F.E.A.R. Project - Руководство пользователя

Fully Encrypted Anonymous Routing

Версия документации: 1.0 **Дата:** 2025 **Автор:** Shchuchkin E. Yu.

Содержание

Введение

F.E.A.R. (Fully Encrypted Anonymous Routing) — это кроссплатформенный мессенджер с открытым исходным кодом, разработанный для обеспечения максимальной конфиденциальности и безопасности коммуникаций.

Ключевые особенности

- **Сквозное шифрование (E2EE):** Все сообщения шифруются на устройстве отправителя и расшифровываются только на устройстве получателя
- **Защита от прослушивания:** Сервер не имеет доступа к содержимому сообщений
- **Комнаты с паролями:** Доступ к комнате имеют только пользователи, знающие правильный ключ
- **Безопасный обмен ключами:** Встроенный механизм обмена ключами по протоколу Диффи-Хеллмана
- **Передача файлов:** Отправка файлов с шифрованием и проверкой целостности
- **Аудиозвонки:** Зашифрованные голосовые вызовы
- **Открытый исходный код:** Полная прозрачность реализации

Архитектура и безопасность

Модель безопасности

F.E.A.R. использует клиент-серверную архитектуру с **сквозным шифрованием**:

```
[Клиент 1] <--зашифровано--> [Сервер] <--зашифровано--> [Клиент 2]
                                |
                          (не видит контент)
```

Важно: Сервер выполняет только функции маршрутизации и **НЕ имеет доступа** к:

- Содержимому сообщений
- Ключам шифрования комнат
- Расшифрованным данным

Криптографические алгоритмы

1. AES-256-GCM (Advanced Encryption Standard)

Назначение: Шифрование всех сообщений в комнатах

Параметры:

- Длина ключа: 256 бит (32 байта)
- Режим: GCM (Galois/Counter Mode)
- Длина nonce: 96 бит (12 байт)
- Длина тега аутентификации: 128 бит (16 байт)

Как это работает:

```
Открытый текст + Ключ комнаты + Nonce → AES-256-GCM → Зашифрованный текст + Тег
```

AES-256-GCM обеспечивает:

- **Конфиденциальность:** Данные невозможно прочитать без ключа
- **Аутентификацию:** Невозможно подделать или изменить сообщение незаметно
- **Защиту от повторов:** Каждое сообщение использует уникальный nonce

2. Curve25519 (Elliptic Curve Diffie-Hellman)

Назначение: Безопасный обмен ключами между пользователями

Параметры:

- Эллиптическая кривая: Curve25519
- Длина публичного ключа: 32 байта

- Длина секретного ключа: 32 байта

Протокол обмена:

```
Пользователь A генерирует: (публичный_A, секретный_A)
Пользователь B генерирует: (публичный_B, секретный_B)

A отправляет публичный_A → B
B отправляет публичный_B → A

Общий секрет A = ECDH(секретный_A, публичный_B)
Общий секрет B = ECDH(секретный_B, публичный_A)

Общий_секрет_A == Общий_секрет_B
```

Curve25519 защищает от:

- Атак "человек посередине" (если обмен выполнен по защищенному каналу)
- Компрометации ключей шифрования при перехвате сетевого трафика

3. XChaCha20-Poly1305

Назначение: Шифрование сообщений при обмене ключами

Параметры:

- Алгоритм: XChaCha20 (потоковое шифрование)
- Аутентификация: Poly1305 MAC
- Длина nonce: 192 бит (24 байта)
- Длина ключа: 256 бит (32 байта)

Используется в модуле key-exchange для защиты ключа комнаты при передаче через открытый канал.

4. CRC32 (Cyclic Redundancy Check)

Назначение: Проверка целостности передаваемых файлов

CRC32 не является криптографическим алгоритмом, но обеспечивает:

- Обнаружение случайных ошибок при передаче
- Проверку, что файл был передан полностью и без повреждений

Структура протокола

Каждое сообщение в F.E.A.R. имеет следующий формат:

```
[2 байта: длина_имени_комнаты]
[имя_комнаты]
[2 байта: длина_имени_отправителя]
[имя_отправителя]
```

```
[2 байта: длина_nonce]
[nonce (12 байт)]
[1 байт: тип_сообщения]
[4 байта: длина_зашифрованных_данных]
[зашифрованные_данные + тег_аутентификации]
```

Типы сообщений:

- `0` - Текстовое сообщение
- `1` - Начало передачи файла
- `2` - Фрагмент файла
- `3` - Конец передачи файла
- `4` - Список участников (служебное)

