юнит-тестирование.

Юнит-тестирование смарт контрактов проводится с использованием фреймворка Truffle. Мы создаем тестовые сценарии для проверки отдельных функций и компонентов смарт контрактов. Целью этого тестирования является уверенность в корректной работе отдельных частей смарт контрактов, таких как функции шифрования, расшифровки, генерации ключей, а также обработки данных [26], схема тестирования приведена на рисунке 4.

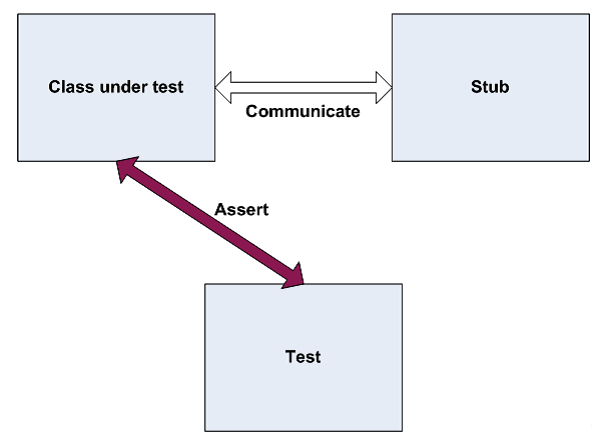


Рисунок 4 – Схема юнит тестирования

Интеграционное тестирование.

Интеграционное тестирование направлено на проверку взаимодействия между различными компонентами системы и правильности работы смарт контрактов в реальных сценариях использования. Мы создаем тестовые сценарии, которые охватывают отправку и получение зашифрованных данных, выполнение операций Proxy re-encryption и проверку правильности результатов [27]. Целью тестирования является убедиться, что смарт контракты взаимодействуют корректно и обеспечивают безопасную передачу конфиденциальных данных, схема тестирования приведена на рисунке 4.

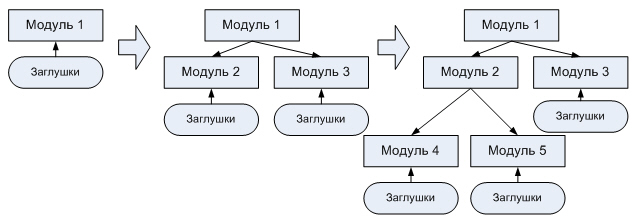


Рисунок 5 – Схема интеграционного тестирования

Функциональное тестирование.

Функциональное тестирование направлено на проверку функциональности схемы Proxy re-encryption в целом. Мы разрабатываем тестовые сценарии, которые охватывают основные функции и возможности системы. Например, мы проверяем правильность шифрования и расшифровки данных с использованием различных ключей, проверяем правильность перешифрования данных и работу алгоритмов Proxy re-encryption. Целью тестирования является убедиться, что система работает согласно спецификации и выполняет свои функции корректно.

Нагрузочное тестирование.

Нагрузочное тестирование проводится для оценки производительности и масштабируемости системы в условиях высокой нагрузки. Мы используем инструменты, такие как Truffle или другие фреймворки для нагрузочного тестирования блокчейн-систем. Целью этого тестирования является проверка работоспособности системы при высокой нагрузке и определение ее производительности [25].

Анализ кода на статичные уязвимости.

Анализ, который учитывает внутренние компоненты приложений, противоположный тестированию по принципу “черный ящик”, структура всех частей приложения известна. Такой анализ проводится с целью выявления уязвимостей в работающем приложении. Существует несколько различных методов: лексический, синтаксический, семантический, taint-анализ, распространение типов, распространение констант и исследование графов потока управления.

Такой род анализа может проводиться, как и специалистами, так и при помощи инструментов-анализаторов, которые можно внедрить непосредственно в код приложения.

Помимо перечисленных видов тестирования, важным этапом в разработке и обеспечении безопасности схемы proxy re-encryption является аудит смарт контрактов. Аудит выполняется независимыми специалистами с целью выявления потенциальных уязвимостей и ошибок в коде смарт контрактов. Аудит помогает обеспечить безопасность системы и предотвратить возможные атаки или нарушения конфиденциальности данных.

Все эти виды тестирования и аудит смарт контрактов играют важную роль в обеспечении качества и надежности программного обеспечения для схемы proxy re-encryption. Они помогают выявить потенциальные проблемы и ошибки, а также обеспечить корректную работу системы в различных сценариях использования.

## Тестирование производительности криптосистем

Для сравнения производительности PRE и классической криптосистемы с использованием ассиметричного RSA шифрования было проведено по 100 запусков программ, получившиеся результаты представлены в таблице 1. Значения в секундах и округлены до 4 знаков после запятой.

Таблица 1 – производительность криптосистем

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | генерация  ключей | шифрование  данных | решифрование  данных | расшифрование  данных | смарт-контракт |
| RSA | 0.0989 | 0.0003 | - | 0.0005 | - |
| PRE | 0.0083 | 0.0113 | 0.0185 | 0.0094 | 0.0637 |

В графе “решифрование данных” подсчитывалось время выполнения всех операций, связанных с решифровочным ключом (его создание и решифрование данных), аналогично посчитаны все операции с смарт-контрактом.

Таким образом, общее среднее время выполнения операций для криптосистем:

* для RSA: 0.0997;
* для PRE: 0.1112.

PRE криптосистема медленнее на 11.5% чем классическая криптосистема.

## Оценка защищенности системы

В соответствии с разработанной математической моделью (п. 2.2) и критериями классификации [25]: проведем идентификацию криптосистемы PRE и классической криптосистемой, использующей ассиметричное RSA шифрование:

* по доступности информации;
* по числу ключей;
* по стойкости криптоалгоритма;
* по используемым средствам шифрования;
* по наличию сертификата.

В таблице 2 приведена классификация сравниваемых криптосистем.

