МОУ «Громовская СОШ»

XVI Межрегиональная научно-практическая конференция учащихся «Ломанская линия», посвященная Году единства народов России

НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ПРОЕКТ

D-MASH (Decentralized Messenger with Asymmetric-Key Encryption, Anti-Forensic Hardening and DSP Overlay)

Разработка прототипа децентрализованного мессенджера с антикриминалистической защитой метаданных и резервными каналами связи на основе DSP

Ученик: _____ С.С. Генералов
(МОУ «Громовская СОШ», 11 класс)

Руководитель: _____ Е.О. Генералова
(Преподаватель, МОУ «Громовская СОШ»)

г. Приозерск
2026

РЕФЕРАТ

Проект содержит 56 страниц, 8 рисунков, 1 таблицу, 10 источников, 6 приложений.

Ключевые слова: информационная безопасность, децентрализация, mesh-сети, криптография ECC, обфускация трафика, антикриминалистика (anti-forensics), Proof-of-Work, DSP, MFSK, скремблирование, PWA, WASM, PQC, ML-KEM, WebAuthn

Объект исследования: Методы обеспечения конфиденциальности в децентрализованных системах обмена сообщениями.

Предмет исследования: Архитектура и программная реализация P2P-мессенджера D-MASH, устойчивого к анализу трафика, блокировкам, квантовым компьютерам и физической компрометации.

Цель работы: Спроектировать и реализовать программный прототип P2P-мессенджера D-MASH, обеспечивающего многоуровневую защиту от современных и будущих угроз за счет комбинации постквантовой криптографии, протоколов обфускации, аппаратных методов защиты и резервных каналов связи

Методы исследования: Системное проектирование, моделирование угроз, разработка криптографических и сетевых протоколов, программная инженерия, цифровая обработка сигналов.

Результаты и новизна: Реализована уникальная архитектура на базе двойного WASM-ядра. Система «Blind Storage» исключает криминалистическое восстановление данных: индексы шифруются эфемерным ключом Argon2id в RAM. Внедрен протокол T-Ratchet с квантовым хендшейком (Kyber-768) и атомарной генерацией ключа с миллисекундной точностью, исключающей компрометацию сессии. Протокол «Tact» и концепция «PWA Decoy» обеспечивают сокрытие трафика и маскировку ПО. Внедрены аппаратные триггеры экстренного стирания ключей (G-Force Wipe) и DSP-протоколы для передачи данных через аналоговые каналы.

Область применения: Защищенные коммуникации для пользователей с максимальными требованиями к приватности в условиях цензуры или угрозы физической компрометации устройств.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API — Application Programming Interface (Программный интерфейс приложения)

BLAKE3 — Высокопроизводительная криптографическая хеш-функция

Curve25519 — Эллиптическая кривая, используемая для асимметричного шифрования и реализации протокола обмена ключами Диффи-Хеллмана (ECDH)

D-MASH — Система, состоящая из P2P-сети маршрутизаторов и клиента на базе PWA

Docker — Программное обеспечение для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации.

DSP — Digital Signal Processing (Цифровая обработка сигналов)

E2EE — End-to-End Encryption (Сквозное шифрование)

ECC — Elliptic Curve Cryptography (Криптография на эллиптических кривых)

Ed25519 — Алгоритм цифровой подписи на основе эллиптических кривых

FFT — Fast Fourier Transform (Быстрое преобразование Фурье)

GVP — Ghost Voice Protocol (Протокол "Призрачный голос")

KDF (Key Derivation Function) — функция формирования ключей на базе пароля и временных меток.

Knox PRF — расширение WebAuthn, используемое на устройствах Samsung, позволяющее извлекать аппаратный секрет из защищенного анклава (TEE) для шифрования ключей.

MFSK — Multi-frequency Shift Keying (Многочастотная манипуляция)

ML-KEM (Kyber-768) — криптографический стандарт постквантовой инкапсуляции.

P2P — Peer-to-Peer (Одноранговая, децентрализованная сеть)

PCP — Phantom Call Protocol (Протокол "Фантомный вызов")

PoW — Proof-of-Work (Доказательство работы)

PQC (Post-Quantum Cryptography) — криптография, устойчивая к атакам с использованием квантовых компьютеров.

PWA (Progressive Web App) — технология, позволяющая установить веб-приложение на устройство, обеспечивая работу в офлайн-режиме и мимикрию под системное ПО.

UI — User Interface (Пользовательский интерфейс)

WASM (WebAssembly) — бинарный формат инструкций, позволяющий исполнять код, написанный на C/C++, в браузере с околонативной производительностью; используется для криптографических вычислений (Argon2id, Kyber).

WebAuthn — стандарт веб-аутентификации, позволяющий использовать биометрию и аппаратные ключи для входа без пароля

ВВЕДЕНИЕ

В современную цифровую эпоху конфиденциальность личных и профессиональных коммуникаций находится под постоянной угрозой. Глобальная централизация интернет-сервисов, находящихся под контролем технологических гигантов, усиление государственного надзора и рост киберпреступности создают среду, в которой популярные мессенджеры перестают быть надежным инструментом для защищенного общения. Существующие решения, несмотря на маркетинговые заявления о безопасности, обладают фундаментальными архитектурными недостатками, которые могут быть и уже используются для компрометации данных пользователей.

Актуальность данного проекта обусловлена наличием доказанных уязвимостей в доминирующих на рынке мессенджерах и отсутствием массового коммуникационного инструмента, который бы комплексно защищал пользователя от всех векторов атак: от перехвата трафика и анализа метаданных до физического принуждения. Ни одно из рассмотренных ниже решений не предлагает действенных механизмов защиты от криминалистического анализа устройства после его изъятия, оставляя пользователя беззащитным в наиболее критических сценариях.

Цель работы: спроектировать и программно реализовать прототип защищенного P2P-мессенджера D-MASH, обеспечивающего качественно новый уровень приватности и устойчивости к блокировкам за счет комбинации децентрализованной архитектуры, протоколов обфускации трафика, антикриминалистической защиты метаданных и наличия резервных каналов связи.

Для достижения поставленной цели необходимо решить следующие задачи:

— Провести анализ уязвимостей существующих централизованных мессенджеров.

— Разработать полностью децентрализованную сетевую архитектуру на основе P2P.

— Спроектировать и реализовать криптографическое ядро, обеспечивающее надежное E2EE на основе гибридной криптосистемы, сочетающей классические ECC-алгоритмы и постквантовый стандарт ML-KEM (Kyber-768).

— Разработать и программно реализовать систему антикриминалистической защиты «Blind Storage» для противодействия физической компрометации устройств, включая автоматические триггеры экстренного стирания ключей на основе данных акселерометра (G-Force Wipe) и гироскопа (Flip-Lock).

— Разработать и программно реализовать протокол обфускации трафика «Tact Protocol» для защиты от анализа сетевого трафика.

— Разработать и программно реализовать комплекс протоколов экстренной связи через голосовые каналы (PCP и GVP) на основе технологий цифровой обработки сигналов.

— Провести интеграционное тестирование реализованного прототипа для валидации ключевых архитектурных решений.

— Разработать стелс-клиент на базе технологии PWA, обеспечивающий мимикрию под системное приложение («PWA Decoy») и установку в обход централизованных магазинов приложений.

— Интегрировать механизм аппаратной привязки ключей с использованием стандарта WebAuthn и расширения PRF для защиты от клонирования данных приложения и реализации беспарольного входа.

— Реализовать гибридную сетевую модель с поддержкой «бесшумных» ретрансляторов (Silent Relays) для обеспечения связи PWA-клиентов в условиях ограничений браузерной среды.

Пояснительная записка имеет следующую структуру: в первой главе приводится анализ предметной области и существующих решений. Во второй главе детально описывается проектируемая архитектура и ключевые протоколы системы D-MASH. В третьей главе рассматривается программная реализация прототипа. В четвертой главе представлены методология и результаты тестирования системы.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ

1.1 Обзор современных угроз конфиденциальности коммуникаций

Современная модель угроз для пользователей систем обмена мгновенными сообщениями является многоуровневой и затрагивает как сам канал передачи данных, так и конечные устройства. Для разработки комплексной системы защиты необходимо учитывать весь спектр потенциальных векторов атак, которые выходят далеко за рамки простого перехвата сообщений.

Ключевые угрозы включают:

— пассивное прослушивание (Eavesdropping): Перехват и анализ сетевого трафика на различных участках сети — от локальной Wi-Fi сети до магистральных каналов провайдеров. Без надежного сквозного шифрования это позволяет злоумышленнику прочитать содержимое сообщений;

— атаки «человек посередине» (Man-in-the-Middle, MITM): Активная атака, при которой злоумышленник встраивается в канал связи между двумя сторонами, имея возможность не только читать, но и модифицировать передаваемые данные. Для противодействия требуются надежные механизмы аутентификации ключей [3];

— анализ метаданных: Даже при использовании сквозного шифрования, централизованные серверы собирают и хранят огромные объемы метаданных: идентификаторы отправителя и получателя, время отправки, частоту и продолжительность сеансов связи, IP-адреса. Анализ этих данных позволяет составить подробный социальный граф пользователя и сделать выводы о его деятельности, что является основной целью систем массового наблюдения;

— сетевые блокировки и цензура: Государственные регуляторы или интернет-провайдеры могут блокировать доступ к центральным серверам мессенджера по IP-адресам или доменным именам (технология DPI), делая сервис полностью недоступным в регионе [2];

— компрометация централизованной инфраструктуры: Серверы мессенджера являются единой точкой отказа. Их взлом или юридическое принуждение владельцев к сотрудничеству может привести к массовой утечке пользовательских данных или ключей шифрования;

— физическая компрометация устройства: Наиболее прямая и критическая угроза, при которой злоумышленник получает физический доступ к устройству пользователя (например, в результате изъятия) и принуждает его предоставить пароль. Стандартные мессенджеры не предоставляют механизмов защиты от криминалистического анализа носителей информации в таком сценарии.

Комплексная система защиты должна обеспечивать противодействие угрозам на каждом из этих уровней, что не достигается в полной мере ни одним из существующих популярных решений.

