Engineer

暗号化

鍵なしではデータを読み取れないようにすること

どのようなものか

暗号化は、NISTの一部の文書ではenciphering とも呼ばれます。読み取れるデータである平文に、公開されたアルゴリズムと、鍵と呼ぶ秘密の値を適用し、暗号文へ変換します。正しい鍵で復号すると、元の平文が一バイトも違わず戻ります。アルゴリズムは、意図して公開します。多くのシステムが使う暗号方式、AESは、2000年にNISTの公開コンペティションで選ばれ、無償で利用できる標準として公開されています。秘密なのは、鍵だけです。

暗号方式には、二つの系統があります。共通鍵暗号は、暗号化と復号に同じ鍵を使います。FIPS 197で規定するAESは、128、192、256ビットのいずれかの鍵を使い、128ビットのブロックごとにデータを暗号化します。RFC 8439のChaCha20は、AES命令のないハードウェアではAESの約三倍の速さで動く、ストリーム暗号です。共通鍵暗号は、ディスク全体にも使えるほど高速ですが、両端に同じ鍵が必要です。その鍵を安全に届けるのが、難しい部分です。

公開鍵暗号は、公開する鍵と、秘密に保つ鍵の組で、この問題に答えます。双方が、それぞれ自分の秘密鍵と相手の公開鍵を組み合わせれば、共通の秘密の値を、一度も送信せずに得られます。公開鍵の演算は遅いため、実際のシステムは、両方を使います。公開鍵による鍵交換で、新しい共通鍵を取り決め、データの通信には共通鍵暗号を使います。すべての `https://` アドレスを支えるプロトコル、TLSも、この仕組みです。

現代の暗号で、最も重要なのは認証です。暗号化だけでは、中身は隠せても、変更を防げません。カウンターモードでは、暗号文の一ビットを反転すると、平文の同じビットも反転します。受信者は、改変されたメッセージを、本物と同じように復号します。関連データ付き認証暗号、AEADは、受信者が一バイトでも信頼する前に確かめる、タグを加えます。よく使うのは、AES-GCMとChaCha20-

Poly1305です。2018年にRFC 8446として公開されたTLS 1.3は、AEAD方式だけを残し、すべての公開鍵による鍵交換に前方秘匿性を持たせました。後からサーバーの鍵を盗まれても、記録済みのセッションは開けません。

概念を判断に生かす

「暗号化とは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

データが、所有者の管理するハードウェアだけにとどまることは、めったにありません。他者のネットワークを通り、他者のデータセンターに置かれ、ノートパソコンで建物の外へ運ばれます。そうした場所を安全に通せるのが、暗号化です。ディスクを盗まれても、ケーブルを盗聴されても、バケットの内容が流出しても、得られるのは暗号文です。鍵のない暗号文は、雑音にすぎません。

問題は、消えるのではなく移ります。データを暗号化すると、その安全性は鍵の安全性にかかります。NISTの鍵管理に関する勧告、SP 800-57は、鍵を金庫の暗証番号に例えます。暗証番号を知る攻撃者は、最も頑丈な金庫でも開けられます。鍵の管理が悪ければ、強いアルゴリズムを使っていても、簡単に保護を破られます。そのため、実務の大部分は鍵の管理です。データと別に保管し、一つの鍵が守る範囲を制限し、予定を決めて交換し、復旧用の複製も元の鍵と同じ慎重さで守ります。

規制も、同じ線を引きます。GDPRの第32条は、個人データの仮名化と暗号化を、安全管理措置の例に挙げます。第34条は、流出したデータが、閲覧を許可されていない人には理解できない場合、管理者が、影響を受けた各人への漏えい通知を省略できるとします。その例が、暗号化です。

暗号化を適用する場面は、二つあります。通信中の暗号化は、主にTLSで、回線を流れるデータを守ります。保存時の暗号化は、保存領域のデータを守ります。データを使用している間は、どちらの保護範囲にも含まれません。処理のためにレコードを復号するサーバーは、平文と鍵を同時に持ちます。そのサーバーを支配した人は、サーバーと同じものを読めます。

実際の活用例

通信中の保護で、日常の仕事になるのは、証明書を有効に保つことです。百のDockerアプリにSSL/TLS証明書が必要だったとき、ACMEのワークフローで、各証明書を作成し、期限前に更新し、ApacheやNginxを動かすUbuntuホストへ配備しました。狙いは、人を手順から外すことでした。忘れるのは人であり、期限切れの証明書は、破綻した暗号方式と同じように、確実にアプリを止めるためです。

保存時の保護では、誰が鍵を持つかが、役立つ問いです。自社のホスト外への暗号化バックアップの実装は、事業者のスナップショットだけに頼らず復旧できるよう設計しました。resticを使い、保存するものをカウンターモードのAES-256で暗号化し、Poly1305-AESで認証します。スクリプトは、別事業者の保存領域へ、SFTPでスナップショットを送れます。この設計では、事業者が受け取るのは暗号文で、会社は、保護したファイルにパスフレーズを保持します。これは、実装した保護措置の説明であり、運用開始を示すものではありません。2026-10-27付の記録では、リポジトリが設定されていないため、タイマーは無効でした。実際に復元を完了した記録もありません。暗号化は、提案した保存経路の性質です。バックアップから復旧できることを示すには、別の復元テストを実行する必要があります。