Таблица 2 – Классификация PRE и классической криптосистемы

|  |  |  |
| --- | --- | --- |
|  | PRE | RSA |
| C1 | ограниченного пользования | общего пользования |
| C2 | многоключевая | двухключевая |
| C3 | доказуемо стойкая | доказуемо стойкая |
| C4 | программная | программная |
| C5 | сертифицированная | сертифицированная |

Следующим этапом в процессе оценки эффективности (рисунок 2) является определение потенциальных взломщиков, на основании угроз из статьи [28] смоделированы два основных профиля атакующих по классификации [25]:

* по технической оснащенности;
* по конечной цели;
* по доступу к шифрующим средствам;
* по уровню подготовки;
* по первичной информации о средстве шифровании;
* по возможности кооперации.

В таблице 3 приведена классификация профилей взломщиков.

Таблица 3 – Классификация профилей взломщиков

|  |  |  |
| --- | --- | --- |
|  | Взломщик 1 | Взломщик 2 |
| B1 | сеть ЭВМ | персональный компьютер |
| B2 | полный взлом алгоритма | обнаружение слабости в алгоритме |
| B3 | “внешний” нарушитель | “внутренний” нарушитель |
| B4 | программирование | математический аппарат |
| B5 | криптограф | пользователь |

В соответствии со следующим этапом определение множества атак основывалось на статьях [13] и [28], наиболее подходящие:

Атаки с выбором шифротекста (Chosen Cyphertext Attack – CCA) – это вид атак, при котором у атакующего есть определенное количество криптограмм и возможность получить соответствующие им открытые текста [12].

Атаки с выбором открытого текста (Chosen Plaintext Attack – CPA) – это вид атак, при котором у атакующего есть определенное количество открытых текстов с соответствующими ими криптограммами [29].

Атаки на побочные каналы (Side channel attacks – SCA) — это вид криптографических атак, использующих информацию, полученную по этим каналам. Под таковой понимается информация, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни шифртекстом. Как правило, предполагается, что криптографические вычисления реализуются в виде идеальных «черных ящиков» в том смысле, что текущее состояние вычислительного процесса закрыто от враждебного наблюдателя. Единственной информацией, доступной криптоаналитику, является общая структура алгоритма шифрования, шифртекст и, зачастую, соответствующий ему открытый текст [30].

Классификация:

* по доступу к открытому и зашифрованному тексту;
* по контролю над процессом шифрования;
* по объему необходимых ресурсов;
* по степени применимости к различным шифрам;
* по используемым средствам;
* по последствиям атаки;
* по возможности распараллеливания.

В таблице 4 приведена классификация типичных атак на криптосистемы.

Таблица 4 – Классификация типичных атак на криптосистемы

|  |  |  |  |
| --- | --- | --- | --- |
|  | CCA - Chosen Cyphertext Attack | CPA - Chosen Plaintext Attack | SCA - Side channel attacks |
| A1 | только шифротекста | открытый текст | информация из побочных каналов |
| A2 | активная, с имитацией и подменой сообщения | пассивная | активная |
| A3 | память и время | данные | память, время и данные |
| A4 | универсальная | универсальная | универсальная |
| A5 | математические методы | математические методы | перехватчики физических параметров процесса шифрования |
| A6 | нарушение  конфиденциальности | нарушение конфиденциальности | нарушение конфиденциальности |
| A7 | распределенная | не распределенная | распределенная |

Для подсчёта функции взаимного влияния, которая определена в формуле (10), были заданы конкретные значения влияния параметров атаки на параметры криптосистемы, и взаимное влияние параметров взломщика и параметров атаки.

Подбор коэффициентов осуществлялся экспертом по информационной безопасности из ФИЦ ИУ РАН и приведен в таблицах 5–8.

Таблица 5 – Экспертные оценки взаимного влияния параметров атак на параметры криптосистемы (CCA; CPA; SCA)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **RSA** | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| C1 | 0.5;0;1 | 0.25;0.5;1 | 1;0.5;0.5 | 0.5;0.5;0.5 | 0.7;0.7;0.5 | 0.5;0.5;0.5 | 0.7;0.3;0.7 |
| C2 | 0.25;0;1 | 0.12;0.25;1 | 1;0.25;0.4 | 0.4;0.4;0.4 | 0.35;0.35;0.4 | 0.25;0.25;0.4 | 0.5;0.25;0.5 |
| C3 | 0.125;0;1 | 0.01;0.1;0.05 | 1;0.01;0.01 | 0.1;0.1;0.1 | 0.1;0.1;0.05 | 0.01;0.01;0.01 | 0.1;0.01;0.1 |
| C4 | 0.3;0;1 | 0.25;0.5;1 | 1;0.5;0.5 | 0.25;0.25;0.25 | 0.7;0.7;1 | 0.5;0.5;0.5 | 0.3;0.1;0.3 |
| C5 | 0.15;0;1 | 0.1;0.1;1 | 1;0.1;0.1 | 0.15;0.15;0.15 | 0.3;0.3;0.1 | 0.2;0.2;0.1 | 0.1;0.01;0.1 |
| **PRE** |  | | | | | | | |
| C1 | 0.25;0;1 | 0.01;0.25;1 | 1;0.25;0.1 | 0.5;0.5;0.5 | 0.35;0.35;0.3 | 0.25;0.25;0.25 | 0.3;0.1;0.3 |
| C2 | 0.125;0;1 | 0.05;0.125;1 | 1;0.125;0.125 | 0.2;0.2;0.2 | 0.35;0.35;0.15 | 0.25;0.25;0.25 | 0.2;0.125;0.2 |
| C3 | 0.125;0;1 | 0.01;0.1;0.05 | 1;0.01;0.01 | 0.1;0.1;0.1 | 0.1;0.1;0.05 | 0.01;0.01;0.01 | 0.1;0.01;0.1 |
| C4 | 0.3;0;1 | 0.25;0.5;1 | 1;0.5;0.5 | 0.25;0.25;0.25 | 0.7;0.7;1 | 0.5;0.5;0.5 | 0.3;0.1;0.3 |
| C5 | 0.15;0;1 | 0.1;0.1;1 | 1;0.1;0.1 | 0.15;0.15;0.15 | 0.3;0.3;0.1 | 0.2;0.2;0.1 | 0.1;0.01;0.1 |

