Engineer

Шифрування

перетворення даних так, щоб їх не прочитали без ключа

Що це таке

Шифрування перетворює зрозумілі дані, відкритий текст, на шифротекст: їх пропускають через загальновідомий алгоритм разом із секретним значенням — ключем. Розшифрування з правильним ключем повертає відкритий текст байт у байт. Алгоритм відкритий навмисно: AES, яким користується більшість систем, у 2000 році переміг у відкритому конкурсі NIST і опублікований як безплатний стандарт. Таємним залишається лише ключ.

Шифри поділяють на дві родини. Симетричний шифр використовує той самий ключ в обидва боки. AES, описаний у FIPS 197, шифрує дані блоками по 128 біт ключем завдовжки 128, 192 або 256 біт. ChaCha20 з RFC 8439 — потоковий шифр, приблизно втричі швидший за AES на обладнанні без спеціальних інструкцій AES. Симетричне шифрування досить швидке для цілого диска, але обидві сторони мають знати той самий ключ, і найважче — безпечно доправити його на інший бік.

Криптографія з відкритим ключем розв’язує це парою ключів: один публікують, другий тримають у таємниці. Дві сторони можуть кожна поєднати свій закритий ключ із відкритим ключем іншої і дійти до того самого спільного секрету, жодного разу його не пересилаючи. Такі обчислення повільні, тож справжні системи поєднують обидва підходи: обмін відкритими ключами узгоджує свіжий симетричний ключ, а симетричний шифр несе самі дані. Так працює TLS, протокол за кожною адресою `https://` .

Найважливіша сучасна ідея — автентифікація. Голий шифр приховує вміст, але лишає його відкритим для змін: у режимі лічильника перевернутий біт шифротексту перевертає той самий біт відкритого тексту, і отримувач розшифровує змінене повідомлення як справжнє. Автентифіковане

шифрування з приєднаними даними, AEAD, додає тег, який отримувач перевіряє, перш ніж довіритися бодай одному байту. AES-GCM і ChaCha20-Poly1305 — звична пара, а TLS 1.3, опублікований як RFC 8446 у 2018 році, залишив тільки шифри AEAD і забезпечує пряму секретність (forward secrecy) для кожного обміну відкритими ключами, тож ключ сервера, викрадений пізніше, не відкриває жодного записаного сеансу.

Застосуйте поняття до рішення

Застосуйте «Що таке шифрування?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Дані рідко лишаються на обладнанні, яке контролює їхній власник: вони йдуть чужими мережами й лежать у чужих дата-центрах. Шифрування дає їм безпечно пройти крізь ці місця: викрадений диск, прослуханий кабель чи сховище, що опинилося у відкритому доступі, дають лише шифротекст, а шифротекст без ключа — це шум.

Це переносить проблему, а не усуває її: щойно дані зашифровано, їхня безпека — це безпека ключа. Рекомендація NIST щодо керування ключами, SP 800-57, порівнює ключі з кодом до сейфа: зловмисник, який знає код, проходить повз найміцніший сейф, а погане керування ключами легко підриває сильний алгоритм. Тож більша частина справжньої роботи — це робота з ключами: зберігати їх окремо від даних, обмежувати, скільки захищає кожен, замінювати за графіком і стерегти резервну копію так само пильно, як оригінал.

Законодавці проводять ту саму межу. Загальний регламент про захист даних (GDPR) у статті 32 називає псевдонімізацію та шифрування персональних даних серед заходів безпеки. Стаття 34 звільняє контролера від обов’язку повідомляти кожного суб’єкта даних про порушення, якщо дані, що витекли, були незрозумілими для будь-кого без дозволу на доступ, і шифрування — це приклад, який вона наводить.

Шифрування під час передавання, здебільшого TLS, захищає дані в мережі, а шифрування під час зберігання — на носії. Саме використання лишається поза обома: сервер, що розшифровує запис, щоб із ним працювати, тримає водночас відкритий текст і ключ, тож хто контролює сервер, той читає те саме, що й він.

Як це виглядає на практиці

Під час передавання щоденна робота — тримати сертифікати дійсними. Коли сто застосунків на Docker потребували сертифікатів SSL/TLS, робочий процес на основі ACME створював кожен сертифікат, поновлював його до завершення терміну дії й розгортав на хостах Ubuntu з Apache та Nginx. Метою було виключити людину з процесу, бо саме людина — та ланка, яка забуває, а прострочений сертифікат виводить застосунок з ладу так само певно, як зламаний шифр.

Під час зберігання корисне питання — у кого ключ. Реалізацію зашифрованих резервних копій поза хостом компанії спроєктовано, щоб розширити можливості відновлення за межі знімка провайдера. Вона використовує restic, який шифрує збережені дані алгоритмом AES-256 у режимі лічильника й автентифікує їх через Poly1305-AES, а її скрипти підтримують надсилання знімків через SFTP до сховища іншого провайдера. У цій схемі провайдер отримує шифротекст, а компанія зберігає парольну фразу в захищеному файлі. Це опис реалізованих засобів захисту, а не їх фактичного введення в дію: у записі від 2026-10-27 зазначено, що таймери були вимкнені за відсутності налаштованого репозиторію; жодного завершеного реального відновлення не зафіксовано. Шифрування є властивістю запропонованого шляху зберігання; щоб довести можливість відновлення резервної копії, потрібно виконати окремий тест відновлення.