1.2 Критический анализ аналогов: Telegram, WhatsApp, Signal

Для оценки текущего состояния рынка был проведен сравнительный анализ трех ключевых игроков по критериям, вытекающим из описанной модели угроз.

Telegram. Позиционируемый как безопасный мессенджер, Telegram по умолчанию использует архитектуру облачных чатов, где сквозное шифрование (E2EE) отсутствует. Сообщения шифруются между клиентом и сервером, а затем между сервером и другим клиентом, однако на самих серверах Telegram они хранятся в расшифрованном или доступном для расшифровки виде. Это позволяет реализовать удобную синхронизацию

истории между устройствами, но полностью компрометирует конфиденциальность переписки перед владельцами инфраструктуры или любыми третьими лицами, получившими к ней доступ. Сквозное шифрование доступно лишь в опциональной функции «Secret Chats», которая неудобна в использовании и не поддерживает групповые чаты. Более того, даже в режиме «Secret Chats» протокол MTProto передает идентификаторы участников (user_id) в метаданных сообщения в виде, доступном серверу. Это позволяет серверам Telegram по-прежнему строить граф социальных связей («кто с кем общается»), даже если содержимое переписки им недоступно, что делает данное решение неполным с точки зрения защиты от анализа метаданных [9].

WhatsApp. Данный мессенджер, принадлежащий компании Meta, внедрил сквозное шифрование на базе протокола Signal по умолчанию для всех чатов. Это стало значительным шагом вперед для массового рынка. Однако его архитектура остается полностью централизованной, что делает его инструментом для массового сбора метаданных. Компания имеет доступ к информации о том, кто, с кем и когда общается. Критической уязвимостью долгое время оставались резервные копии в облачные сервисы (Google Drive, iCloud), которые не шифровались E2EE-ключом, создавая легальный «бэкдор» для доступа к полной истории переписки. Несмотря на недавнее введение опционального шифрования бэкапов, проблема сбора метаданных и уязвимость к блокировкам остаются нерешенными.

Signal. Signal является эталоном в области защищенных коммуникаций, предлагая надежное сквозное шифрование с открытым исходным кодом. Однако его архитектура также централизована и требует обязательной привязки аккаунта к номеру телефона. Это создает две фундаментальные проблемы. Во-первых, привязка к номеру телефона позволяет легко деанонимизировать пользователя и построить граф его социальных связей,

что противоречит цели обеспечения полной приватности. Во-вторых, наличие центральных серверов делает сервис уязвимым к блокировкам на государственном уровне, что уже неоднократно происходило в разных странах.

Таким образом, даже самые продвинутые из существующих мессенджеров не предлагают комплексного решения. Они либо жертвуют конфиденциальностью ради удобства (Telegram), либо являются инструментом сбора метаданных (WhatsApp), либо уязвимы для блокировок и деанонимизации (Signal). Ни один из них не предоставляет механизмов защиты от физической компрометации устройства и последующего криминалистического анализа.

1.3 Сравнительный анализ и обоснование необходимости решения

Результаты критического анализа существующих решений сведены в Таблицу Е.1 в приложении Е для наглядного сравнения по ключевым параметрам безопасности.

Как следует из проведенного анализа, каждое из популярных приложений имеет одну или несколько фундаментальных уязвимостей, делающих их непригодными для использования в средах с высокими требованиями к приватности. Telegram жертвует конфиденциальностью ради удобства. WhatsApp является инструментом для массового сбора метаданных. Даже эталонный Signal уязвим для блокировок и деанонимизации.

Ни одно из рассмотренных решений не предлагает механизма защиты от физического принуждения и последующего криминалистического анализа, оставляя пользователя беззащитным в случае изъятия устройства. Этот системный пробел в безопасности современных коммуникационных инструментов и призван решить проектируемый мессенджер D-MASH, архитектура и протоколы которого будут рассмотрены в следующей главе.

2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ D-MASH

2.1 Общая архитектура системы

В основе D-MASH лежит гибридная архитектура, сочетающая мощь децентрализованной (P2P) mesh-сети с гибкостью современных веб-клиентов. Такой подход обеспечивает как высокую отказоустойчивость, так и доступность для конечного пользователя без компрометации безопасности.

Архитектура логически разделена на два фундаментальных компонента:

— **Магистральная сеть (Backbone Network):** Ядро системы, состоящее из независимых узлов (пиров). Каждый узел представляет собой программный Демон (Demon/Core), реализованный на Python с использованием FastAPI. Эти узлы соединяются друг с другом напрямую через WebSockets, формируя самоорганизующуюся и устойчивую к блокировкам mesh-сеть. Основная задача демона — служить «сортировочным центром»: он принимает зашифрованные пакеты, маршрутизирует их дальше по сети согласно протоколу «Tact». Демон не имеет доступа к содержимому сообщений и оперирует только обезличенными метаданными, хранящимися в «Blind Storage».

— **Клиентская среда (Client/UI):** Пользовательский интерфейс, который является точкой входа в систему. В данном проекте он реализован как прогрессивное веб-приложение (PWA). PWA-клиент не является постоянным узлом сети; вместо этого он подключается к одному из доверенных Демонов магистральной сети. Вся криптографическая логика, включая генерацию ключей (на базе WASM-модулей Argon2id и Kyber), шифрование/дешифрование сообщений и хранение истории переписки в

зашифрованной и ослепленной «Blind Storage» локальной базе данных (IndexedDB) выполняется исключительно на стороне клиента.

Такое разделение позволяет изолировать сложную сетевую и криптографическую логику от пользовательского интерфейса, повышая безопасность и модульность системы. Общая схема взаимодействия узлов представлена на Рисунке Г.1 в приложении Г.

Для обеспечения работы PWA-клиента, который не может поддерживать постоянное P2P-соединение, используется вспомогательный слой асинхронной доставки. Легковесные PHP-скрипты могут выступать в роли «почтовых ящиков» или сигнальных серверов. Они временно хранят зашифрованные «конверты» для офлайн-клиентов и могут инициировать push-уведомления (например, через Telegram-бота), не имея при этом доступа к ключам или содержимому. Это решает проблему «последней мили» для браузерных клиентов, сохраняя при этом безопасность E2EE.

Проблема трансляции сетевых адресов (NAT) остается актуальной для соединений между Демонами в магистральной сети. Для ее решения могут быть интегрированы стандартные протоколы, такие как STUN и TURN [7]. В то же время, соединение PWA-клиента с выбранным им узлом-демоном является классическим клиент-серверным WebSocket-соединением, что упрощает начальное подключение.

В рамках текущего прототипа основной фокус был сделан на валидации ключевых протоколов безопасности: маршрутизации внутри mesh-сети демонов и защищенного E2EE-канала между PWA-клиентами через эту сеть. Полноценная поддержка обхода NAT для демонов и развертывание публичной сети ретрансляторов являются ключевыми направлениями для дальнейшего развития проекта.

2.2 Гибридное криптографическое ядро

Криптографическое ядро является фундаментальной основой безопасности системы D-MASH. В отличие от классических мессенджеров, опирающихся исключительно на эллиптические кривые (ECC), в проекте спроектирована и реализована передовая гибридная криптосистема, обеспечивающая защиту от перспективных атак с использованием квантовых компьютеров.

Программная реализация ядра концептуально разделена на два уровня: серверный (на базе Python/PyNaCl) для слепой маршрутизации и высокопроизводительный клиентский. В стелс-клиенте (PWA) применена уникальная архитектура Dual WASM Engine — криптографические преобразования исполняются непосредственно в браузере через скомпилированные WebAssembly-модули на околонативной скорости, что исключает утечку секретов за пределы оперативной памяти клиента.

2.2.1 Используемые примитивы

Архитектура системы исключает использование устаревших и ресурсоемких алгоритмов (таких как RSA) в пользу высокопроизводительных современных стандартов. Это критически важно для работы в условиях ограниченных ресурсов мобильных устройств и выполнения в веб-среде.

Основными строительными блоками криптографического ядра являются:

— **Ed25519 и Curve25519:** Базовый уровень классической асимметричной криптографии [4]. Ed25519 применяется для строгой аутентификации узлов и пакетов (цифровые подписи), исключая возможность

подмены данных. Curve25519 используется для первичного протокола обмена ключами Диффи-Хеллмана (ECDH) при установке соединения.

— **ML-KEM (Kyber-768):** Инновационный стандарт постквантовой инкапсуляции ключей (KEM), утвержденный NIST в 2024 году [10]. Внедрен в клиентскую часть через нативный WASM-модуль. Он работает поверх ECDH, создавая "квантовый мост", который защищает переписку от угрозы «Store Now, Decrypt Later» (когда злоумышленник сохраняет зашифрованный трафик сейчас, чтобы расшифровать его квантовым компьютером в будущем).

— **XSalsa20-Poly1305:** Высокоскоростной потоковый шифр с аутентификацией, используемый для сквозного шифрования (E2EE) самого сетевого трафика. Гарантирует конфиденциальность и невозможность незаметной модификации перехваченных пакетов в Mesh-сети.

— **AES-GCM (256-bit):** Аппаратно-ускоренный алгоритм, применяемый исключительно на стороне клиента (интегрирован через стандарт Web Crypto API) для шифрования локального хранилища сообщений и базы контактов в рамках подсистемы «Blind Storage».

— **Argon2id:** Функция формирования ключа, устойчивая к атакам по сторонним каналам и аппаратному брутфорсу на GPU/ASIC. Исполняется в выделенном WASM-потоке. Используется для "выковывания" мастер-ключей и энтропии из пароля пользователя, обеспечивая высочайшую криптостойкость даже при вводе относительно коротких PIN-кодов.

— **BLAKE3 и SHA-256:** BLAKE3 применяется как сверхбыстрая криптографическая хеш-функция для маршрутизации и генерации "слепых" индексов (keyed-hash) в базах данных. SHA-256 задействован в каскадном формировании атомарных сессионных ключей в рамках протокола T-Ratchet.

2.2.2 Протокол сквозного шифрования T-Ratchet

Для обеспечения конфиденциальности переписки в децентрализованной среде D-MASH используется авторский криптографический протокол T-Ratchet (Time-Based Ratchet). Его архитектура специально спроектирована для работы в высоконагруженных и нестабильных P2P-сетях (Mesh-сетях), где пакеты могут доставляться с большой задержкой, с нарушенной последовательностью или теряться.