Таблица 6 – Экспертные оценки взаимного влияния параметров злоумышленника и параметров атаки CCA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CCA** | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| B1 | 1;1.2 | 1;1 | 1;0.5 | 1;1 | 1.2;1 | 1;0.5 | 1;1 |
| B2 | 0.1;0.5 | 1.5;1 | 0.1;0.5 | 1;1 | 1;0.3 | 1;0.4 | 0.1;0.3 |
| B3 | 0;1.5 | 0;1.5 | 0;1.5 | 0;1.5 | 0;1.5 | 0;1.5 | 0;1.5 |
| B4 | 1;0.8 | 1.5;1 | 0.8;1 | 1;1 | 1;1.5 | 1;0.8 | 1.5;0.7 |
| B5 | 0;1 | 0;1 | 0;1 | 0;1 | 0;1 | 0;1 | 0;1 |
| B6 | 1.2;1 | 1.5;0.5 | 1;1 | 1;1 | 1.2;1 | 1.2;1 | 1.2;1 |

Таблица 7 – Экспертные оценки взаимного влияния параметров злоумышленника и параметров атаки CPA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CPA** | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| B1 | 0.5;1 | 1.2;1 | 1.2;1 | 1;1 | 1;1 | 0.7;0.2 | 0.9;1 |
| B2 | 0.2;0.7 | 0;1 | 0.2;0.5 | 1;1 | 1.2;0.9 | 1;0.8 | 0.1;0.3 |
| B3 | 1.5;0 | 1.5;0 | 1.5;0 | 1;1 | 1.5;0 | 1.5;0 | 1;1 |
| B4 | 1;0.8 | 1.5;1 | 0.8;1 | 1;1 | 1;1.5 | 1;0.8 | 1.5;0.7 |
| B5 | 1.2;0 | 1.5;0 | 1;1 | 1;1 | 1.5;0 | 1.5;0 | 1;0 |
| B6 | 1.2;1 | 1.5;0.8 | 1;1 | 1;1 | 1.5;1 | 1.5;0 | 1.5;1 |

Таблица 8 – Экспертные оценки взаимного влияния параметров злоумышленника и параметров атаки SCA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SCA** | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| B1 | 1.2;1 | 1.5;1 | 1.5;1 | 1;1 | 1;1.5 | 1;1 | 1.5;1 |
| B2 | 0.4;0.9 | 1;1 | 0.2;0.5 | 1;1 | 1.2;0.9 | 1;0.8 | 0.1;0.3 |
| B3 | 1.5;0 | 1.5;0 | 1.5;0 | 1;1 | 1.5;0 | 1.5;0 | 1;1 |
| B4 | 1;0.8 | 1.5;1 | 0.8;1 | 1;1 | 1;1.5 | 1;0.8 | 1.5;0.7 |
| B5 | 1.1;0 | 1.2;0 | 1;1 | 1;1 | 1.2;0 | 1.2;0 | 1;0 |
| B6 | 1.1;1 | 1.2;0.8 | 1;1 | 1;1 | 1.2;1 | 1.1;0 | 1.2;1 |

На основе экспертных оценок с помощью программы (см. приложение Б) был посчитан уровень ущерба от применения атаки к криптосистеме

Для PRE. ,,.

Для RSA. ,,.

Значения округлены до 9 знаков после запятой.

Также была посчитана с помощью программы (см. приложение В) вероятность того, что злоумышленник предпримет атаку .

Для CCA. , .

Для CPA., .

Для SCA., .

Значения округлены до 9 знаков после запятой.

На основе посчитанных данных можно рассчитать функцию риска по формуле (13), в таблице 9 приведены получившиеся значения:

Таблица 9 – Функция риска для PRE и классической криптосистемы

|  |  |  |  |
| --- | --- | --- | --- |
| PRE | CCA | CPA | SCA |
| Злоумышленник 1 | 0 | 0 |  |
| Злоумышленник 2 |  | 0 | 0 |
| RSA |  |  |  |
| Злоумышленник 1 | 0 | 0 |  |
| Злоумышленник 2 |  | 0 | 0 |

## Анализ полученных результатов

На основании полученных значений для функции риска можно заметить, что классическая криптосистема ассиметричного шифрования, как и криптосистема прокси решифрования полностью устойчивы с CPA атаке.

В случае CCA атаки обе криптосистемы достаточно устойчивы, однако PRE приблизительно в 5 раз более защищена от атаки на основе подобранного шифротекста.

Что касается SCA PRE менее уязвима в приблизительно 1.13 раза чем классическая криптосистема.

Таким образом, криптосистема прокси решифрования демонстрирует повышенную криптостойкость по сравнению с классической криптосистемой, особенно в случае CCA атаки. Тем не менее, обе криптосистемы показывают хорошую устойчивость к CPA атаке и достаточную защиту от SCA атаки, с небольшим преимуществом PRE.

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была рассмотрена тема «Криптосистема proxy re-encryption на базе блокчейн для безопасной передачи конфиденциальных данных в финансовом секторе». В процессе исследования были осуществлены следующие шаги:

* проведен обзор существующих решений для безопасной передачи конфиденциальных данных. Были изучены различные методы и протоколы, используемые для защиты данных в процессе их передачи, включая схему proxy re-encryption;
* описаны основные принципы криптографии, необходимые для понимания работы схемы proxy re-encryption. Рассмотрены алгоритмы шифрования, электронной подписи и асимметричного шифрования на основе эллиптических кривых;
* проведен анализ преимуществ и недостатков существующих методов защиты данных. Выявлены их особенности, сильные и слабые стороны, а также области применения;
* описана схема proxy re-encryption и принцип ее работы. Рассмотрены ключевые компоненты схемы, включая генерацию ключей, операции шифрования и расшифровки, а также алгоритмы решифрования;
* разработан подход к разработке алгоритмического и программного обеспечения для схемы прокси решифрования. Выбраны соответствующие инструменты, такие как онлайн-среда Remix Ethereum и локальная сеть блокчейна Ethereum Ganache, для разработки и тестирования смарт контрактов;
* описана методология тестирования, включающая юнит-тестирование, интеграционное тестирование, функциональное тестирование и нагрузочное тестирование. Обозначены цели каждого вида тестирования и их вклад в обеспечение качества и безопасности схемы proxy re-encryption.