Требования к системе

Минимальные требования

- **Операционная система:**
  - Windows 10/11 (64-bit)
  - Linux (Ubuntu 20.04+, Debian 11+, Fedora 35+)
  - macOS 10.15+ (Catalina или новее)
- **Процессор:** Intel Core i3 / AMD Ryzen 3 или эквивалент
- **Оперативная память:** 2 ГБ
- **Свободное место на диске:** 200 МБ
- **Сеть:** Подключение к Интернету или локальной сети

Для сборки из исходников

- **Git** (для клонирования репозитория)
- **CMake** версии 3.15 или выше
- **Компилятор C++17:**
  - GCC 8+ (Linux)
  - Clang 7+ (macOS/Linux)
  - MSVC 2019+ (Windows)
  - MinGW-w64 8+ (Windows)
- **Библиотеки:**
  - **libsodium** (криптография)
  - **Qt 6.2+** (графический интерфейс)
  - **PortAudio** (аудио)
  - **Opus** (кодек для аудио)
- **Система управления пакетами (опционально):**
  - vcpkg
  - Conan

Сборка проекта

Linux

Установка зависимостей (Ubuntu/Debian)

```
sudo apt update
sudo apt install -y git cmake build-essential pkg-config \
    libsodium-dev qt6-base-dev portaudio19-dev libopus-dev
```

Установка зависимостей (Fedora)

```
sudo dnf install -y git cmake gcc-c++ libsodium-devel \
    qt6-qtbase-devel portaudio-devel opus-devel
```

Сборка

```
# Клонирование репозитория
git clone https://github.com/shchuchkin-pkims/fear.git
cd fear

# Сборка всех компонентов
make

# Или сборка с помощью CMake вручную
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

Исполняемые файлы будут расположены в:

- `client-console/fear` - консольный клиент/сервер
- `gui/fear-gui` - графический интерфейс
- `key-exchange/key-exchanger` - утилита обмена ключами
- `audio_call/audio_call` - утилита аудиозвонков
- `updater/updater` - утилита обновления

Windows

Установка зависимостей

1. Установите **Visual Studio 2019/2022** с поддержкой C++
2. Установите **CMake** (https://cmake.org/download/)

3. Установите **Qt 6** (https://www.qt.io/download)
4. Установите **vcpkg** для управления библиотеками:

```
git clone https://github.com/microsoft/vcpkg.git
cd vcpkg
bootstrap-vcpkg.bat
vcpkg integrate install
vcpkg install libsodium:x64-windows portaudio:x64-windows opus:x64-windows
```

Сборка

```
git clone https://github.com/shchuchkin-pkims/fear.git
cd fear

mkdir build
cd build
cmake .. -DCMAKE_TOOLCHAIN_FILE=C:/path/to/vcpkg/scripts/buildsystems/vcpkg.cmake
cmake --build . --config Release
```

macOS

Установка зависимостей

```
# Установка Homebrew (если не установлен)
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Установка зависимостей
brew install git cmake libsodium qt@6 portaudio opus
```

Сборка

```
git clone https://github.com/shchuchkin-pkims/fear.git
cd fear

mkdir build
cd build
cmake .. -DCMAKE_PREFIX_PATH=$(brew --prefix qt@6)
cmake --build . --config Release
```

Описание программ

F.E.A.R. Project состоит из нескольких исполняемых модулей:

1. fear / fear.exe (Консольный клиент/сервер)

Расположение: `client-console/fear` или `bin/fear.exe`

Назначение: Основная консольная утилита для запуска сервера, подключения клиента и генерации ключей

Команды

Генерация ключа комнаты

```
fear genkey
```

Описание: Генерирует криптографически стойкий 256-битный ключ в формате base64 URL-safe

Пример вывода:

```
Room key (base64 urlsafe, save/share securely):
z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I
```

Важно: Сохраните этот ключ в безопасном месте. Он потребуется всем участникам комнаты.

Запуск сервера

```
fear server --port <порт>
```

Аргументы:

- `--port` (обязательно) - TCP порт для прослушивания (1024-65535)

Примеры:

```
# Запуск на порту 7777
fear server --port 7777

# Запуск на порту 8888
fear server --port 8888
```

Вывод при успешном запуске:

```
[server] listening on 0.0.0.0:7777
```

Примечания:

- Сервер не требует ключей шифрования
- Сервер не хранит сообщения (передача в реальном времени)
- Для доступа из интернета необбходима настройка маршрутизатора (проброс портов)

Подключение клиента