В отличие от классического протокола Double Ratchet (Signal Protocol), который является stateful (зависимым от состояния цепочки), T-Ratchet является stateless (бессостоятельным) протоколом. В Double Ratchet потеря одного сообщения в цепочке KDF приводит к рассинхронизации ключей и полной невозможности дешифровки последующего трафика без повторного выполнения процедуры рукопожатия (handshake). T-Ratchet фундаментально решает эту проблему за счет детерминированной генерации атомарных ключей на основе точных временных меток и энтропийных сдвигов.

Основные архитектурные отличия протокола T-Ratchet от классических реализаций:

— **Квантовый хендшейк**: В отличие от традиционного обмена Диффи-Хеллмана, первый обмен секретами защищен механизмом инкапсуляции ключа ML-KEM-768 (Kyber), исполняемым через WASM. Это делает невозможным вскрытие сессии методом "записи и последующей дешифровки" на квантовых компьютерах (угроза Store Now, Decrypt Later).

— **Атомная дискретность:** Для каждого отдельного сообщения вычисляется строго уникальный ключ шифрования. Ключ привязан не к порядковому номеру сообщения, а к точному миллисекундному таймстампу (Unix time в мс), присвоенному пакету в момент отправки.

— **Миллиардный сдвиг (Shift):** Между участниками согласуется секретное временное смещение в диапазоне $\pm 2^{31}$ миллисекунд. Это "искажает" реальное время пакета для криптографических вычислений, делая невозможным брутфорс ключа сторонним наблюдателем, даже если он знает точное время перехвата трафика.

— **Двойная печать**: Каждый пакет содержит строгую Ed25519-подпись, охватывающую внутренний nonce, шифротекст и сам миллисекундный таймстамп пакета. Это математически исключает атаки повтора (Replay attacks) и подмену времени внутри сети.

Сессионный ключ для каждого отдельного пакета генерируется каскадным методом из трех независимых компонентов:

— **Базовый общий секрет (Static Shared Secret):** Уникальный массив данных, вычисляемый один раз при установлении контакта между двумя конкретными узлами с помощью классического ECDH (Curve25519) и постквантового секрета (ML-KEM).

— **Предотсчитанный ключ (PSK - Pre-Shared Key):** 32-байтная динамическая криптографическая добавка, которая генерируется случайным образом и ротируется в рамках сессии. Обеспечивает механизм Post-Compromise Security (восстановление секретности после гипотетической компрометации прошлых ключей).

— **Искаженное время (Warped Time Entropy):** Энтропия, получаемая путем применения ресурсоемкой функции Argon2id к сумме точного миллисекундного времени отправки и секретного сдвига (Shift), где в качестве соли выступает часть ключа PSK.

Ключевой особенностью T-Ratchet является использование энтропийного сдвига (Epoch Shift). Это случайное значение в диапазоне от 0 до 10^9 которое согласовывается между участниками чата в момент

инициализации канала. Таким образом, результирующая формула генерации ключа через функцию деривации (KDF) выглядит следующим образом:

$$Key_{msg} = KDF(StaticSecret, PSK, timestamp + EpochShift)$$

Подобная архитектура обеспечивает три критических уровня защиты:

— **Миллисекундная уникальность и Perfect Forward Secrecy**: Благодаря привязке к миллисекунде отправки и использованию необратимых функций хеширования (Argon2id и SHA-256), каждое сообщение шифруется совершенно новым ключом. Компрометация одного ключа в оперативной памяти не позволит расшифровать ни предыдущие, ни последующие пакеты.

— **Защита от временного анализа:** Использование функции Argon2id для хеширования времени делает попытки прямого перебора (брутфорса) временных меток вычислительно нерентабельными. Атакующему пришлось бы выполнять ресурсоемкую функцию Argon2id для каждой миллисекунды в огромном диапазоне сдвига.

— **Асинхронность без потери данных**: Так как ключ для расшифровки детерминированно вычисляется принимающей стороной на основе открытого, но подписанного таймстампа, прикрепленного к конверту пакета, система способна корректно расшифровывать сообщения, пришедшие в любом порядке или после длительного пребывания в offline-хранилище (Mailbox).

Использование протокола T-Ratchet в сочетании с фиксированным размером пакетов в модуле Tact Engine превращает сетевой трафик в монотонный, неразличимый поток "белого шума", полностью защищенный от любых методов криминалистического, статистического или квантового анализа.

2.3 Протоколы сетевого уровня и защиты от атак

Помимо криптографической защиты содержания сообщений, архитектура D-MASH включает в себя протоколы, нацеленные на защиту от атак на сам сетевой уровень. Эти механизмы усложняют идентификацию участников, анализ их активности и попытки манипулирования сетью.

2.3.1 Идентификация узлов и противодействие Sybil-атакам

Одной из фундаментальных уязвимостей открытых децентрализованных сетей является «атака Сивиллы» (Sybil attack), при которой злоумышленник создает огромное количество поддельных узлов (идентичностей), чтобы получить непропорционально большое влияние на сеть, например, для перехвата трафика или нарушения маршрутизации.

Для противодействия этой угрозе в D-MASH введен механизм **Proof-of-Work (доказательство работы)**, который устанавливает [3] вычислительную «стоимость» для создания каждой новой идентичности. Идентификатором узла (Node ID) является его публичный ключ подписи Ed25519. Однако, чтобы этот ID был признан валидным в сети, он должен удовлетворять определенному условию: хэш от этого ID, вычисленный с помощью функции BLAKE3, должен начинаться с заранее определенного префикса (в текущей реализации — 0520).

Для нахождения такого ключа узел вынужден перебирать множество вариантов, выполняя вычислительно затратную работу. Для легитимного пользователя, создающего один узел, эти затраты (порядка нескольких десятков секунд процессорного времени) являются приемлемыми. Однако для злоумышленника, пытающегося создать тысячи «сивилл», совокупная стоимость такой операции становится непомерно высокой, делая атаку экономически невыгодной и практически нереализуемой.

Ключевым усилением защиты является то, что проверка соответствия идентификатора узла требованиям Proof-of-Work производится не только им самим при генерации, но и каждым другим узлом при попытке установки P2P-соединения. Таким образом, узел, не обладающий валидным, «вычисленным» идентификатором, не сможет подключиться к сети, что делает атаки с использованием поддельных ID неэффективными на сетевом уровне.

2.3.2 Протокол аутентификации P2P-соединений

Для противодействия атакам с подменой личности узла (Node Impersonation), при которых злоумышленник мог бы представиться другим участником сети, в D-MASH реализован протокол криптографического рукопожатия по принципу «вызов-ответ» (challenge-response). Процесс установки соединения выглядит следующим образом:

— Инициирующий узел «А» отправляет узлу «Б» свой идентификатор (публичный ключ) и криптографически стойкую случайную строку – «вызов»;

— Узел «Б», получив «вызов», сначала верифицирует соответствие ID узла «А» требованиям Proof-of-Work. В случае успеха, он подписывает полученный «вызов» своим приватным ключом и отправляет ответ, содержащий свой ID и сгенерированную подпись;

— Узел «А», получив ответ, в свою очередь верифицирует PoW узла «Б», а затем, используя публичный ключ узла «Б», проверяет подлинность подписи «вызова».

Только после успешного прохождения всех проверок соединение считается установленным. Данный механизм гарантирует, что каждая из сторон владеет приватным ключом, соответствующим ее публичному

идентификатору, что делает невозможной подмену личности на сетевом уровне.

2.3.2.1 Защита от атак «Человек посередине» (MITM)

Фундаментальной проблемой обмена ключами по открытому каналу (включая ECDH и ML-KEM) является уязвимость к атакам типа Man-in-the-Middle (MITM). Злоумышленник, контролирующий транзитный узел связи, может подменить публичные ключи собеседников своими, осуществляя прозрачную расшифровку трафика.

Для полного исключения данного вектора атак в D-MASH применяется концепция Out-of-Band (OOB) аутентификации. Первичный обмен идентификаторами узлов (64-значный публичный ключ) осуществляется вне сети интернет — посредством визуального сканирования QR-кода с экрана собеседника. Встроенный в PWA-клиент сканер считывает публичный ключ, после чего фоновый процесс двойного WASM-ядра автоматически инициирует квантовый хендшейк, гарантируя, что канал связи устанавливается именно с владельцем отсканированного ключа. Взломать такую схему удаленно математически невозможно.

2.3.3 Протокол обфускации трафика и маршрутизации «Tact Protocol»

Даже при использовании надежного сквозного шифрования, анализ метаданных сетевого трафика может раскрыть критически важную информацию: кто, с кем, когда и как часто общается. Для противодействия этому вектору атак, известному как Traffic Analysis, в D-MASH реализован **«Тактовый протокол» (Tact Protocol).** Его задача — не просто генерировать шум, а создать полноценный уровень обфускации, внутри которого скрывается реальная маршрутизация пакетов.

Принцип работы протокола заключается в устранении любых паттернов активности пользователя и превращении каждого узла в **непрозрачный сортировочный центр.**

— **Постоянный ритм и фиксированный размер**: Каждый узел сети с фиксированным, заданным интервалом (например, 1.5 секунды) формирует и отправляет каждому своему прямому соседу один большой унифицированный пакет-контейнер — «такт». Все такты имеют одинаковый, заранее определенный размер (например, 4096 байт).

— **Агрегация и маршрутизация:** Перед отправкой такта узел анализирует свою внутреннюю очередь исходящих пакетов **(outbox)**. В этой очереди находятся пакеты всех типов (**DATA**, **PROBE**, а также транзитные пакеты для других участников сети). Узел группирует все пакеты, предназначенные для конкретного соседа, и "упаковывает" их внутрь одного такта, адресованного этому соседу.

— **Маскировка бездействия:** Если для какого-либо соседа в очереди нет реальных пакетов, узел все равно формирует и отправляет ему такт. В этом случае такт заполняется зашифрованными случайными данными — «пустышкой» (**DUMMY**).

В результате для внешнего наблюдателя, анализирующего сетевой канал между двумя любыми узлами, весь трафик представляет собой абсолютно монотонный и неразличимый поток данных. Пакеты одинакового размера отправляются через равные промежутки времени, независимо от того, происходит ли реальное общение, идет ли поиск маршрута или узел просто находится в режиме ожидания.