Необходимо отметить, что практическая часть работы, включающая разработку алгоритмического и программного обеспечения, а также проведение тестирования, осталась за пределами данной работы.

В дальнейшем планируется реализация и тестирование схемы proxy re-encryption в реальной среде, чтобы подтвердить ее эффективность и безопасность. Это позволит дополнительно проверить и оценить применимость схемы в различных сценариях использования и ее соответствие требованиям безопасности.

Как показал подсчет защищенности PRE и сравнение с классической криптосистемой, система, использующая прокси решифрование, более устойчива к различным типам атак. Производительность при этом только на 11.5% хуже, чем у классической криптосистемы, что не сильно увеличивает общее время выполнения. Таким образом, в системах, где критически важен вопрос безопасной передачи конфиденциальных данных, например в финансовом секторе, стоит использовать PRE.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гатченко Н.А., Исаев А.С., Яковлев А.Д. «Криптографическая защита информации» – СПб: НИУ ИТМО, 2012. – 142 с.
2. Бузов Г.A. Защита от утечки информации по техническим каналам: Учебн. пособие / Бузов Г.A., Калинин C.B., Кондратьев A.B.- M.: Горячая линия - Телеком, 2005. – 416 c.
3. Темирбулатова Х. А., Шогенова Е. З., Хаджиев А. А. Современные программные средства защиты информации от утечек // Аллея науки. – 2021. – Т. 1. – №. 6. – С. 1166–1171.
4. Акбаров Д. Е. и др. Исследование особенностей критерия стойкости алгоритма хеш-функции // Central Asian journal of mathematical theory and computer sciences. – 2021. – Т. 2. – №. 11. – С. 60–64.
5. Шайтура, С. В., Питкевич, П. Н. Методы резервирования данных для критически важных информационных систем предприятия. – Russian Technological Journal. – 2022. – T. 10. – №1. – C. 28–34.
6. Базеева Н. А., Ушакова Е. Е. Криптография. Ее основополагающие методы, развитие и перспективы // E-Scio. – 2021. – №. 5 (56). – С. 572–578.
7. Ахметова А. Д., Мечкаев А. А., Галиахметова Д. Д. Симметричное шифрование // Инструменты, механизмы и технологии современного инновационного развития. – 2023. – С. 75–79.
8. Марков А. С., Цирлов В. Л. Основы криптографии: подготовка к CISSP // Вопросы кибербезопасности. – 2015. – №. 1 (9). – С. 65–73.
9. Иванов В. В., Лубова Е. С., Черкасов Д. Ю. Аутентификация и авторизация // Проблемы современной науки и образования. – 2017. – №. 2 (84). – С. 31–33.
10. Д. Г. Сущевский, О. В. Панченко, В. Н. Кугураков. Современные криптосистемы и их особенности // Вестник технологического университета. – 2015. – T.18. – С. 194–197.
11. Лапицкий Е. В. Алгоритмы шифрования данных // Развитие современной науки и образования. – 2023. – С. 42.
12. Manzoor, A., Liyanage, M., Braeken, A., Kanhere, S. S., Ylianttila, M. Blockchain based Proxy Re-Encryption Scheme for Secure IoT Data Sharing. – 2019.
13. Nunez D., Agudo I., Lopez J. Proxy Re-Encryption: Analysis of Constructions and its Application to Secure Access Delegation, – 2017.
14. Kirshanova E. Proxy re-encryption from lattices // Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17. – Springer Berlin Heidelberg, 2014. – P. 77–94.
15. Ge C., Xia J., Fang L. Key-private identity-based proxy re-encryption // Comput. Mater. Continua. – 2020. – Т. 63. – №. 2. – С. 633–647.
16. Yan L. Pairing-Free Certificate-Based Proxy Re-Encryption Plus Scheme for Secure Cloud Data Sharing // Electronics. – 2024. – Т. 13. – №. 3. – С. 534.
17. Кнэпп Э. Эллиптические кривые. – М.: Факториал, 2004. – 488 c.
18. Лось А. Б., Даниелян Ю. Ю. Сравнительный анализ систем обнаружения вторжений, представленных на отечественном рынке // Вестник Московского финансово-юридического университета. – 2014. – №. 3. – С. 181–187.
19. Savelieva A. Formal methods and tools for evaluating crypto- graphic systems security // St. Petersburg, ISP RAS, In Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering (SYRCoSE’2008), 2008. Часть 1. С. 33—36.
20. Кудряшов В. Е., Фионов А. Н. Проблема устойчивости современных криптосистем на фоне появления квантовых компьютеров // Интерэкспо Гео-Сибирь. – 2022. – Т. 6. – С. 109–115.
21. Bashir I. Mastering Blockchain: A Technical Reference Guide to What's Under the Hood of Blockchain, from Cryptography to DeFi and NFTs. – Packt Publishing, Limited, 2023.
22. Ahmad K.A., Ahmad K., Dulhare U. N. Functional Encryption. – EAISICC, 2021. – 309 с.
23. Башир И. Блокчейн: архитектура, криптовалюты, инструменты разработки, смарт-контракты. – М.: ДМК Пресс, 2019. – 538 c.
24. Прасти Н. Блокчейн. Разработка приложений. – СПб.: БХВ-Петербург, 2018. – 256 с.
25. Авдошин С. М., Савельева А. А. Проблемы оценки криптозащищенности информационных систем // “Новые информационные технологии”. Тез. докл. XVI Международной студенческой школы-семинара. М.: МИЭМ, 2008. – С. 15–29.
26. Мясников С. О., Намиот Д. Е. Инструменты нагрузочного тестирования // Прикладная информатика. – 2018. – Т. 13. – №. 1. – С. 92–102.
27. Гадельзянов Р. Ш., Хафизов И. Ф. Различные типы тестирования программного обеспечения // Современное состояние и перспективы развития науки и образования: проблемы и решения. – 2023. – С. 11–17.
28. Carter P. 6 of the biggest threats banks faced in 2023 [Электронный ресурс] // American Banker. – Режим доступа: <https://www.americanbanker.com/list/6-of-the-biggest-threats-banks-faced-in-2023>
29. Петров А. А. Компьютерная безопасность. Криптографические методы защиты. – М.: ДМК Пресс, 2022. – 448 с.
30. Жуков А.Е. Криптоанализ по побочным каналам (side channel attacks) // Защита информации. Инсайд. – 2010 № 5 (35). – С. 28–33.