```
fear client --host <адрес> --port <порт> --room <комната> --key <ключ> --name
<имя>
```

Аргументы:

- `--host` (обязательно) - IP-адрес или доменное имя сервера
- `--port` (обязательно) - TCP порт сервера
- `--room` (обязательно) - Имя комнаты (1-255 символов)
- `--key` (обязательно) - Ключ шифрования комнаты (base64, 44 символа)
- `--name` (обязательно) - Ваше имя пользователя (1-255 символов)

Примеры:

```
# Подключение к локальному серверу
fear client --host 127.0.0.1 --port 7777 \
    --room testroom \
    --key z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I \
    --name Alice

# Подключение к удаленному серверу
fear client --host example.com --port 7777 \
    --room myroom \
    --key z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I \
    --name Bob
```

Вывод при успешном подключении:

```
[client] connected to 127.0.0.1:7777, Room name: testroom
Type messages and press Enter. Use '/sendfile filename' to send files. Ctrl+C to
exit.
```

Использование:

- Введите сообщение и нажмите Enter для отправки

- Команда `/sendfile путь/к/файлу` - отправка файла
- Ctrl+C для выхода

Важные ограничения:

- Имя пользователя должно быть уникальным в пределах одной комнаты
- Все участники комнаты должны использовать один и тот же ключ
- Неправильный ключ не позволит расшифровать сообщения

---

2. fear-gui / fear-gui.exe (Графический интерфейс)

Расположение: `gui/fear-gui` или `bin/fear-gui.exe`

Назначение: Графическое приложение с удобным интерфейсом для работы с мессенджером

Запуск

Linux/macOS:

```
./fear-gui
```

Windows:

```
fear-gui.exe
```

Или просто двойной клик по исполняемому файлу.

Возможности

- **Простое подключение:** Диалоговое окно с полями для ввода параметров
- **Генерация ключей:** Встроенная функция генерации ключей комнат
- **История чата:** Сохранение истории сообщений в сессии
- **Список участников:** Отображение всех подключенных пользователей
- **Отправка файлов:** Кнопка для выбора и отправки файлов
- **Настройки шрифта:** Изменение размера и начертания текста в чате
- **Аудиозвонки:** Запуск зашифрованных голосовых вызовов
- **Обмен ключами:** Встроенный модуль безопасного обмена ключами

Основные действия

Создание сервера:

1. Меню → Connection → Create server
2. Укажите порт (например, 7777)
3. Нажмите "Create Server"

Подключение к комнате:

1. Меню → Connection → Connect
2. Заполните поля:
    - Host: адрес сервера
    - Port: порт сервера
    - Room name: имя комнаты
    - Room key: ключ шифрования (можно сгенерировать через Keys → Generate keypair)
    - Your name: ваше имя
3. Нажмите "Connect"

Отправка сообщения:

1. Введите текст в нижнее поле
2. Нажмите "Send" или Enter

Отправка файла:

1. Кнопка "Send file" над окном чата
2. Выберите файл
3. Файл автоматически отправится всем участникам комнаты

Аудиозвонок:

1. Меню → Audio call → Start audio call
2. Сгенерируйте ключ или введите существующий
3. Для исходящего вызова: укажите IP и порт собеседника, нажмите "Start Call"
4. Для входящего вызова: нажмите "Start Listening", сообщите свой IP и порт звонящему

---

3. key-exchanger / key-exchanger.exe (Обмен ключами)

Расположение: `key-exchange/key-exchanger` или `bin/key-exchanger.exe`

Назначение: Консольная утилита для безопасного обмена ключами комнаты по протоколу Диффи-Хеллмана на эллиптических кривых (Curve25519)

Запуск

```
key-exchanger
```

Программа интерактивная и предлагает выбор действий.

Главное меню

```
=== F.E.A.R. Key Exchange ===
1. Generate key pair
2. Encrypt message
```

```
3. Decrypt message
4. Exit
Choose option:
```

Сценарий использования

Шаг 1. Пользователь A генерирует ключевую пару

```
Choose option: 1
Your public key (share with partner):
a1b2c3d4e5f6...

Your secret key (keep private!):
x9y8z7w6v5u4...
```

Шаг 2. Пользователь A отправляет свой публичный ключ пользователю B (через любой канал)

Шаг 3. Пользователь B генерирует свою ключевую пару и получает публичный ключ A

```
Choose option: 1
```

Шаг 4. Пользователь A шифрует ключ комнаты

```
Choose option: 2
Enter your secret key:
x9y8z7w6v5u4...