Такая архитектура делает невозможным определение типа трафика, его конечного адресата или даже факта наличия полезной нагрузки внутри такта. Каждый узел эффективно выполняет функцию почтового сортировочного

центра, который получает и перераспределяет зашифрованные "контейнеры", не имея представления об их содержимом и конечном пункте назначения, что обеспечивает высочайший уровень защиты метаданных.

2.3.4 Прохождение межсетевых экранов и механизмы NAT Traversal

Для реализации стабильного прямого соединения (P2P) между узлами, находящимися за сложными корпоративными или домашними роутерами (NAT), в сетевой стек D-MASH интегрированы протоколы STUN (Session Traversal Utilities for NAT) и TURN (Traversal Using Relays around NAT).

Технология позволяет автоматически определять публичный IP-адрес узла и, при невозможности прямого проброса портов, устанавливать ретрансляционный канал через доверенные TURN-ноды. Использование протокола WebRTC обеспечивает надежную передачу потоковых данных (включая голосовой трафик GVP) в режиме реального времени, минимизируя задержки в Mesh-сети даже в условиях глубокой инспекции пакетов (DPI) провайдерами связи.

2.4 Система антикриминалистической защиты «Blind Storage»

Ключевой инновацией проекта D-MASH является система **«Blind Storage»** (Слепое хранилище), разработанная для противодействия наиболее прямой и критической угрозе — **физической компрометации устройства** и последующему криминалистическому анализу (anti-forensics). Задача системы — сделать невозможным восстановление графа социальных связей пользователя и истории сетевой активности, даже при получении полного доступа к файловой системе устройства.

2.4.1 Принцип работы: RAM-only Secret и Keyed Hashing

Ключ для дешифровки индексов базы данных (Secret Salt) выковывается в WASM-модуле Argon2id и существует исключительно в

оперативной памяти (RAM). Внедрена функция Wipe PIN, при вводе которой происходит мгновенная аннигиляция RAM-ключей и физическая перезапись секторов базы данных, превращающая историю переписки в невосстановимый шум за 0.8 секунды.

Механизм реализуется следующим образом:

— Г**енерация эфемерной соли:** При каждом запуске приложение генерирует криптографически стойкую случайную последовательность байт — **secret_salt** (секретная соль). Эта соль хранится в переменной в оперативной памяти и никогда не записывается на диск.

— Хеширование с ключом: Все чувствительные идентификаторы, которые могли бы раскрыть структуру сети (ID соседей, ID маршрутов, ID пакетов), не хранятся в базе данных в открытом виде. Вместо этого в качестве индексов и значений в таблицах хранятся их keyed-хэши, вычисленные с помощью функции **BLAKE3**, где в качестве ключа хеширования используется та самая secret_salt. Формула выглядит так: **db_value = BLAKE3(real_value, key=secret_salt).**

— **Шифрование "на себя":** Дополнительные метаданные, которые необходимо хранить в расшифрованном виде для работы узла (например, IP-адрес соседа), шифруются асимметрично с помощью **NaCl SealedBox** на публичный ключ самого узла. Расшифровать их может только данный узел, владеющий соответствующим приватным ключом.

2.4.2 Эффект "ослепления" данных

После выключения или перезагрузки устройства содержимое оперативной памяти, включая secret_salt, безвозвратно стирается. При следующем запуске будет сгенерирована уже совершенно новая, другая соль. В результате все keyed-хэши, хранящиеся в базе данных, становятся

"сиротами" — невозможно ни проверить их корректность, ни восстановить исходное значение, так как ключ для хеширования утерян.

Таким образом, для аналитика, получившего доступ к файлу базы данных, он будет представлять собой набор случайных, не связанных между собой хэшей и зашифрованных блоков. Ответить на ключевые вопросы криминалистического анализа — "С кем общался этот узел?", "Через какие узлы проходил трафик?", "Какие пакеты видел этот узел?" — становится математически невозможным.

Важно отметить, что после перезапуска системы старые, ставшие невалидными записи в таблицах маршрутизации и соседей сознательно не удаляются. Их сохранение служит цели усиления правдоподобного отрицания (plausible deniability). Для внешнего анализа база данных всегда выглядит наполненной зашифрованными данными, что создает «информационный шум» и не позволяет определить факт и время перезапуска системы, а также реальный объем ее сетевой активности в прошлом. Очистка устаревших записей происходит постепенно во время работы на основе их времени жизни (TTL).

Логическая схема работы системы «Blind Storage» представлена на Рисунке Г.2 в приложении Г.

2.4.3 «Маскировка клиентской среды через PWA»

Для исключения физического обнаружения коммуникационного ПО используется технология Progressive Web App (PWA).

— **Скрытая инсталляция:** Мессенджер устанавливается напрямую через браузер, обходя контроль централизованных магазинов приложений (App Store/Google Play).

— **Механизм декой-интерфейса (Decoy):** После установки приложение мимикрирует под системную утилиту «MathPro» (Калькулятор).

— **Аутентификация в поле вычислений калькулятора:** Реальный интерфейс системы активируется только после ввода управляющей последовательности в поле вычислений, что реализует концепцию Plausible Deniability (правдоподобного отрицания) наличия приложения на устройстве.

2.5 Комплекс протоколов экстренной связи (DSP Overlay)

Для обеспечения максимальной устойчивости системы, особенно в условиях полного или частичного отсутствия интернет-доступа, в D-MASH разработан и программно реализован уникальный комплекс из двух протоколов, использующих для передачи данных **голосовые каналы сотовой связи** или любые другие аналоговые средства (например, рации). Эти протоколы функционируют как "надстройка" (overlay) над основной сетью, используя технологии цифровой обработки сигналов (DSP).

Логические схемы работы DSP-протоколов представлены в Приложении А.

2.5.1 Phantom Call Protocol (PCP) для передачи цифровых данных

Протокол «Фантомный вызов» (PCP) является резервным транспортным модулем, предназначенным для гарантированной асинхронной доставки небольших пакетов цифровых данных.

Принцип работы: Критически важные данные (например, короткое текстовое сообщение, сигнал тревоги, GPS-координаты или ключ шифрования) кодируются в аудиосигнал с помощью устойчивой к помехам модуляции **MFSK (Multi-frequency Shift Keying) [2].** При MFSK-модуляции каждый небольшой фрагмент бинарных данных (например, 4 бита) преобразуется в аудио-тон определенной, заранее известной частоты из

набора 16 возможных. Полученная последовательность тонов формирует аудиопоток. Этот аудиосигнал может быть воспроизведен через динамик одного устройства и записан микрофоном другого в рамках обычного голосового вызова.

На приемной стороне полученный аудиосигнал обрабатывается с помощью алгоритма **Быстрого преобразования Фурье (FFT)**, который позволяет определить доминирующую частоту в каждый момент времени и, таким образом, декодировать аудио-тоны обратно в исходные бинарные данные. Для обеспечения целостности данных в протокол встроены механизмы контроля (заголовки, контрольные суммы CRC).

Скрытность и надежность: Для систем мониторинга оператора связи (СОРМ) такая передача выглядит как обычный короткий звонок с низким качеством связи или фоновым шумом, что делает его крайне сложным для обнаружения и блокировки. Протокол невосприимчив к DPI и любым методам блокировки интернет-трафика. Скорость передачи данных невысока (порядка 100-200 бит/с), однако этого более чем достаточно для доставки экстренных сообщений.

2.5.2 Ghost Voice Protocol (GVP) для защищенных голосовых вызовов

В отличие от PCP, протокол «Призрачный голос» (GVP) предназначен не для передачи данных, а для обеспечения конфиденциальности голосовых переговоров в реальном времени.

Принцип работы: Протокол использует не цифровое шифрование, а **аналоговое скремблирование** — обратимое искажение аудиосигнала речи, основанное на сессионном ключе. Ключевой особенностью GVP является его способность работать без предварительного обмена ключами

непосредственно перед вызовом. Вместо этого он элегантно использует уже существующий механизм T-Ratchet.

Процесс выглядит следующим образом:

— **Генерация базового ключа:** Отправляющая сторона, используя логику протокола T-Ratchet, вычисляет "базовый" ключ для текущей временной эпохи на основе общего секрета с получателем.

— **Создание сессионного ключа:** Генерируется криптографически случайная строка — «соль» (salt). Сессионный ключ для скремблирования одного конкретного голосового сообщения вычисляется как хэш от конкатенации базового ключа и этой соли: **SessionKey = HASH(BaseKey + Salt).**

— **Скремблирование и отправка:** Аудиопоток в реальном времени обрабатывается алгоритмами скремблирования, использующими **SessionKey**. В результате речь превращается в неразборчивый шум. Полученный скремблированный аудиопоток отправляется получателю **вместе с использованной «солью» в открытом виде.**

— **Восстановление речи:** Принимающая сторона получает пакет, извлекает из него «соль». Затем, используя ту же логику T-Ratchet, она самостоятельно вычисляет тот же самый "базовый" ключ для текущей (или соседних) временной эпохи. Объединив свой базовый ключ и полученную соль, она генерирует точную копию сессионного ключа и применяет алгоритмы дескремблирования в обратном порядке, восстанавливая исходную речь.

Методы скремблирования: В качестве алгоритмов искажения речи в реальном времени используются **инверсия спектра** (высокие частоты становятся низкими и наоборот) и **частотные скачки** (frequency hopping),

при которых частотные полосы спектра постоянно меняются местами по псевдослучайному закону, заданному сессионным ключом.

Таким образом, GVP является эффективным средством защиты от массовой автоматической прослушки, полностью интегрированным в асинхронную криптографическую модель D-MASH.

2.6 Моделирование угроз по фреймворку MITRE ATT&CK

Для систематизации векторов атак и оценки эффективности защитных механизмов D-MASH применялась международная база знаний тактик и техник злоумышленников MITRE ATT&CK (матрицы Enterprise и Mobile). Архитектура мессенджера спроектирована таким образом, чтобы митигировать (нейтрализовать) критические техники, представленные в приложении Д, кроме того, при проектировании системы особое в внимание было уделено следующим возможным рискам:

— Атаки на отказ в обслуживании (Denial of Service). Система может быть подвержена двум основным типам DoS-атак:

— **Исчерпание ресурсов через API.** API-эндпоинты, выполняющие ресурсоемкие операции (например, эндпоинт аутентификации, использующий Argon2id, или обработка аудио в GVP), уязвимы для атак через многократную отправку запросов. В производственной системе является целесообразным внедрение механизма ограничения частоты запросов (rate limiting) для предотвращения исчерпания вычислительных ресурсов.