# ПРИЛОЖЕНИЕ A

ca.py

**import** http.server  
**import** ssl  
**import** json  
**import** logging  
  
**from** util **import** convert\_object\_to\_hex\_str, setup\_pre  
  
  
logging.basicConfig(level=logging.INFO)  
  
pre, group = setup\_pre()  
(master\_secret\_key, params) = pre.setup()  
  
  
**class** Handler(http.server.SimpleHTTPRequestHandler):  
 **def** do\_POST(self):  
 **if** self.path == **'/key'**:  
 content\_length = int(self.headers[**'Content-Length'**])  
 post\_data = self.rfile.read(content\_length)  
  
 **try**:  
 data = json.loads(post\_data.decode(**'utf-8'**))  
  
 sender\_id = data.get(**'sender\_id'**)  
 logging.info(**"Got key request from: %s"**, sender\_id)  
  
 id\_secret\_key = pre.keyGen(master\_secret\_key, sender\_id)  
  
 \_, \_, params\_hex\_str, id\_secret\_key\_hex\_str = convert\_object\_to\_hex\_str(  
 group,  
 id\_secret\_key=id\_secret\_key,  
 params=params  
 )  
  
 response\_message = json.dumps({  
 **'id\_secret\_key\_hex\_str'**: id\_secret\_key\_hex\_str,  
 **'params\_hex\_str'**: params\_hex\_str,  
 })  
  
 self.send\_response(200)  
 self.send\_header(**'Content-type'**, **'text/html'**)  
 self.end\_headers()  
 self.wfile.write(response\_message.encode(**'utf-8'**))  
  
 **except** json.JSONDecodeError:  
 self.send\_response(400)  
 self.send\_header(**'Content-type'**, **'text/html'**)  
 self.end\_headers()  
 self.wfile.write(b'Invalid JSON received')  
  
 **return  
  
 return** super().do\_GET()  
  
  
**def** main():  
 server\_address = (**'0.0.0.0'**, 443)  
 httpd = http.server.HTTPServer(server\_address, Handler)  
 httpd.socket = ssl.wrap\_socket(  
 httpd.socket,  
 server\_side=**True**,  
 certfile=**'/etc/nginx/ssl/cert.pem'**,  
 keyfile=**'/etc/nginx/ssl/key.pem'**,  
 ssl\_version=ssl.PROTOCOL\_TLS  
 )  
 httpd.serve\_forever()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

sender.py:

**import** socket  
**import** argparse  
**import** json  
**import** logging  
  
**from** util **import** get\_key\_params, setup\_pre, convert\_object\_to\_hex\_str, send\_large\_message  
**from** constants **import** CA\_URL  
**from** smart\_contract **import** contract  
  
logging.basicConfig(level=logging.INFO)  
pre, group = setup\_pre()  
ID = **"client-a"**ID2 = **"client-b"  
  
  
def** main(host\_ip, host\_port):  
 id\_secret\_key, params = get\_key\_params(CA\_URL, ID, group)  
  
 client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 client\_socket.connect((host\_ip, host\_port))  
  
 message = **"<confidential message>"** encrypted\_message = pre.encrypt(params, ID, message)  
 rekey = pre.rkGen(params, id\_secret\_key, ID, ID2)  
 logging.info(**f"PARAMS: {**params**}"**)  
 logging.info(**f"MESSAGE: {**encrypted\_message**}"**)  
 logging.info(**f"REKEY: {**rekey**}"**)  
 encrypted\_message\_hex\_str, rekey\_hex\_str, \_, \_ = convert\_object\_to\_hex\_str(  
 group,  
 message=encrypted\_message,  
 rekey=rekey,  
 )  
  
 rekey = json.dumps(rekey\_hex\_str)  
 contract.addReKey(ID, rekey)  
  
 logging.info(encrypted\_message\_hex\_str)  
  
 message\_data = json.dumps({  
 **'sender\_id'**: ID,  
 **'encrypted\_message\_hex\_str'**: encrypted\_message\_hex\_str,  
 **'rekey\_hex\_str'**: rekey\_hex\_str,  
 })  
  
 print(type(message\_data))  
  
 send\_large\_message(client\_socket, message\_data)  
  
 response = client\_socket.recv(1024)  
 logging.info(**"Get response from client: %s"**, response.decode(**'utf-8'**))  
  
 client\_socket.close()  
  
  
**def** parse\_args():  
 parser = argparse.ArgumentParser()  
 parser.add\_argument(**"--host-ip"**, type=str, default=**"0.0.0.0"**)  
 parser.add\_argument(**"--host-port"**, type=int, default=1024)  
 parser.add\_argument(**"--client-name"**, type=str, default=**"client-a"**)  
  
 **return** parser.parse\_args()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 args = parse\_args()  
 main(args.host\_ip, args.host\_port)

proxy.py

**import** requests  
**import** socket  
**import** argparse  
**import** logging  
**import** json  
  
**from** constants **import** CA\_URL  
**from** util **import** receive\_large\_message, convert\_hex\_str\_to\_object, setup\_pre, get\_key\_params, convert\_object\_to\_hex\_str  
**from** smart\_contract **import** contract  
  
logging.basicConfig(level=logging.INFO)  
pre, group = setup\_pre()  
  
  
**def** main(args):  
 server\_host = **'0.0.0.0'** server\_port = 1024  
  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 server\_socket.bind((server\_host, server\_port))  
 server\_socket.listen(5)  
 \_, params = get\_key\_params(CA\_URL, **""**, group)  
  
 logging.info(**f"Server listen {**server\_host**}:{**server\_port**}"**)  
  