Enter partner's public key:
[публичный ключ B]

Enter message to encrypt:
z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I

Encrypted message:
f4e3d2c1b0a9...
```

Шаг 5. Пользователь A отправляет зашифрованное сообщение пользователю B

Шаг 6. Пользователь B расшифровывает ключ комнаты

```
Choose option: 3
Enter your secret key:
[секретный ключ B]

Enter partner's public key:
```

```
a1b2c3d4e5f6...

Enter encrypted message:
f4e3d2c1b0a9...

Decrypted message:
z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I
```

Важно:

- Секретный ключ НИКОГДА не передается никому
- Публичные ключи можно передавать по открытым каналам
- Даже если злоумышленник перехватит оба публичных ключа и зашифрованное сообщение, он не сможет расшифровать ключ комнаты

---

4. audio_call / audio_call.exe (Аудиозвонки)

Расположение: `audio_call/audio_call` или `bin/audio_call.exe`

Назначение: Утилита для зашифрованных голосовых вызовов P2P (peer-to-peer)

Команды

Генерация ключа для звонка

```
audio_call genkey
```

Вывод:

```
Generated key (hex, 32 bytes):
a1b2c3d4e5f67890123456789012345678901234567890abcdef1234567890abcdef123456
```

Ожидание входящего вызова (режим прослушивания)

```
audio_call listen <локальный_порт> <ключ>
```

Аргументы:

- `<локальный_порт>` - UDP порт для прослушивания (1024-65535)
- `<ключ>` - 256-битный ключ в hex-формате (64 символа)

Пример:

```
audio_call listen 50000
a1b2c3d4e5f67890123456789012345678901234567890abcdef1234567890abcdef123456
```

Вывод:

```
Listening on UDP port 50000
Waiting for incoming call...
```

Исходящий вызов

```
audio_call call <удаленный_IP> <удаленный_порт> <ключ> [локальный_порт]
```

Аргументы:

- `<удаленный_IP>` - IP-адрес собеседника
- `<удаленный_порт>` - UDP порт собеседника
- `<ключ>` - 256-битный ключ в hex-формате (64 символа)
- `[локальный_порт]` - (опционально) локальный UDP порт

Примеры:

```
# Вызов с автоматическим выбором локального порта
audio_call call 192.168.1.100 50000 a1b2c3d4...

# Вызов с указанным локальным портом
audio_call call 192.168.1.100 50000 a1b2c3d4... 50001
```

Технические характеристики:

- Кодек: Opus (высокое качество, низкая задержка)
- Частота дискретизации: 48 кГц
- Битрейт: адаптивный (8-128 кбит/с)
- Шифрование: XChaCha20-Poly1305
- Задержка: < 50 мс (при хорошем соединении)

Примечания:

- Оба участника должны использовать один и тот же ключ
- Требуется прямое соединение или настроенный NAT traversal
- Для работы через интернет может потребоваться проброс портов на роутере

---

5. updater / updater.exe (Обновление)

Расположение: `updater/updater` или `bin/updater.exe`

Назначение: Автоматическое обновление F.E.A.R. до последней версии с GitHub

Запуск

```
updater
```

Программа автоматически:

1. Проверяет текущую версию
2. Загружает последний релиз с GitHub
3. Распаковывает файлы
4. Обновляет исполняемые файлы
5. Предлагает перезапустить приложение

Использование через GUI:

1. Меню → Help → Check for updates
2. Нажмите "Check Version"
3. Если доступно обновление, нажмите "Update"
4. Дождитесь завершения обновления
5. Перезапустите приложение

Примечание: Для корректной работы требуется подключение к интернету.

Быстрый старт

Сценарий 1: Два пользователя в локальной сети

Цель: Alice и Bob хотят безопасно общаться через локальную сеть.

Шаг 1: Alice генерирует ключ комнаты

```
cd bin
./fear genkey
```

Результат: `z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I`

Шаг 2: Alice запускает сервер

```
./fear server --port 7777
```

Шаг 3: Alice узнает свой IP-адрес

```
# Linux/macOS
ip addr show  # или ifconfig

# Windows
ipconfig
```

Предположим IP: `192.168.1.10`

Шаг 4: Alice сообщает Bob:

- IP-адрес: `192.168.1.10`
- Порт: `7777`
- Ключ комнаты: `z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I`

Шаг 5: Alice подключается к своему серверу