— **Исчерпание ресурсов на сетевом уровне.** Несмотря на то, что протокол «Tact Protocol» подразумевает отправку пакетов фиксированного размера, злонамеренный клиент может проигнорировать это требование и отправлять пакеты максимального допустимого для WebSocket-сервера размера (в текущей реализации – до 10 МБ). Это создает нагрузку на память и

процессор. В качестве потенциального улучшения рекомендуется внедрить механизм проверки размера пакета на сетевом уровне до его полной обработки, с немедленным разрывом соединения при превышении установленного лимита.

Безопасность среды выполнения. Прототип системы разворачивается и тестируется с использованием технологии контейнеризации Docker. В целях упрощения тестового стенда, приложение внутри контейнера запускается от имени пользователя root. С точки зрения безопасности, это нарушает принцип наименьших привилегий. Для производственного развертывания обязательной мерой является создание в контейнере непривилегированного пользователя и запуск приложения от его имени, что значительно снизит потенциальный ущерб в случае компрометации процесса приложения.

2.6.1 Ограничения архитектуры и векторы атак вне скоупа

Несмотря на многоуровневую защиту, архитектура D-MASH имеет объективные ограничения и не гарантирует защиту от следующих векторов атак, которые выходят за рамки текущей модели угроз:

— Компрометация операционной системы (Zero-day, Kernel-level Spyware)

Программная среда (PWA и WASM) полностью доверяет операционной системе смартфона. В случае заражения устройства шпионским ПО правительственного уровня (класса Pegasus), имеющим доступ к ядру ОС (Ring 0), злоумышленник может перехватывать нажатия клавиш при вводе Master PIN, осуществлять скрытую запись экрана или считывать оперативную память до того, как данные будут зашифрованы алгоритмом Kyber/XSalsa20. Безопасность конечной точки (Endpoint Security) является зоной ответственности пользователя.

— Глобальные атаки изоляции (Eclipse Attack)

Хотя внедрение Proof-of-Work делает создание множества поддельных узлов (Sybil) экономически невыгодным для обычных злоумышленников, актор с государственными ресурсами (State-sponsored actor) или доступом к огромным вычислительным мощностям (ASIC/GPU-фермы) способен сгенерировать достаточное количество валидных идентификаторов. Это теоретически позволяет окружить целевой узел магистральной сети своими узлами, изолировав его от остальной сети (Eclipse Attack) для блокировки доставки пакетов. Данная проблема будет решаться в будущем путем внедрения децентрализованной репутационной системы (Web-of-Trust).

— Неотвратимое физическое принуждение

Если злоумышленник получает физический контроль над пользователем и заставляет его добровольно разблокировать устройство и ввести действительный Master PIN (до того, как пользователь успеет ввести Wipe PIN или сработают аппаратные триггеры G-Force/Flip-Lock), криптографическая защита теряет смысл. D-MASH предоставляет инструменты для экстренного уничтожения сессии, но их успешное применение зависит от реакции и психофизиологического состояния пользователя в критической ситуации.

Разработанные механизмы защиты являются нейтральным инструментом (Dual-use technology). Система D-MASH создана для обеспечения фундаментального права человека на приватность переписки, защиту профессиональной тайны журналистов и медицинских работников, а также для связи в условиях чрезвычайных ситуаций и техногенных катастроф

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРОТОТИПА

3.1 Стек технологий и выбор средств реализации

Учитывая гибридную архитектуру проекта, технологический стек был разделен на две независимые части: для клиентского приложения (PWA) и для магистрального узла маршрутизации.

Стек клиентской части (PWA Decoy):

— JavaScript (ES6+) и Service Workers: Основной язык логики интерфейса, обеспечения автономной работы и фонового опроса сети.

— WebAssembly (WASM): Используется для достижения нативной скорости криптографических вычислений в браузере. Внедрены скомпилированные C-библиотеки: kyber768.wasm для постквантового хендшейка и argon2.wasm для вычисления ключей.

— Web Crypto API и IndexedDB: Применяются для реализации системы «Blind Storage» — локального зашифрованного хранилища на базе AES-GCM.

— HTML5 WebRTC и Canvas API: Используются для захвата медиаданных, аналогового скремблирования аудио (GVP) и сканирования QR-кодов.

Стек магистрального узла (Demon) и ретрансляторов:

— Python 3.10 и FastAPI: Используются для создания высоконагруженного демона маршрутизации.

— WebSockets и aiosqlite: Обеспечивают асинхронную поддержку P2P-соединений и работу с базами данных узла.

— PHP: Применяется для реализации легковесных «бесшумных» ретрансляторов (Silent Relays).

3.2 Архитектура программных модулей

Исходный код проекта структурирован по принципу разделения ответственности (Separation of Concerns), что обеспечивает его модульность и упрощает дальнейшее развитие. Каждый модуль отвечает за определенный аспект функциональности системы.

Полное дерево файлов исходного кода представлено в Приложении Б.

Структура бэкенд-части проекта включает следующие ключевые файлы:

— **main.py** и **core.py**: Точка входа в приложение и ядро системы. Эти модули отвечают за инициализацию всех компонентов: запуск P2P-сервера, подключение к системной базе данных, запуск **Tact Engine**, а также управление жизненным циклом приложения и состоянием (state) ноды.

— **api.py**: Реализует API-эндпоинты, через которые пользовательский интерфейс отправляет команды (например, «отправить сообщение», «подключиться к узлу») и получает информацию о состоянии системы.

— **crypto.py**: Наиболее критичный модуль, содержащий полную реализацию криптографического ядра. В нем инкапсулированы классы **CryptoManager** (для пользовательского уровня) и **NodeCryptoManager** (для уровня демона), реализующие протокол T-Ratchet, систему «Blind Storage», генерацию ключей и подписей.

— **network.py**: Отвечает за всю сетевую логику. Содержит класс **P2PNode**, который управляет входящими и исходящими WebSocket-соединениями, а также реализует логику обработки и маршрутизации пакетов (**PROBE**, **DATA**).

— **database.py**: Модуль для взаимодействия с базами данных. Класс **DatabaseManager** предоставляет унифицированный интерфейс для работы как с пользовательской, так и с системной базой данных, инкапсулируя всю логику SQL-запросов и механизмов «Blind Storage».

— **dsp.py**: Модуль цифровой обработки сигналов. Содержит статичные методы для кодирования и декодирования аудиосигнала по протоколу PCP (MFSK), а также для скремблирования и дескремблирования голоса по протоколу GVP.

— **tact.py**: Реализация **Tact Engine**. Этот модуль в фоновом режиме с фиксированным интервалом опрашивает очередь исходящих пакетов и формирует «такты» для отправки соседям, обеспечивая обфускацию трафика.

Файлы PWA-клиента:

— **sw.js:** Service Worker, обеспечивающий кэширование и автономную работу клиента.

— **core_engine.js**: Ядро клиента, управляющее вызовами WASM-модулей, T-Ratchet сессиями и аппаратными сенсорами (гироскоп для Flip-Lock).

— **storage.js:** Модуль управления базой данных IndexedDB, реализующий логику "Слепого хранилища" на стороне клиента.

— **kyber768.js / kyber768.wasm**: Модули инкапсуляции постквантовых ключей.

Такая модульная архитектура позволяет изолированно разрабатывать и тестировать различные компоненты системы, а также упрощает их возможное обновление или замену в будущем.

Важной частью реализации API является строгая валидация всех входящих данных с помощью библиотеки Pydantic. Для полей, принимающих

такие данные, как сетевые адреса, применяются ограничения по длине и формату (с использованием регулярных выражений). Это предотвращает класс уязвимостей, связанных с передачей некорректных, чрезмерно длинных или вредоносных данных, и повышает общую надежность системы

3.3 Пользовательский интерфейс

Для взаимодействия с системой D-MASH был разработан клиентский веб-интерфейс, реализованный с использованием стандартных веб-технологий: HTML, CSS и JavaScript. Такой подход обеспечивает кроссплатформенность, позволяя получить доступ к мессенджеру через любой современный браузер без необходимости установки отдельного приложения. Скриншоты пользовательского интерфейса представлены в Приложении В.

Визуальный стиль интерфейса выполнен в минималистичной эстетике «киберпанк-терминала», что подчеркивает фокус проекта на безопасности и технологичности.

Основные функции, реализованные в интерфейсе:

— **Аутентификация**: Пользователь входит в систему, указывая имя пользователя и пароль, из которых детерминированно генерируются его криптографические ключи. Это позволяет получить доступ к своему аккаунту с любого устройства, не храня на нем закрытые ключи в открытом виде.

— **Управление контактами и подключениями:** Интерфейс позволяет добавлять новые контакты по их публичным ключам (ID), а также инициировать P2P-подключения к другим узлам сети по их IP-адресам.

— **Обмен сообщениями**: Реализован базовый функционал чата, включая отображение истории сообщений и отправку новых.

— **Отображение статуса:** В интерфейсе в реальном времени отображается статус подключения к сети и количество активных соседей, что дает пользователю представление о состоянии системы.

Взаимодействие между фронтендом (браузером) и бэкендом (локальным демоном) происходит асинхронно через вызовы к API, реализованному на FastAPI. Это обеспечивает отзывчивость интерфейса и быструю доставку обновлений.

Защита от XSS-атак. Особое внимание было уделено безопасности клиентской части. Для предотвращения атак межсайтового скриптинга (Cross-Site Scripting, XSS), при которых вредоносный HTML или JavaScript-код, отправленный в сообщении, мог бы выполниться в браузере получателя, реализован механизм санитаризации. Весь текстовый контент, получаемый от других пользователей, перед отображением на странице проходит обработку, в ходе которой специальные символы HTML преобразуются в их безопасные эквиваленты. Это гарантирует, что любой потенциально вредоносный код будет отображен как обычный текст, а не исполнен браузером.