 **while True**:  
 client\_socket, client\_address = server\_socket.accept()  
  
 message\_data = receive\_large\_message(client\_socket)logging.info(**f"Got encrypted message from client: {**message\_data**}"**)  
  
 data = json.loads(message\_data)  
 sender\_id = data.get(**'sender\_id'**)  
 encrypted\_message\_hex\_str = data.get(**'encrypted\_message\_hex\_str'**)  
  
 rekey\_hex\_str = json.loads(contract.getReKey(sender\_id))  
 logging.info(**f"{**type(rekey\_hex\_str)**} REKEY\_HEX\_STR\_FROM\_CONTRACT: {**rekey\_hex\_str**}"**)  
  
 logging.info(**f"REKEY\_ENCODED: {**rekey\_hex\_str**}"**)  
  
 encrypted\_message, rekey, \_, \_ = convert\_hex\_str\_to\_object(  
 group,  
 message\_hex\_str=encrypted\_message\_hex\_str,  
 rekey\_hex\_str=rekey\_hex\_str  
 )  
  
 logging.info(**f"REKEY: {**rekey**}"**)  
  
 encrypted\_message = pre.reEncrypt(params, sender\_id, rekey, encrypted\_message)  
 encrypted\_message\_hex\_str, \_, \_, \_ = convert\_object\_to\_hex\_str(  
 group,  
 message=encrypted\_message  
 )  
  
 logging.info(**f"REENCRYPTED\_MESSAGE {**encrypted\_message\_hex\_str**}"**)  
  
 data[**'encrypted\_message\_hex\_str'**] = encrypted\_message\_hex\_str  
 data = json.dumps(data)  
  
 url = args.client\_url  
 response = requests.post(url, data=data.encode(**'utf-8'**), verify=args.client\_cert)  
  
 client\_socket.send(response.text.encode(**'utf-8'**))  
 client\_socket.close()  
  
  
**def** parse\_args():  
 parser = argparse.ArgumentParser()  
 parser.add\_argument(**"--client-url"**, type=str, default=**"http://client-b.test-zone.ru/secure"**)  
 parser.add\_argument(**"--client-cert"**, type=str, metavar=**"PATH"**,  
 default=**"/usr/local/share/ca-certificates/client-b.crt"**)  
  
 **return** parser.parse\_args()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 args = parse\_args()  
 main(args)

receiver.py

**import** http.server  
**import** json  
**import** ssl  
**import** logging  
  
**from** util **import** get\_key\_params, setup\_pre, convert\_hex\_str\_to\_object  
**from** constants **import** CA\_URL  
  
  
logging.basicConfig(level=logging.INFO)  
pre, group = setup\_pre()  
ID = **"client-b"***# REKEY: {'N': "<class 'integer.Element'>", 'R': "<class 'pairing.Element'>"}  
# {  
# MESSAGE 'S': "<class 'pairing.Element'>", 'C': {  
# 'A': "<class 'pairing.Element'>",  
# 'B': "<class 'pairing.Element'>",  
# 'C': "<class 'integer.Element'>"  
# }  
# }***class** Handler(http.server.SimpleHTTPRequestHandler):  
 **def** do\_POST(self):  
 **if** self.path == **'/secure'**:  
 content\_length = int(self.headers[**'Content-Length'**]) *# Получаем размер данных* post\_data = self.rfile.read(content\_length) *# Читаем данные* **try**:  
 id\_secret\_key, params = get\_key\_params(CA\_URL, ID, group)  
  
 logging.info(**"Got data: %r"**, post\_data)  
 data = json.loads(post\_data.decode(**'utf-8'**))  
  
 sender\_id = data.get(**'sender\_id'**)  
 encrypted\_message\_hex\_str = data.get(**'encrypted\_message\_hex\_str'**)  
 encrypted\_message, \_, \_, \_ = convert\_hex\_str\_to\_object(  
 group,  
 message\_hex\_str=encrypted\_message\_hex\_str,  
 )  
  
 logging.info(**f"PARAMS: {**params**}"**)  
 logging.info(**f"MESSAGE: {**encrypted\_message**}"**)  
  
 decrypted\_message = pre.decryptSecondLevel(params, id\_secret\_key, **"client-a"**, ID, encrypted\_message)  
 logging.info(**"Decrypted message: %s, from client: %s"**, decrypted\_message, sender\_id)  
  
 self.send\_response(200)  
 self.send\_header(**'Content-type'**, **'text/html'**)  
 self.end\_headers()  
 response\_message = **"Data received"**.encode()  
 self.wfile.write(response\_message)  
  
 **except** json.JSONDecodeError **as** e:  
 logging.info(**"Got error %s"**, e)  
 self.send\_response(400)  
 self.send\_header(**'Content-type'**, **'text/html'**)  
 self.end\_headers()  
 self.wfile.write(b'Invalid JSON received')  
  
 **return  
  
 return** super().do\_GET()  
  
  
**def** main():  
 server\_address = (**'0.0.0.0'**, 443)  
 httpd = http.server.HTTPServer(server\_address, Handler)  
 httpd.socket = ssl.wrap\_socket(  
 httpd.socket,  
 server\_side=**True**,  
 certfile=**'/etc/nginx/ssl/cert.pem'**,  
 keyfile=**'/etc/nginx/ssl/key.pem'**,  
 ssl\_version=ssl.PROTOCOL\_TLS  
 )  
 httpd.serve\_forever()  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

smart\_contract.py

**from** web3 **import** Web3  
**import** logging  
  
**from** constants **import** CONTRACT\_ADDRESS, SENDER\_ADDRESS  
  
  
**class** SmartContract:  
 abi = [  
 {**'inputs'**: [],  
 **'stateMutability'**: **'nonpayable'**,  
 **'type'**: **'constructor'** },  
 {  
 **'inputs'**: [  
 {**'internalType'**: **'string'**, **'name'**: **'clientName'**, **'type'**: **'string'**},  
 {**'internalType'**: **'string'**, **'name'**: **'reKey'**, **'type'**: **'string'**}  
 ],  
 **'name'**: **'addReKey'**,  
 **'outputs'**: [],  
 **'stateMutability'**: **'nonpayable'**,  
 **'type'**: **'function'** },  
 {  
 **'inputs'**: [  
 {**'internalType'**: **'string'**, **'name'**: **'clientName'**, **'type'**: **'string'**}  
 ],  
 **'name'**: **'getReKey'**,  
 **'outputs'**: [{**'internalType'**: **'string'**, **'name'**: **''**, **'type'**: **'string'**}],  
 **'stateMutability'**: **'view'**,  
 **'type'**: **'function'** },  
 {  
 **'inputs'**: [],  
 **'name'**: **'owner'**,  
 **'outputs'**: [  
 {**'internalType'**: **'address'**, **'name'**: **''**, **'type'**: **'address'**}  
 ],  
 **'stateMutability'**: **'view'**, **'type'**: **'function'** }  
 ]  
  
