Захищений клієнт-серверний застосунок OffGrid із E2E-шифруванням і контрольованим файлообміном

УДК 004.056.55

Катерина Холодова[1]

Національний університет «Одеська політехніка», [1]9480541@stud.op.edu.ua

Системи миттєвого обміну повідомленнями є одним із ключових каналів передавання персональних і службових даних. Навіть за наявності наскрізного шифрування зберігаються ризики витоку метаданих, небезпечного обміну вкладеннями, підміни ключів і компрометації кінцевих пристроїв [1]. Тому актуальним є проєктування месенджера, у якому сервер не вважається повністю довіреним, а критичні криптографічні операції виконуються на клієнті.

Метою роботи є розроблення клієнт-серверного застосунку OffGrid для захищеного обміну повідомленнями та файлами в умовах недовіреної серверної інфраструктури. Для цього сформульовано модель загроз, спроєктовано архітектуру «тонкого ретранслятора», реалізовано механізми керування ключами і проведено експериментальну перевірку. Науково-практична новизна полягає у поєднанні E2E-шифрування, двоконтурного передавання вкладень, локального контролю довіри до пристроїв і очищення чутливих даних після secure-сесій.

Архітектура OffGrid складається з клієнтського застосунку, TCP-сервера, HTTP-файлового контуру та підсистем зберігання. TCP-канал використовується для автентифікації, маршрутизації подій і видачі короткоживучих токенів доступу, а передавання великих вкладень винесено в окремий HTTP-контур. Сервер зберігає лише шифротекст і мінімально необхідні метадані, тоді як приватні ключі, стани сесій і операції шифрування залишаються на клієнті.

Керування ключами організовано на рівні пристроїв. Кожен клієнт має стабільний device_id і власні пари ключів: Ed25519 використовується для цифрового підпису, а X25519 - для узгодження спільного секрету. Автентичність ключів співрозмовника контролюється через TOFU з локальним закріпленням: зміна ключового матеріалу для відомого пристрою блокує взаємодію до явного підтвердження користувачем.

Для отримання ключового матеріалу з узгодженого секрету використано HKDF, що забезпечує доменне розділення ключів між різними протокольними задачами [2]:

$$K = HKDF(z; salt, info, 32)\,, \tag{1}$$

де K – майстер-ключ; z – спільний секрет, отриманий через X25519;

salt – сіль (за потреби); info – параметри контексту й доменного розділення; 32 – довжина ключа в байтах.

Для шифрування E2E-повідомлень застосовано ChaCha20-Poly1305 як AEAD-примітив:

$$\begin{aligned}(C, T) &= AEAD\_Enc(K, N, AAD, P),\\ P &= AEAD\_Dec(K, N, AAD, C, T),\end{aligned} \tag{2}$$

де C – шифротекст; T – тег автентичності; K – симетричний ключ; N – одноразове випадкове значення (нонс); AAD – додаткові автентифіковані дані; P – відкритий текст.

Додаткові автентифіковані дані прив'язують шифротекст до контексту сесії, епохи, напряму передавання та ідентифікатора повідомлення; перенесення шифротексту в інший контекст призводить до помилки автентифікації [3].

У файловій підсистемі використано модель «ticket → bearer-token → HTTP». Клієнт через TCP-канал запитує дозвіл, сервер виконує ACL-перевірку і видає підписаний bearer-token з обмеженим строком дії. Далі клієнт завантажує або скачує файл через HTTP, передаючи токен у заголовку Authorization. Файловий сервер перевіряє підпис, часові межі й тип операції, але не має доступу до відкритого вмісту, оскільки файл шифрується на клієнті до передавання.

Захист облікових і користувацьких секретів реалізовано через bcrypt для паролів та PBKDF2-HMAC-SHA256 для секретів відновлення і локальних механізмів доступу. Приватні ключі й токени інтеграцій зберігаються через системні сховища, зокрема keyring або DPAPI, що зменшує ризик їх потрапляння у відкриті конфігураційні файли.

Програмну реалізацію виконано на Python із розділенням відповідальності між сервером, мережевим шаром, графічним інтерфейсом і криптографічними модулями. Така декомпозиція дозволила окремо перевірити маршрутизацію, роботу сесій, доступ до вкладень і локальні процедури захисту секретів.

Експериментальна верифікація підтвердила ключові інваріанти безпеки. Тест зміни identity-ключа показав, що TOFU/pinning виявляє зміну ключового матеріалу для відомого device_id і блокує взаємодію до відновлення довіри. Негативне тестування авторизації засвідчило, що сторонній користувач не може виконувати операції з приватними групами та пов'язаними вкладеннями, а сервер не розкриває факт існування приватного ресурсу. Тест очищення RAM у secure-режимі підтвердив очищення внутрішнього secure-сховища і занулення змінюваних буферів.

Отримані результати показують, що запропонована архітектура забезпечує конфіденційність і цілісність E2E-вмісту, контроль автентичності ключів і розмежування доступу до вкладень без надання серверу відкритих даних або приватного ключового матеріалу. Рішення може бути використане як основа для систем обміну чутливою інформацією, де важливими є мінімізація довіри до інфраструктури, контроль метаданих і захист кінцевої точки.

1. Alatawi M., Saxena N. Sok: An analysis of end-to-end encryption and authentication ceremonies in secure messaging systems. Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks. 2023.
2. Serrano R. et al. ChaCha20–Poly1305 authenticated encryption with additional data for transport layer security 1.3. Cryptography. 2022. Vol. 6, No. 2. P. 30.
3. Onik A. R. et al. A systematic literature review of secure instant messaging applications from a digital forensics perspective. ACM Computing Surveys. 2025. Vol. 57, No. 9. P. 1–36.