3.3.1 Архитектура стелс-клиента на базе PWA

Клиент реализован как PWA-приложение с глубокой мимикрией под инженерный калькулятор. Доступ к терминалу связи открывается только после ввода мастер-кода в строке вычислений. Реализован G-Force Wipe Trigger: акселерометр отслеживает резкое векторное ускорение (рывок телефона из рук), что ведет к мгновенной блокировке и стиранию ключей сессии.

Критическим элементом является использование Service Workers. Это изолированные скрипты, которые работают в фоновом режиме даже при закрытой вкладке мессенджера. Они выполняют:

— **Фоновый опрос (Polling)** входящих пакетов из распределенной очереди.

— **Анонимные PUSH-уведомления (Telegram Beacon):** Поскольку фоновые процессы браузера (Service Workers) часто принудительно завершаются операционной системой для экономии батареи, разработана система внешних маяков. Узел-ретранслятор при получении пакета связывается с Telegram-ботом, отправляя триггер по зашифрованному идентификатору (хэшу). Пользователь получает системное уведомление через Telegram, при этом сам сервер Telegram не имеет доступа ни к IP-адресам, ни к содержимому, ни к графу связей, так как оперирует исключительно одноразовыми хэшами.

— **Автономный кэш ключей**, позволяющий мгновенно инициировать T-Ratchet сессию оффлайн.

3.4 Окружение развертывания и кроссплатформенность

Архитектура D-MASH разделена на фронт-офис и ядро маршрутизации:

— **Магистральный узел (Node Engine):** Контейнеризирован через Docker Compose, что обеспечивает мгновенную миграцию ноды между серверами и стабильность P2P-линков.

— **Клиентская среда:** Базируется на Service Workers, которые управляют кешем и входящими пакетами в фоновом режиме, позволяя пользователю сохранять автономность.

Реализован протокол бесшовной миграции: при полной потере ключей (Hard Reset) или смене устройства, система автоматически пересогласовывает параметры T-Ratchet при первом же «такте» обмена данными с доверенным соседом, что критически важно для Mesh-сетей без интернета.

4 ТЕСТИРОВАНИЕ И РЕЗУЛЬТАТЫ

4.1 Методология интеграционного стресс-тестирования

Для проверки и валидации ключевых архитектурных решений, реализованных в D-MASH, был выбран подход **интеграционного стресс-тестирования.** Стандартное модульное тестирование (unit-testing) является недостаточным для систем с асинхронной и децентрализованной природой, так как не позволяет оценить корректность взаимодействия множества независимых узлов в динамической сетевой среде.

Целью тестирования являлась проверка комплексной работоспособности системы в условиях, приближенных к реальным. Основные задачи теста включали:

— валидацию механизма P2P-соединений между узлами;

— проверку корректности работы протокола обнаружения маршрута (PROBE) в сети из множества узлов;

— подтверждение успешной доставки DATA-пакетов по многошаговому (multi-hop) маршруту;

— оценку общей стабильности системы при симулированной нагрузке.

4.2 Тестовый стенд на базе Docker

Для проведения тестирования был разработан специальный скрипт **stress_test.py**, который использует технологию контейнеризации **Docker** и инструмент **Docker Compose**. Такой подход позволяет на одной физической машине развернуть изолированную, полностью воспроизводимую виртуальную сеть, состоящую из множества независимых узлов D-MASH, каждый из которых работает в своем собственном контейнере.

Тестовый стенд автоматически выполняет следующие шаги:

— **Генерация конфигурации:** Скрипт генерирует файл docker-compose.yml для запуска сети из заданного числа узлов (в данном случае, **30 узлов**).

— **Развертывание сети:** С помощью Docker Compose происходит сборка образов и запуск 30 контейнеров, каждый из которых представляет собой полноценный узел D-MASH.

— **Инициализация узлов**: Скрипт выдерживает паузу, необходимую для того, чтобы каждый узел успел запуститься и сгенерировать свою идентичность посредством механизма Proof-of-Work.

— **Построение топологии**: Для создания mesh-сети скрипт автоматически отправляет API-запросы на подключение узлов друг к другу, формируя линейную цепь (node1 -> node2 -> ...) и добавляя несколько случайных связей (EXTRA_LINKS = 3) для создания более сложной и реалистичной топологии.

4.3 Анализ результатов тестирования

После развертывания сети скрипт **stress_test.py** выполняет серию тестовых прогонов, каждый из которых симулирует отправку сообщения между двумя случайными узлами в сети. Процесс наглядно демонстрирует работоспособность всех ключевых протоколов:

Фаза 1: **Обнаружение маршрута (PROBE)**. Отправляющий узел, не имея прямого пути к получателю, инициирует отправку PROBE-пакета. Вывод скрипта в консоли в реальном времени отслеживает распространение этого пакета по сети, показывая, как он лавинно передается от узла к узлу, пока не достигает целевого.

Фаза 2: Симметричный ответ. Получив PROBE, целевой узел инициирует отправку ответного PROBE для установления двунаправленного маршрута.

Фаза 3: Передача данных (DATA). После успешного установления маршрута, отправляющий узел посылает сообщение, инкапсулированное в DATA-пакет. Скрипт подтверждает, что система корректно переключилась на адресную (unicast) передачу по построенному туннелю, а не использует лавинную рассылку повторно.

Фаза 4: Верификация доставки. В завершение теста скрипт отправляет API-запрос к узлу-получателю и проверяет наличие доставленного сообщения в его локальной базе данных, подтверждая тем самым успешное завершение всего цикла: от отправки до получения и расшифровки.

Результаты всех тестовых прогонов показали **100% успешную доставку сообщений** между случайными узлами в сети из 30 пиров. Это подтверждает корректность реализации P2P-логики, протоколов маршрутизации и сквозного шифрования, и доказывает жизнеспособность выбранной архитектуры.

Проведенное исследование доказывает готовность D-MASH к работе в условиях постквантовой эпохи благодаря интеграции стандарта NIST 2024 (ML-KEM). Проект реализован как кроссплатформенное WASM-приложение, что позволяет в кратчайшие сроки развернуть нативный APK-стек с расширенными DSP-модулями для прямого управления аудио-трактом. Система является законченным инструментом автономной защиты информации, превосходящим современные централизованные аналоги по уровням анонимности и криптостойкости.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного творческого проекта была успешно решена поставленная задача: спроектирована, детально описана и программно реализована архитектура децентрализованного мессенджера D-MASH, обладающего уникальными, комплексно реализованными свойствами безопасности, а также проведен анализ дополнительных векторов атак и предложены пути их нейтрализации.

В результате проделанной работы были получены следующие ключевые результаты:

— **Проведен анализ,** выявивший фундаментальные уязвимости централизованных архитектур популярных мессенджеров (Telegram, WhatsApp, Signal), включая проблемы с E2EE по умолчанию, сбором метаданных, уязвимостью к блокировкам и отсутствием защиты от физического принуждения.

— **Разработана система «Blind Storage»** – инновационный механизм правдоподобного отрицания на основе RAM-only ключей, который делает невозможным восстановление социального графа при физической компрометации устройства и скрывает факт перезапуска системы за счет сохранения невалидных данных. Система также дополнена аппаратными триггерами (G-Force Wipe, Flip-Lock).

— **Разработан протокол обфускации трафика «Tact Protocol»**, который эффективно противодействует анализу сетевых метаданных путем создания монотонного, неразличимого потока зашифрованных данных, а также оптимизирует доставку сообщений в mesh-сетях.

— **Разработан и реализован комплекс протоколов экстренной связи** через голосовые каналы сотовой сети, включающий «PhantomCall Protocol»

для передачи цифровых данных и «GhostVoice Protocol» для проведения конфиденциальных переговоров. Этот комплекс позволяет сохранять связь даже в условиях полного отключения интернета, что является уникальной особенностью проекта.

— **Подтверждена эффективность протокола T-Ratchet,** обеспечивающего генерацию уникального ключа для каждого сообщения на основе его миллисекундного таймстампа, что гарантирует максимальный уровень защиты метаданных и PFS.

— **Внедрено гибридное постквантовое ядро:** Реализован протокол обмена ключами на основе стандарта NIST ML-KEM (Kyber-768), исполняемый в браузере через высокопроизводительный WASM-модуль, что обеспечивает защиту от атак с использованием квантовых компьютеров.

— **Разработан полноценный стелс-клиент:** Создано PWA-приложение с полной мимикрией под системную утилиту и функцией аппаратной привязки ключей через WebAuthn (Knox PRF), что кардинально повышает защиту от несанкционированного доступа и клонирования устройства.

Проект находится в фазе активной реализации: ведется доработка Service Workers для усиления автономности клиентской части в режиме оффлайн.

Таким образом, представленный проект D-MASH, даже на уровне концепции и прототипа, демонстрирует возможность создания коммуникационной системы нового поколения, где безопасность, приватность и устойчивость к цензуре являются не опциональными функциями, а базовым, неотъемлемым принципом архитектуры.

5. ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОЕКТА

Представленный проект является полностью функциональным прототипом, демонстрирующим жизнеспособность предложенной архитектуры. Дальнейшее развитие системы D-MASH может включать следующие ключевые направления:

— **Реализация групповых коммуникаций.** Разработка и внедрение протокола для создания защищенных групповых чатов в децентрализованной среде. В качестве основы для исследования может быть рассмотрен современный стандарт Messaging Layer Security (MLS), адаптированный для работы в P2P-сетях без центрального сервера.

— **Внедрение системы децентрализованного доверия.** Интеграция репутационной системы, например, на основе концепции «сети доверия» (Web-of-Trust), для верификации публичных ключей контактов без участия централизованных удостоверяющих центров. Это позволит противодействовать атакам типа «человек-посередине» (MITM) при первом контакте с пользователем.