 **def** \_\_init\_\_(self, blockchain\_addr):  
 self.web3 = Web3(Web3.HTTPProvider(blockchain\_addr))  
  
 contract\_address = CONTRACT\_ADDRESS  
 self.contract = self.web3.eth.contract(address=contract\_address, abi=SmartContract.abi)  
  
 **def** addReKey(self, clientName, reKey):  
 logging.info(**"Add rekey for %s"**, clientName)  
 tx\_hash = self.contract.functions.addReKey(clientName, reKey).transact({  
 **'from'**: SENDER\_ADDRESS,  
 **'gas'**: 2000000,  
 })  
  
 self.web3.eth.wait\_for\_transaction\_receipt(tx\_hash)  
  
 **def** getReKey(self, clientName):  
 logging.info(**"Get rekey for client: %s"**, clientName)  
 rekey\_str = self.contract.functions.getReKey(clientName).call()  
  
 **return** rekey\_str  
  
  
contract = SmartContract(**"http://127.0.0.1:8545"**, **""**)

util.py

**import** os  
**import** json  
**import** zlib  
  
**import** requests  
  
**from** datetime **import** datetime, timedelta  
**from** charm.core.engine.util **import** objectToBytes, bytesToObject  
**from** charm.toolbox.pairinggroup **import** PairingGroup  
**from** charm.schemes.pre\_mg07 **import** PreGA  
**from** charm.toolbox.conversion **import** Conversion  
  
  
**def** serialize\_data(group, data):  
 **if** isinstance(data, dict):  
 serialized\_data = {}  
 **for** key, value **in** data.items():  
 serialized\_data[key] = serialize\_data(group, value)  
 **return** serialized\_data  
 **else**:  
 **try**:  
 **return** objectToBytes(data, group).hex()  
 **except** TypeError:  
 **return** Conversion.IP2OS(data).hex()  
  
  
**def** deserialize\_data(group, serialized\_data):  
 **if** isinstance(serialized\_data, dict):  
 data = {}  
 **for** key, value **in** serialized\_data.items():  
 data[key] = deserialize\_data(group, value) *# Рекурсивная обработка для словарей* **return** data  
 **else**:  
 **try**:  
 **return** bytesToObject(bytes.fromhex(serialized\_data), group)  
 **except** zlib.error:  
 **return** Conversion.OS2IP(bytes.fromhex(serialized\_data), element=**True**)  
  
  
**def** convert\_object\_to\_hex\_str(group, message=**None**, rekey=**None**, params=**None**, id\_secret\_key=**None**):  
 id\_secret\_key\_hex\_str = **None  
 if** id\_secret\_key:  
 id\_secret\_key\_bytes = objectToBytes(id\_secret\_key, group)  
 id\_secret\_key\_hex\_str = id\_secret\_key\_bytes.hex()  
  
 params\_hex\_str = **None  
 if** params:  
 params\_hex\_str = {k: objectToBytes(v, group).hex() **for** k, v **in** params.items()}  
  
 message\_hex\_str = **None  
 if** message:  
 message\_hex\_str = serialize\_data(group, message)  
  
 rekey\_hex\_str = **None  
 if** rekey:  
 rekey\_hex\_str = serialize\_data(group, rekey)  
  
 **return** message\_hex\_str, rekey\_hex\_str, params\_hex\_str, id\_secret\_key\_hex\_str  
  
  
**def** convert\_hex\_str\_to\_object(group, message\_hex\_str=**None**, rekey\_hex\_str=**None**,  
 params\_hex\_str=**None**, id\_secret\_key\_hex\_str=**None**):  
 id\_secret\_key = **None  
 if** id\_secret\_key\_hex\_str:  
 id\_secret\_key\_bytes = bytes.fromhex(id\_secret\_key\_hex\_str)  
 id\_secret\_key = bytesToObject(id\_secret\_key\_bytes, group)  
  
 params = **None  
 if** params\_hex\_str:  
 params = {k: bytesToObject(bytes.fromhex(v), group) **for** k, v **in** params\_hex\_str.items()}  
  
 message = **None  
 if** message\_hex\_str:  
 message = deserialize\_data(group, message\_hex\_str)  
  
 rekey = **None  
 if** rekey\_hex\_str:  
 rekey = deserialize\_data(group, rekey\_hex\_str)  
  
 **return** message, rekey, params, id\_secret\_key  
  
  
**def** get\_key\_params(url, id, group):  
 message\_data = json.dumps({  
 **'sender\_id'**: id,  
 })  
 response = requests.post(url, data=message\_data)  
 id\_secret\_key\_hex\_str = json.loads(response.text).get(**"id\_secret\_key\_hex\_str"**)  
 params\_hex\_str = json.loads(response.text).get(**"params\_hex\_str"**)  
  
 \_, \_, params, id\_secret\_key = convert\_hex\_str\_to\_object(group, id\_secret\_key\_hex\_str=id\_secret\_key\_hex\_str,  
 params\_hex\_str=params\_hex\_str)  
  
 **return** id\_secret\_key, params  
  
  
**def** setup\_pre():  
 group = PairingGroup(**'SS512'**, secparam=1024)  
 pre = PreGA(group)  
 **return** pre, group**def** send\_large\_message(socket, message\_data, chunk\_size=4096):  
 *# Конвертируем данные в байты* message\_bytes = message\_data.encode(**'utf-8'**)  
  
 *# Отправляем размер сообщения перед самим сообщением* total\_size = len(message\_bytes)  
 socket.sendall(total\_size.to\_bytes(4, **'big'**))  
  
 *# Разбиваем сообщение на части и отправляем каждую часть* **for** i **in** range(0, total\_size, chunk\_size):  
 chunk = message\_bytes[i:i + chunk\_size]  
 socket.sendall(chunk)  
  