```
./fear client --host 192.168.1.10 --port 7777 \
    --room myroom \
    --key z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I \
    --name Alice
```

Шаг 6: Bob подключается к серверу Alice

```
./fear client --host 192.168.1.10 --port 7777 \
    --room myroom \
    --key z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I \
    --name Bob
```

Шаг 7: Alice и Bob могут общаться!

Alice печатает: `Привет, Bob!` [Enter] Bob видит: `[12:34:56] Alice: Привет, Bob!`

Сценарий 2: Безопасный обмен ключом через интернет

Цель: Alice и Bob хотят обменяться ключом комнаты безопасно, не доверяя каналу связи.

Шаг 1: Alice генерирует ключ комнаты (но пока не отправляет его)

```
./fear genkey
```

Результат: `z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I`

Шаг 2: Alice генерирует ключевую пару для обмена

```
./key-exchanger
```

```
Choose option: 1

Your public key: a1b2c3d4e5f6...
Your secret key: x9y8z7w6v5u4...
```

Alice сохраняет секретный ключ у себя!

Шаг 3: Alice отправляет Bob свой публичный ключ (например, через email, Telegram, и т.д.)

Шаг 4: Bob генерирует свою ключевую пару

```
./key-exchanger
Choose option: 1

Your public key: m1n2o3p4q5r6...
Your secret key: s7t8u9v0w1x2...
```

Шаг 5: Bob отправляет Alice свой публичный ключ

Шаг 6: Alice шифрует ключ комнаты

```
./key-exchanger
Choose option: 2

Enter your secret key: x9y8z7w6v5u4...
Enter partner's public key: m1n2o3p4q5r6...
Enter message to encrypt: z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I

Encrypted message: f4e3d2c1b0a9...
```

Шаг 7: Alice отправляет зашифрованное сообщение Bob

Шаг 8: Bob расшифровывает ключ комнаты

```
./key-exchanger
Choose option: 3

Enter your secret key: s7t8u9v0w1x2...
Enter partner's public key: a1b2c3d4e5f6...
Enter encrypted message: f4e3d2c1b0a9...

Decrypted message: z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I
```

Шаг 9: Теперь и Alice, и Bob знают ключ комнаты и могут безопасно общаться!

Важно: Даже если злоумышленник перехватил все сообщения (публичные ключи и зашифрованный текст), он НЕ СМОЖЕТ восстановить ключ комнаты без секретных ключей.

Подробное руководство

Работа с комнатами

Создание комнаты

Комната создается автоматически при первом подключении клиента. Просто выберите уникальное имя комнаты и сгенерируйте ключ:

```
# Генерация ключа
./fear genkey

# Подключение создает комнату
./fear client --host <сервер> --port <порт> \
    --room mynewroom \
    --key <ваш_ключ> \
    --name <ваше_имя>
```

Подключение к существующей комнате

Для подключения к существующей комнате вам нужны:

1. Адрес и порт сервера
2. Имя комнаты
3. Правильный ключ шифрования

```
./fear client --host server.example.com --port 7777 \
    --room existingroom \
    --key z6aK3_k9I7rmpy6Sn-84QZ9Yc0p3T7VhzReWCKE0x4I \
    --name NewUser
```

Важно: Если вы используете неправильный ключ, вы не сможете расшифровать сообщения других участников (будет ошибка дешифрования).

Множественные комнаты

Один сервер может обслуживать неограниченное количество комнат одновременно. Каждая комната полностью изолирована:

```
# Пользователь в комнате "work"
./fear client --host server.com --port 7777 \
    --room work --key <ключ1> --name Alice

# Другой пользователь в комнате "friends"
```

```
./fear client --host server.com --port 7777 \
    --room friends --key <ключ2> --name Bob
```

Alice и Bob не увидят сообщения друг друга, даже находясь на одном сервере.

Управление пользователями

Уникальность имен

Имя пользователя должно быть уникальным в пределах одной комнаты. Если два пользователя с одинаковым именем попытаются подключиться к одной комнате, второй получит отказ в подключении:

```
[server] client rejected: name 'Alice' already exists in room 'testroom'
```

Решение: Выберите другое имя или дождитесь, пока первый пользователь отключится.

Список участников

В GUI-версии список участников отображается в левой панели "Contacts". Список автоматически обновляется при подключении/отключении пользователей.

В консольной версии используйте команду:

```
[USERS] Room participants (3): Alice, Bob, Charlie
```

Эта информация появляется автоматически при подключении нового пользователя.

Безопасный обмен ключами

Зачем нужен обмен ключами?

Главная проблема любого шифрования — как безопасно передать ключ собеседнику? Если вы отправите ключ комнаты по незащищенному каналу (email, SMS, социальные сети), злоумышленник может его перехватить.

Решение: Протокол Диффи-Хеллмана на эллиптических кривых (ECDH) позволяет двум сторонам выработать общий секрет через незащищенный канал без передачи самого секрета.

Как работает ECDH (упрощенно)

1. Alice и Bob генерируют пары ключей: (публичный, секретный)
2. Alice и Bob обмениваются **только публичными ключами**
3. Alice комбинирует свой секретный ключ с публичным ключом Bob → получает общий секрет
4. Bob комбинирует свой секретный ключ с публичным ключом Alice → получает тот же общий секрет
5. Злоумышленник, перехвативший оба публичных ключа, **не может** вычислить общий секрет (вычислительно невозможно)

Практическое использование

Через консольную утилиту

См. Сценарий 2 выше.

Через GUI

1. Откройте GUI: `./fear-gui`
2. Меню → Keys → Key exchange
3. Нажмите "Generate Key Pair"
4. Скопируйте ваш Public key и отправьте собеседнику
5. Получите Public key собеседника
6. **Для отправки ключа комнаты:**
   - Введите Message to send: (ваш ключ комнаты)
   - Введите Friend's public key
   - Нажмите "Encrypt"
   - Отправьте Encrypted message собеседнику
7. **Для получения ключа комнаты:**
   - Введите Encrypted message от собеседника
   - Введите Sender's public key
   - Нажмите "Decrypt"
   - Используйте Decrypted message как ключ комнаты

Лучшие практики

✅ **Рекомендуется:**

- Проверяйте публичные ключи собеседника по дополнительному каналу связи (телефонный звонок, личная встреча)
- Используйте key-exchanger для каждой новой комнаты
- Регулярно меняйте ключи комнат (например, раз в неделю)

❌ **Избегайте:**

- Отправки ключей комнаты в открытом виде
- Переиспользования одного ключа для разных групп людей
- Хранения ключей в незашифрованных файлах на диске

Аудиозвонки

Требования для аудиозвонков

- Микрофон и динамики/наушники
- Прямое сетевое соединение между участниками (или настроенный NAT traversal)
- Низкая задержка сети (< 100 мс для комфортного разговора)

Настройка звонка

Способ 1: Через GUI

1. Откройте GUI обоих участников
2. Меню → Audio call → Start audio call
3. Один участник (A) нажимает "Generate" для создания ключа
4. Участник A сообщает ключ участнику B (можно через key-exchanger для безопасности)
5. Участник B вводит полученный ключ
6. **Участник B (принимающий):**
    - Вводит свой Local Port (например, 50000)
    - Нажимает "Start Listening"
    - Сообщает свой IP и порт участнику A
7. **Участник A (звонящий):**
    - Вводит Remote IP участника B
    - Вводит Remote Port участника B (50000)
    - Вводит свой Local Port (например, 50001)
    - Нажимает "Start Call"
8. Звонок установлен!

Способ 2: Через консоль

Участник B (принимающий):

```
# Генерируем ключ
./audio_call genkey
# Результат: a1b2c3d4e5f6...

# Запускаем прослушивание
./audio_call listen 50000 a1b2c3d4e5f6...
```

Участник B сообщает участнику A:

- IP: 192.168.1.100
- Порт: 50000
- Ключ: a1b2c3d4e5f6...

Участник A (звонящий):

```
./audio_call call 192.168.1.100 50000 a1b2c3d4e5f6... 50001
```

Устранение проблем со звуком

Проблема: Нет звука, но соединение установлено

Решение:

1. Проверьте микрофон и динамики в системных настройках
2. Убедитесь, что микрофон не заглушен
3. Проверьте, что выбраны правильные аудиоустройства
4. В Linux проверьте настройки ALSA/PulseAudio

Проблема: Высокая задержка или прерывания

Решение:

1. Проверьте задержку сети: `ping <IP_собеседника>`
2. Используйте проводное подключение вместо Wi-Fi
3. Закройте программы, потребляющие интернет (торренты, стримы)
4. Проверьте настройки QoS на роутере

Проблема: Не удается установить соединение

Решение:

1. Убедитесь, что оба используют один и тот же ключ
2. Проверьте, что порты открыты в файерволе
3. Для подключения через интернет настройте проброс портов (Port Forwarding) на роутере
4. Рассмотрите использование VPN для прямого соединения

Передача файлов

Отправка файла

Через GUI

1. Подключитесь к комнате
2. Нажмите кнопку "Send file" над окном чата
3. Выберите файл в диалоговом окне
4. Файл автоматически отправится всем участникам комнаты

Через консоль