Навіть шлях, яким парольна фраза потрапляє в програму, мав значення. Її читають із файлу, бо диспетчер процесів обробляє екранування у значеннях середовища, і парольна фраза зі зворотною похилою рискою мовчки відрізнялася б від тієї, що створила репозиторій. Реалізований щомісячний тест призначений для відновлення з репозиторію файлу з відомим вмістом і порівняння його з очікуваним результатом. Оскільки тест потребує парольної фрази, успішний запуск також підтвердив би, що ключ і далі відкриває дані.

Коли платформа ось-ось мала отримати другий хост, те саме міркування визначило, як кожен експлуатаційний секрет розділили за хостами. Рядок репозиторію резервних копій визначає розташування. Окремі репозиторії restic можуть мати різні головні ключі, навіть якщо використовують однакову парольну фразу. Той, хто отримає цю фразу й зможе прочитати зашифровані дані та файли ключів обох репозиторіїв, зможе відкрити обидва, тож спільна фраза пов'язує ризики їх компрометації. Кожен секрет став відображенням із ключем за хостом, а аргумент записано поруч із налаштуванням. SP 800-57 формулює загальне правило: обмежуючи, що захищає один ключ, обмежуєш шкоду, коли він витече.

Де виникають проблеми

Ключ існує в одному місці. Сильний шифр поводиться з власником, який забув пароль, так само як зі зловмисником. Документація restic каже прямо: втрата пароля означає, що дані втрачено безповоротно. Кожне зашифроване сховище потребує другої копії ключа, яку тримають там, куди не дістане те, що знищить першу.

Ключ лежить поруч із даними. База даних, зашифрована ключем із конфігурації того самого сервера, або резервна копія, чия парольна фраза лежить у тому самому сховищі, зупиняє лише злодія, що забрав половину файлів. Шифрування диска — тихіший варіант: воно захищає накопичувач, вийнятий із вимкненої машини, а сервер, що працює, уже розблокував свій диск, і зловмисник на ньому читає відкритий текст.

Таємність плутають із цілісністю. Система на голому режимі шифру приховує вміст, але все одно дає зловмисникові його змінити. Використовуйте конструкцію AEAD і відкидайте кожне повідомлення з тегом, що не зійшовся, перш ніж діяти на основі бодай одного його байта.

Nonce повторюється. AES-GCM і ChaCha20-Poly1305 приймають nonce — значення, яке має бути унікальним для кожного повідомлення під одним ключем. RFC 8439 називає це найважливішим міркуванням безпеки: два повідомлення з однаковим nonce мають спільний потік ключа й одноразовий ключ автентифікації, а це розкриває XOR двох відкритих текстів. Лічильник — безпечний спосіб отримувати nonce, якщо ключем володіє один лічильник: два процеси, що мають спільний ключ і кожен рахує від нуля, дають саме те повторення, від якого застерігає RFC.

Пароль стає ключем. Люди обирають паролі, а шифрам потрібні випадкові ключі фіксованої довжини. Парольна фраза спершу проходить функцію виведення ключа, як у restic зі scrypt, який RFC 7914 робить вибагливим до пам'яті, щоб масовий перебір на спеціалізованому обладнанні лишався дорогим. Збережені паролі для входу — окремий випадок: їм місце в такій функції у вигляді хешів, бо сервер, що вміє їх розшифрувати, може їх і злити.

Метадані лишаються видимими. Шифрування приховує вміст, а розмір, час і адреси на обох кінцях лишаються на виду. TLS 1.3 дозволяє відправникові додавати до записів заповнення (padding), щоб приховати їхній розмір, але воно необов'язкове, тож довжина часто каже більше, ніж розраховує власник.

Що почитати далі

- NIST Glossary: encryption — визначення терміна з публікацій NIST, кожне з джерелом.
- FIPS 197: Advanced Encryption Standard (AES) — специфікація AES із розміром блока і трьома довжинами ключа.
- NIST SP 800-57 Part 1 Rev. 5: Recommendation for Key Management — як ключі створюють, зберігають, використовують і виводять з обігу та що означає компрометація одного з них.
- RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 — протокол за `https://` із захистом записів лише через AEAD і прямою секретністю обміну ключами.
- RFC 8439: ChaCha20 and Poly1305 for IETF Protocols — потоковий шифр, автентифікатор і їхнє поєднання в AEAD із правилами для nonce.
- restic: References — як інструмент резервного копіювання виводить ключі з пароля, а потім шифрує й автентифікує свої дані.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке шифрування?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/encryption/