— **Дальнейшая оптимизация PWA-клиента** для мобильных платформ, включая улучшение энергопотребления фоновых процессов (Service Workers) и исследование компиляции в нативные приложения с помощью фреймворков, таких как Capacitor или Tauri, для более глубокой интеграции с ОС.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – Введ. 2018-07-01. – Москва : Стандартинформ, 2017. – 24 с.
2. Таненбаум, Э. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл ; пер. с англ. – 5-е изд. – Санкт-Петербург : Питер, 2012. – 960 с. – ISBN 978-5-459-00342-0.
3. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер ; пер. с англ. – 2-е изд. – Москва : Триумф, 2003. – 816 с. – ISBN 5-89392-055-4. (дата обращения: 15.12.2025).
4. Bernstein, D. J. Curve25519: new Diffie-Hellman speed records / D. J. Bernstein // Public Key Cryptography-PKC 2006. – Springer Berlin Heidelberg, 2006. – P. 207–228. – ISBN 978-3-540-33852-9. (дата обращения: 15.12.2025).
5. Marlinspike, M. The Signal Protocol / M. Marlinspike, T. Perrin. – Текст : электронный // Signal Messenger. – 2016. – URL: https://signal.org/docs/specifications/doubleratchet/ (дата обращения: 10.11.2025).
6. O'Donoghue, J. The BLAKE3 Cryptographic Hash Function / J. O'Donoghue, J. P. Aumasson, S. Neves, Z. Wilcox-O'Hearn. – Текст : электронный // GitHub repository. – 2020. – URL: https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf (дата обращения: 15.12.2025).
7. Rosenberg, J. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN) / J. Rosenberg, R.

Mahy, P. Matthews, D. Wing. – Текст : электронный // IETF. – 2010. – (RFC 5766). – URL: https://datatracker.ietf.org/doc/html/rfc5766 (дата обращения: 15.12.2025).

8. Vaudenay, S. A Classical Introduction to Cryptography: Applications for Communications Security / S. Vaudenay. – New York : Springer, 2006. – 349 p. – ISBN 978-0-387-25464-7. (дата обращения: 15.12.2025).
9. Jakobsen, J. On the CCA (in)security of MTProto / J. Jakobsen, C. Orlandi // IEEE European Symposium on Security and Privacy. – 2016. (дата обращения: 15.12.2025).
10. National Institute of Standards and Technology (NIST). FIPS PUB 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard [Электронный ресурс] // NIST. – 2024. – URL: https://csrc.nist.gov/pubs/fips/203/final (дата обращения: 20.02.2026).

ПРИЛОЖЕНИЯ

Приложение А

Схемы работы DSP-протоколов

На рисунках А.1 и А.2 представлены логические схемы, иллюстрирующие принцип работы реализованных в проекте протоколов экстренной связи, использующих технологии цифровой обработки сигналов.

Рисунок А.1 – Принцип работы Phantom Call Protocol (PCP)

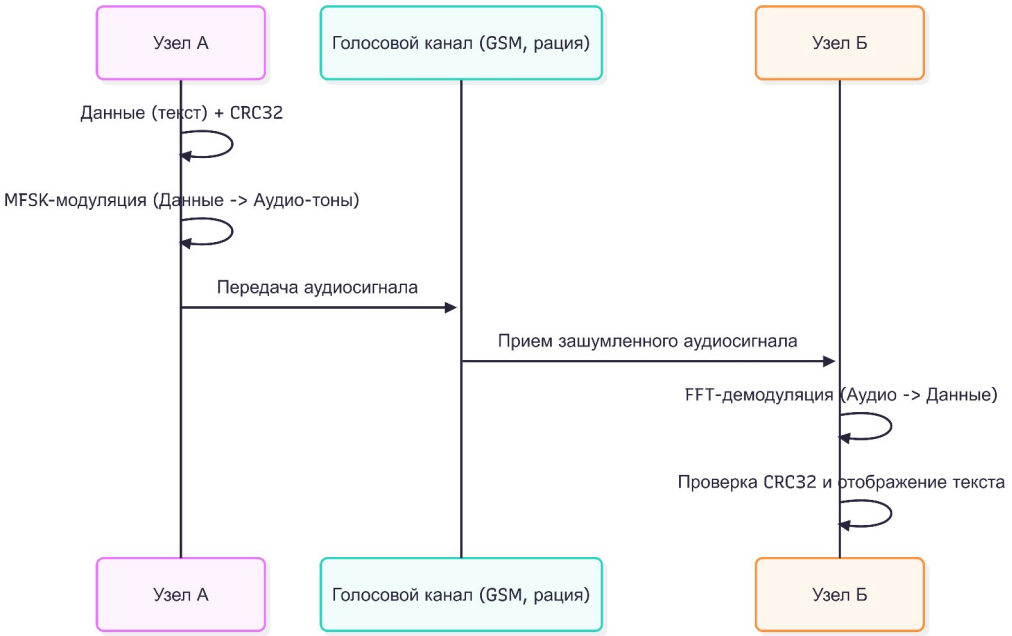
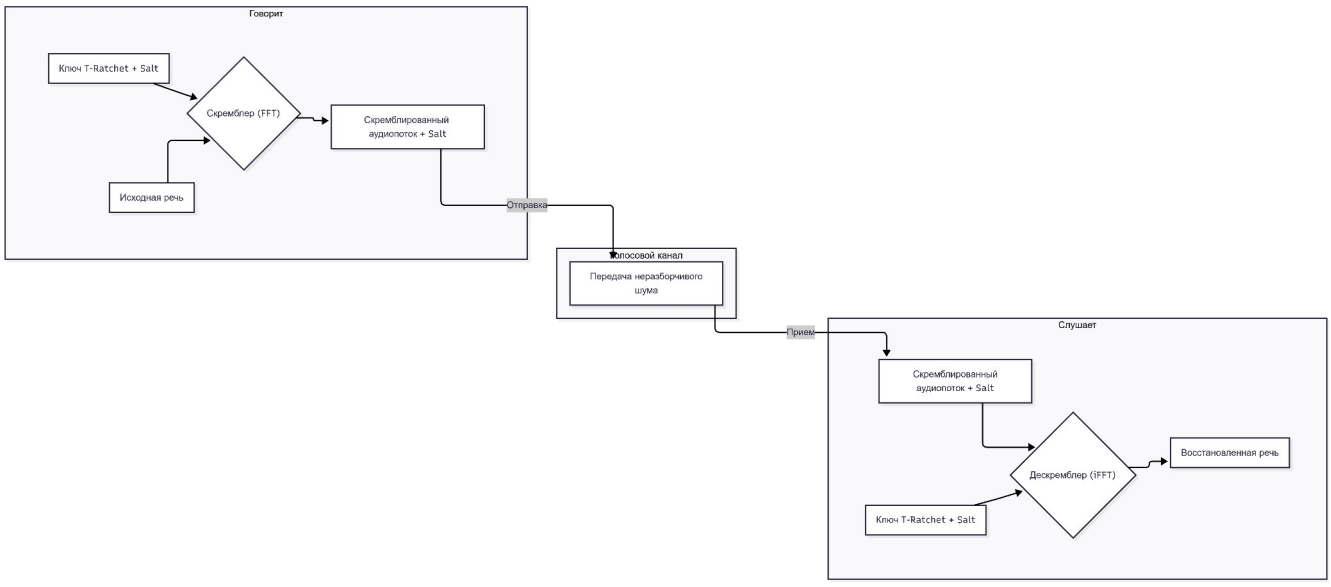


Рисунок А.2 – Принцип работы Ghost Voice Protocol (GVP)



Приложение Б

Структура проекта

Структура исходного кода проекта организована для логического разделения ответственности между компонентами системы. Актуальная структура файлов проекта представлена на Рисунке Б.1.

Рисунок Б.1 – Структура файлов проекта D-MASH (Docker-стек)

```
D-MASH/
├── client/
│   ├── backend/          # Серверная часть (демон)
│   │   ├── __init__.py
│   │   ├── api.py        # API для взаимодействия с UI
│   │   ├── core.py       # Ядро приложения, управление состоянием и PoW
│   │   ├── crypto.py     # Криптографическое ядро (T-Ratchet, Blind Storage)
│   │   ├── database.py   # Менеджер баз данных
│   │   ├── dsp.py        # Модуль цифровой обработки сигналов (PCP, GVP)
│   │   ├── main.py       # Точка входа, запуск FastAPI-сервера
│   │   ├── network.py    # Управление P2P-сетью и маршрутизацией
│   │   └── tact.py       # Реализация "Тактового протокола"
│   │
│   ├── docker/           # Файлы для контейнеризации
│   │   ├── messenger.Dockerfile
│   │   └── start.sh
│   │
│   ├── frontend/         # Клиентская часть (веб-интерфейс)
│   │   ├── auth/
│   │   │   └── login.html
│   │   └── chat/
│   │       ├── chat.css
│   │       ├── chat.html
│   │       └── chat.js
│   │
│   ├── requirements.txt          # Список зависимостей проекта
│   ├── stress-test-compose.yml   # Конфигурация, генерируемая stress_test.py
│   └── user-docker-compose.yml   # Docker Compose для ручного запуска сети
│
└── stress_test.py        # Скрипт интеграционного стресс-тестирования
```

Рисунок Б.2 – Структура файлов проекта D-MASH (PWA-стек)

```
D-MASH PWA/
├── api
│   ├── api.php
│   └── tg_webhook.php
├── not_messenger
│   ├── js
│   │   ├── vendor
│   │   │   ├── argon2-bundled.min.js
│   │   │   ├── argon2.wasm
│   │   │   ├── html5-qrcode.min.js
│   │   │   ├── kyber768.js
│   │   │   ├── kyber768.wasm
│   │   │   ├── nacl-fast.min.js
│   │   │   ├── nacl-util.min.js
│   │   │   └── qrcode.min.js
│   │   ├── core_engine.js
│   │   ├── storage.js
│   │   └── ui_logic.js
│   ├── calc_icon.png
│   ├── index.html
│   ├── manifest.json
│   └── sw.js
└── welcome.html
```

Полный исходный код проекта доступен в публичном репозитории на платформе GitHub по адресу: https://github.com/Seriy615/D-MASH

Приложение В

Скриншоты пользовательского интерфейса

Рисунок В.1 – Экран входа в систему, выполненный в стилистике терминала

Рисунок В.2 – Основной интерфейс чата с панелью выбора протоколов отправки (STANDARD, PCP, GVP)