```
# Во время работы клиента введите команду:
/sendfile /путь/к/файлу.txt

# Пример (Linux/macOS):
/sendfile /home/user/Documents/report.pdf

# Пример (Windows):
/sendfile C:\Users\User\Documents\report.pdf
```

Получение файла

Файлы автоматически сохраняются в папку `Downloads` в директории программы:

```
fear/
├── bin/
│   ├── fear.exe
│   └── Downloads/          ← Полученные файлы здесь
│       ├── report.pdf
│       └── photo.jpg
```

Проверка целостности

F.E.A.R. автоматически проверяет целостность файлов с помощью CRC32:

При отправке:

```
Sending file: report.pdf (1024000 bytes)
Progress: 1024000/1024000 bytes (100.0%)
File sent successfully: report.pdf
```

При получении:

```
Receiving file: report.pdf (1024000 bytes)
Progress: 1024000/1024000 bytes (100.0%)
File received successfully: report.pdf
```

Если файл поврежден:

```
File corrupted: report.pdf (CRC mismatch)
```

В этом случае попросите отправителя переотправить файл.

Ограничения

- Максимальный размер файла: ограничен только доступной памятью
- Размер чанка: 8192 байт (оптимизирован для сетевой передачи)
- Файлы передаются **зашифрованными** с использованием того же ключа комнаты

Устранение неполадок

Проблемы подключения

Ошибка: "Connection refused"

Причина: Сервер не запущен или указан неправильный адрес/порт

Решение:

1. Убедитесь, что сервер запущен: `./fear server --port 7777`
2. Проверьте правильность IP-адреса и порта
3. Проверьте файервол на сервере
4. Попробуйте подключиться с самого сервера: `--host 127.0.0.1`

Ошибка: "Failed to register with server"

Причина: Проблема при отправке первого сообщения серверу

Решение:

1. Проверьте сетевое соединение
2. Убедитесь, что ключ корректный (44 символа base64)
3. Перезапустите клиент

Ошибка: "Name already exists in room"

Причина: Пользователь с таким именем уже в комнате

Решение:

1. Выберите другое имя пользователя
2. Или попросите другого пользователя отключиться

Проблемы с шифрованием

Сообщения не расшифровываются

Признаки:

- Сообщения других пользователей не появляются
- Или отображаются как мусор/ошибка

Причина: Неправильный ключ комнаты

Решение:

1. Убедитесь, что все участники используют ТОЧНО ОДИН И ТОТ ЖЕ ключ
2. Проверьте, что ключ не был поврежден при копировании (должен быть ровно 44 символа)
3. Пересоздайте ключ и раздайте его снова

Проблемы с производительностью

Высокая нагрузка на процессор

Причина: Шифрование/дешифрование — вычислительно интенсивные операции

Решение:

1. Используйте более современный процессор с AES-NI инструкциями
2. Ограничьте количество одновременно открытых комнат
3. Используйте GUI вместо консоли (меньше перерисовок)

Задержки при передаче файлов

Причина: Ограничения сети или шифрование больших объемов данных

Решение:

1. Используйте проводное подключение
2. Разделите большие файлы на части
3. Сжимайте файлы перед отправкой (ZIP/7z)

Проблемы сборки

Ошибка: "libsodium not found"

Решение (Ubuntu/Debian):

```
sudo apt install libsodium-dev
```

Решение (Fedora):

```
sudo dnf install libsodium-devel
```

Решение (Windows с vcpkg):

```
vcpkg install libsodium:x64-windows
```

Ошибка: "Qt6 not found"

Решение (Linux):

```
# Ubuntu/Debian
sudo apt install qt6-base-dev
```

```
# Fedora
sudo dnf install qt6-qtbase-devel
```

Решение (Windows): Скачайте и установите Qt с официального сайта: https://www.qt.io/download

Решение (macOS):