  
**def** receive\_large\_message(socket, chunk\_size=4096):total\_size\_bytes = socket.recv(4)  
 total\_size = int.from\_bytes(total\_size\_bytes, **'big'**)  
chunks = []  
 bytes\_recd = 0  
 **while** bytes\_recd < total\_size:  
 chunk = socket.recv(min(total\_size - bytes\_recd, chunk\_size))  
 **if** chunk == b'':  
 **raise** RuntimeError(**"Socket connection broken"**)  
 chunks.append(chunk)  
 bytes\_recd += len(chunk)  
**return** b''.join(chunks).decode(**'utf-8'**)

constants.py

CA\_URL = **"http://127.0.0.1:1026/key"**CONTRACT\_ADDRESS = **"0x7Dc0fa574F7cb11FE3E86CcD1e39AC4b7e17d658"**

# ПРИЛОЖЕНИЕ Б

**import** csv  
**from** copy **import** deepcopy  
  
  
sumc\_a = {}  
**for** i **in** range(1,8):  
 sumc\_a[i] = {  
 **"CCA"**: 0,  
 **"CPA"**: 0,  
 **"SCA"**: 0  
 }  
  
c\_a = {  
 **"CCA"**: [[0.0] \* 7 **for** i **in** range (5)],  
 **"CPA"**: [[0.0] \* 7 **for** i **in** range (5)],  
 **"SCA"**: [[0.0] \* 7 **for** i **in** range (5)],  
}  
  
attacks = [**'CCA'**, **'CPA'**, **'SCA'**]  
**with** open(**'tables/c\_a\_rsa.csv'**, newline=**""**) **as** csvfile:  
 data = csv.reader(csvfile, delimiter=**";"**)  
 next(data)  
 **for** index, row **in** enumerate(data):  
 row = row[1:]  
 **for** row\_index, values **in** enumerate(row):  
 values = values.split(**';'**)  
 **for** attack\_index, attack **in** enumerate(attacks):  
 sumc\_a[row\_index + 1][attack] += float(values[attack\_index])  
 c\_a[attack][index][row\_index] = float(values[attack\_index])  
  
normalized\_func = deepcopy(c\_a)  
**for** key, items **in** c\_a.items():  
 **for** row\_index, row **in** enumerate(items):  
 **for** index, value **in** enumerate(row):  
 normalized\_func[key][row\_index][index] = value / sumc\_a[index + 1][key] **if** sumc\_a[index + 1][key] != 0.0 **else** 0.0  
  
min\_normalized\_func = []  
**for** index, row **in** enumerate(normalized\_func[**'CCA'**]):  
 min\_cca = 2.0  
 min\_cpa = 2.0  
 min\_sca = 2.0  
 **for** value\_index, value **in** enumerate(row):  
 print(**f"{**value**};{**normalized\_func[**'CPA'**][index][value\_index]**};{**normalized\_func[**'SCA'**][index][value\_index]**}"**, end=**' '**)  
 min\_cca = min(min\_cca, value)  
 min\_cpa = min(min\_cpa, normalized\_func[**'CPA'**][index][value\_index])  
 min\_sca = min(min\_sca, normalized\_func[**'SCA'**][index][value\_index])  
 print(**f"MIN: {**min\_cca**};{**min\_cpa**};{**min\_sca**}"**)  
 min\_normalized\_func.append((min\_cca, min\_cpa, min\_sca))  
 print()  
  
i\_c\_a\_all = [1, 1, 1]  
**for** i **in** range(5):  
 **for** index, min\_value **in** enumerate(min\_normalized\_func[i]):  
 i\_c\_a\_all[index] \*= min\_valueprint(\*i\_c\_a\_all)

# ПРИЛОЖЕНИЕ В

**import** csv  
**from** copy **import** deepcopy  
  
sumb\_a = {}  
**for** i **in** range(1,8):  
 sumb\_a[i] = {  
 **"first\_attacker"**: 0,  
 **"second\_attacker"**: 0,  
 }  
  
c\_a = {  
 **"first\_attacker"**: [[0.0] \* 7 **for** i **in** range (6)],  
 **"second\_attacker"**: [[0.0] \* 7 **for** i **in** range (6)],  
}  
  
attackers = [**'first\_attacker'**, **'second\_attacker'**, ]  
**with** open(**'tables/b\_a\_sca.csv'**, newline=**""**) **as** csvfile:  
 data = csv.reader(csvfile, delimiter=**";"**)  
 next(data)  
 **for** index, row **in** enumerate(data):  
 row = row[1:]  
 **for** row\_index, values **in** enumerate(row):  
 values = values.split(**';'**)  
 **for** attack\_index, attack **in** enumerate(attackers):  
 sumb\_a[row\_index + 1][attack] += float(values[attack\_index])  
 c\_a[attack][index][row\_index] = float(values[attack\_index])  
  
normalized\_func = deepcopy(c\_a)  
**for** key, items **in** c\_a.items():  
 **for** row\_index, row **in** enumerate(items):  
 **for** index, value **in** enumerate(row):  
 normalized\_func[key][row\_index][index] = value / sumb\_a[index + 1][key] **if** sumb\_a[index + 1][key] != 0.0 **else** 0.0  
  
min\_normalized\_func = []  
**for** index, row **in** enumerate(normalized\_func[**'first\_attacker'**]):  
 min\_first = 2.0  
 min\_second = 2.0  
 min\_sca = 2.0  
 **for** value\_index, value **in** enumerate(row):  
 print(**f"{**value**};{**normalized\_func[**'second\_attacker'**][index][value\_index]**}"**, end=**' '**)  
 min\_first = min(min\_first, value)  
 min\_second = min(min\_second, normalized\_func[**'second\_attacker'**][index][value\_index])  
 print(**f"MIN: {**min\_first**};{**min\_second**};"**)  
 min\_normalized\_func.append((min\_first, min\_second))  
 print()  
  
print(min\_normalized\_func)  
i\_c\_a\_all = [1, 1]  
**for** i **in** range(6):  
 **for** index, min\_value **in** enumerate(min\_normalized\_func[i]):  
 i\_c\_a\_all[index] \*= min\_valueprint(\*i\_c\_a\_all)