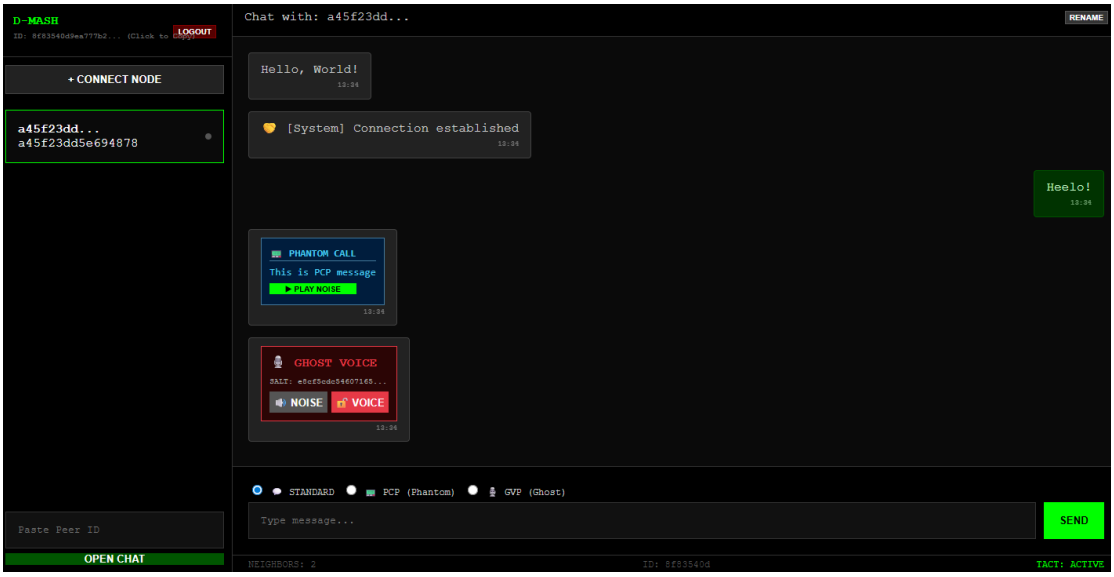


Рисунок Г.1 – Общая архитектура взаимодействия узлов D-MASH

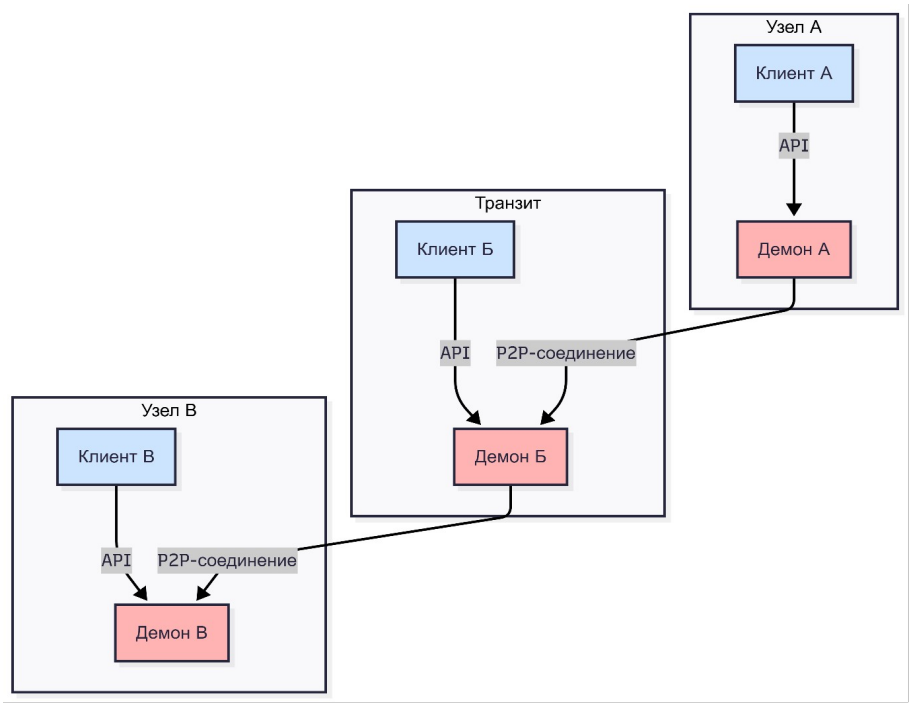
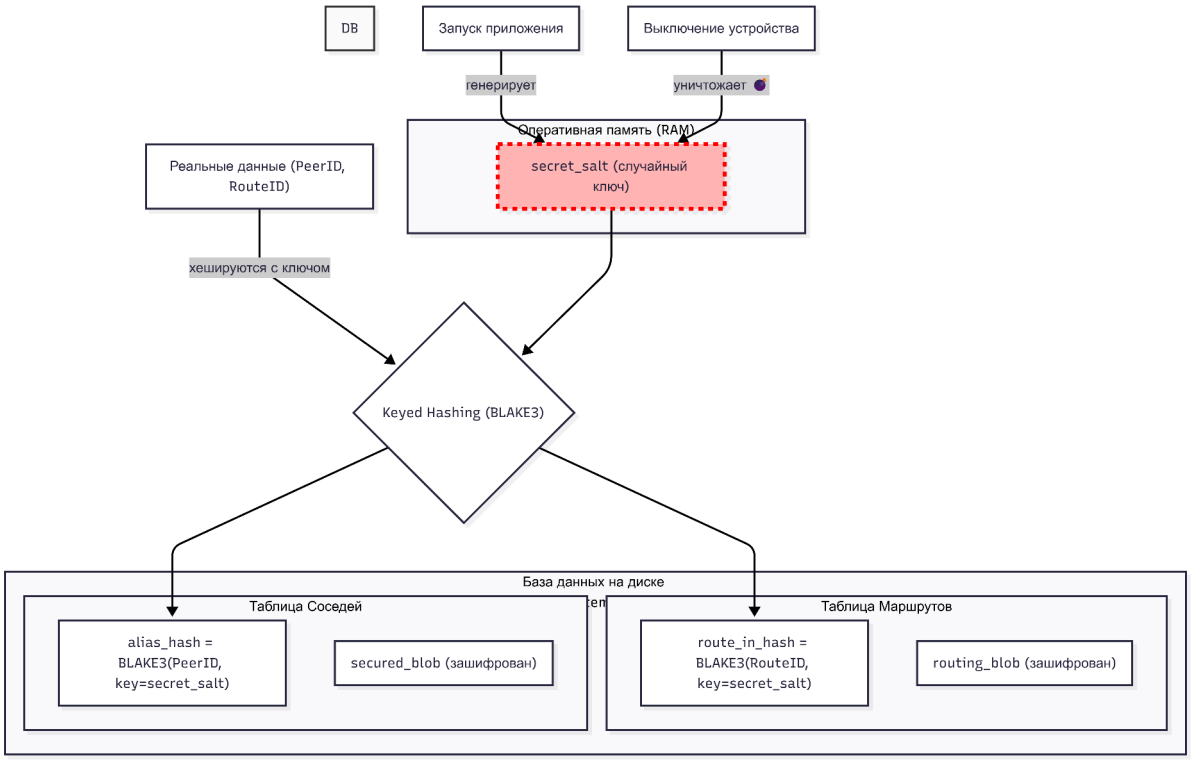


Рисунок Г.2 – Логическая схема работы системы «Blind Storage» -



Приложение Д

Тактики MITRE ATT&CK

1. Network Traffic Discovery / Traffic Analysis (T1040 / T1020)

Суть угрозы: Перехват и анализ сетевого трафика (например, с помощью систем DPI) для выявления факта коммуникации, определения адресатов и сбора метаданных.

Митигация в D-MASH: Протокол обфускации «Tact Engine» генерирует константный поток пакетов фиксированного размера, скрывая реальную сетевую активность за «белым шумом». Постквантовый протокол T-Ratchet исключает возможность расшифровки даже при накоплении огромных массивов трафика (защита от вектора Store Now, Decrypt Later).

2. Data from Local System (T1005 / Mobile T1409)

Суть угрозы: Физическое изъятие устройства с последующим криминалистическим дампом файловой системы для извлечения истории переписки и социального графа (контактов).

Митигация в D-MASH: Технология «Blind Storage». База данных (IndexedDB) хранит только зашифрованные блобы и слепые хэши. Мастер-ключ и secret_salt генерируются через Argon2id и существуют исключительно в RAM. При выключении устройства или срабатывании аппаратных триггеров (G-Force Wipe / Flip-Lock) данные безвозвратно превращаются в криптографический мусор.

3. Steal Application Access Token / Credentials (T1528 / Mobile T1412)

Суть угрозы: Клонирование состояния приложения или извлечение ключей доступа вредоносным ПО для последующего входа с другого устройства.

Митигация в D-MASH: Аппаратная привязка ключей (Hardware Binding). Использование стандарта WebAuthn (включая расширение Knox PRF) позволяет зашифровать локальные секреты ключом, который жестко вшит в защищенный сопроцессор (TEE/Secure Enclave) конкретного смартфона. Клонирование файлов PWA на другое устройство не даст злоумышленнику доступа к профилю.

4. Hide Artifacts / Masquerading (T1564 / T1036)

Суть защиты: В отличие от классического применения этих техник атакующими, D-MASH использует их в интересах пользователя для защиты от физического обнаружения (Defense Evasion). Установка в виде PWA-клиента в обход магазинов приложений и мимикрия интерфейса под системный калькулятор («PWA Decoy») позволяют реализовать концепцию правдоподобного отрицания (Plausible Deniability).

Приложение Е

Таблица Е.1 - Сравнительный анализ мессенджеров по ключевым параметрам безопасности

Критерий	Telegram	WhatsApp	Signal	D-MASH (проект)
Архитектура	Централизованная	Централизованная	Централизованная	P2P
E2EE по умолч.	Нет (только Secret Chats)	Да	Да	Да (T-Ratchet на базе ECC)
Привязка к номеру	Да	Да	Да	Нет (Identity на базе ключей)
Сбор метаданных	Высокий	Высокий	Минимальный	Теоретический минимум
Устойчивость к блок.	Низкая	Низкая	Низкая	Высокая (P2P + PCP)
Защита от анализа трафика	Нет	Нет	Частично	Да (Tact Engine)
Защита от физ. компрометации	Нет	Нет	Нет	Да (Blind Storage)