```
brew install qt@6
cmake .. -DCMAKE_PREFIX_PATH=$(brew --prefix qt@6)
```

Часто задаваемые вопросы

Общие вопросы

Q: Является ли F.E.A.R. полностью анонимным?

A: F.E.A.R. обеспечивает конфиденциальность **содержимого** сообщений, но не скрывает факт коммуникации. Сервер и сетевые наблюдатели могут видеть:

- IP-адреса участников
- Время подключения/отключения
- Размер передаваемых данных

Для полной анонимности используйте F.E.A.R. через VPN или Tor.

Q: Можно ли восстановить удаленные сообщения?

A: Нет. F.E.A.R. не хранит историю сообщений на сервере. Сообщения существуют только в памяти клиентов во время сессии.

Q: Нужно ли доверять серверу?

A: Сервер НЕ МОЖЕТ прочитать ваши сообщения благодаря E2EE. Однако сервер может:

- Видеть метаданные (кто, когда, с кем общается)
- Сохранять зашифрованные сообщения (но не расшифровать их)
- Блокировать пользователей

Для максимальной безопасности используйте **собственный сервер**.

Q: Можно ли использовать F.E.A.R. для групповых чатов?

A: Да! Неограниченное количество пользователей может подключиться к одной комнате. Все сообщения зашифрованы одним ключом комнаты.

Технические вопросы

Q: Какой алгоритм используется для шифрования?

A: AES-256-GCM (Advanced Encryption Standard, 256-битный ключ, режим Galois/Counter Mode). Это один из самых надежных и быстрых алгоритмов симметричного шифрования.

Q: Как часто нужно менять ключи?

A: Рекомендуется менять ключ комнаты:

- При добавлении нового участника (для сохранения PFS - Perfect Forward Secrecy)
- При удалении участника
- Минимум раз в месяц для долгосрочных комнат

Q: Поддерживается ли видео?

A: В текущей версии — нет. Видеозвонки запланированы на будущие релизы.

Q: Можно ли запустить несколько серверов?

A: Да, но они будут независимыми. F.E.A.R. не поддерживает федерацию серверов (пока).

Q: Работает ли F.E.A.R. на мобильных устройствах?

A: Мобильное приложение для Android/iOS находится в разработке. Следите за обновлениями на GitHub.

Безопасность

Q: Может ли правительство/спецслужбы взломать F.E.A.R.?

A: При правильном использовании AES-256 невозможно взломать методом перебора (потребуются миллиарды лет). Однако возможны атаки:

- Взлом конечных устройств (вирусы, кейлоггеры)
- Компрометация ключей
- Атаки "человек посередине" при обмене ключами

Защита: Используйте антивирус, храните ключи в безопасности, проверяйте публичные ключи собеседников.

Q: Безопасно ли использовать F.E.A.R. на работе?

A: С технической точки зрения — да (E2EE). С юридической — зависит от политики компании. F.E.A.R. не предназначен для обхода корпоративных политик.

Q: Что делать, если ключ комнаты был скомпрометирован?

A:

1. Немедленно сгенерируйте новый ключ: `./fear genkey`
2. Сообщите новый ключ всем доверенным участникам (используя key-exchanger)
3. Создайте новую комнату с новым именем
4. НЕ используйте старую комнату

Q: Логирует ли F.E.A.R. сообщения на диск?

A: Нет, по умолчанию логирование отключено. Сообщения существуют только в памяти во время работы программы.

Лицензия

F.E.A.R. Project распространяется под лицензией **MIT**.

Это означает:

- ✅ Свободное использование в коммерческих и некоммерческих целях
- ✅ Модификация исходного кода
- ✅ Распространение копий
- ✅ Использование в закрытых проектах

Условие: Сохранение копирайта и лицензии в исходниках.

Контакты и поддержка

GitHub: https://github.com/shchuchkin-pkims/fear **Email:** shchuchkin-pkims@yandex.ru **Issues:** https://github.com/shchuchkin-pkims/fear/issues

Как сообщить об ошибке

1. Откройте issue на GitHub
2. Опишите проблему:
    - Версия F.E.A.R.
    - Операционная система
    - Шаги для воспроизведения
    - Ожидаемое поведение
    - Фактическое поведение
3. Приложите логи (если есть)

Как предложить улучшение

1. Откройте issue с тегом "enhancement"
2. Опишите предлагаемую функцию
3. Объясните, как она улучшит F.E.A.R.

Благодарности

F.E.A.R. использует следующие открытые библиотеки:

- **libsodium** - Современная криптографическая библиотека
- **PortAudio** - Кроссплатформенная аудиобиблиотека
- **Opus** - Высококачественный аудиокодек
- **CMake** - Система сборки

Спасибо всем контрибьюторам и пользователям за поддержку проекта!

Оставайтесь анонимными. Оставайтесь в безопасности. Щучкин Е. Ю. F.E.A.R. Project © 2025