パスフレーズが通る経路まで、重要でした。ファイルから読むのは、プロセスを管理するスーパーバイザーが、環境変数の値にあるエスケープを処理するためです。パスフレーズにバックスラッシュが含まれていれば、リポジトリ作成時のものと、警告なく異なる値になっていました。実装した月次の復元テストのワークフローは、リポジトリから既知のファイルを復元し、期待する結果と比較する設計です。テストにはパスフレーズが必要なので、実行に成功すれば、その鍵で今もデータを開けることも確認できます。

プラットフォームに二台目のホストを加えようとしたときも、同じ考え方で、すべての運用シークレットをホスト別に分ける方法を決めました。バックアップのリポジトリを示す文字列は、保存先を示します。別々のresticリポジトリは、同じパスフレーズを使っていても、異なるマスターキーを持てます。そのパスフレーズを入手し、両方のリポジトリの暗号化データと鍵ファイルを読める人は、どちらも復号できます。そのため、パスフレーズを共有すると、漏えい時の影響が両方に及び得ます。各シークレットは、ホストをキーとするマップにし、その理由を設定の隣へ書きました。SP 800-57が述べる一般原則も同じです。一つの鍵が守る範囲を制限すると、漏えい時の被害も制限できます。

問題が起きるところ

鍵が、一か所にしかありません。 強い暗号方式は、パスワードを忘れた所有者も、攻撃者も、まったく同じように扱います。resticの文書は、明確に述べています。パスワードを失えばデータも失い、復旧する方法はありません。暗号化した保存領域には、必ず鍵の二つ目の複製が必要です。最初の鍵を壊すものが届かない場所へ置きます。

鍵を、データの隣へ置きます。 同じサーバーの設定に鍵を置く暗号化データベースや、同じバケットにパスフレーズを置くバックアップが止められるのは、ファイルの半分だけを盗む泥棒です。ディスク暗号化でも、より気付きにくい形で同じ問題が起きます。電源を切ったマシンから抜き取ったドライブを守りますが、稼働中のサーバーは、すでにディスクを解錠しています。そのサーバーへ入った攻撃者は、平文を読めます。

秘匿性を、完全性と取り違えます。 認証のない暗号モードで作ったシステムは、中身を隠しても、攻撃者による変更を許します。AEADの構成を使い、タグの検証に失敗したメッセージは、一バイトでも処理する前に破棄します。

nonceを再利用します。 AES-GCMとChaCha20-Poly1305は、同じ鍵の下で、メッセージごとに一意でなければならない値、nonceを受け取ります。RFC 8439は、これを最重要のセキュリティ上の考慮事項と呼びます。同じnonceを使う二つのメッセージは、鍵ストリームと一回限りの認証鍵を共有し、二つの平文のXORが分かってしまいます。カウンターは、その鍵に対して一つだけなら、nonceを作る安全な方法です。同じ鍵を共有する二つのプロセスが、それぞれゼロから数えると、RFCが警告する再利用を、そのまま起こします。

パスワードを、鍵の代わりに使います。 人が選ぶのはパスワードですが、暗号方式が必要とするのは、長さの決まったランダムな鍵です。パスフレーズは、まず鍵導出関数に通します。resticではscryptを使います。RFC 7914は、大量のメモリーを必要とするよう設計し、専用ハードウェアでも、大量のパスワードを試す費用が高いままになるようにしています。保存するログインパスワードは、別の話です。このような関数から得たハッシュとして保存します。復号できるサーバーは、それを流出させることもできるためです。

メタデータは、見えたままです。 暗号化が隠すのは中身で、サイズ、通信のタイミング、両端のアドレスは見えます。TLS 1.3は、送信者がレコードへパディングを加え、サイズを隠せるようにします。ただし、パディングは任意です。そのため、長さから、所有者の予想以上のことが分かる場合がよくあります。

参考資料

- NIST用語集：暗号化 — NISTの文書が使うすべての定義と、それぞれの出典文書です。
- FIPS 197：Advanced Encryption Standard（AES） — ブロックサイズと、三つの鍵長を含む、AESの仕様です。
- NIST SP 800-57第1部・改訂5：鍵管理の勧告 — 鍵の作成、保管、使用、廃止と、漏えい時に起きることです。
- RFC 8446：Transport Layer Security（TLS）プロトコル、バージョン1.3 — `https://` を支えるプロトコルです。AEADだけのレコードと、前方秘匿性のある鍵交換を定めます。
- RFC 8439：IETFプロトコル向けのChaCha20とPoly1305 — ストリーム暗号、認証器、両者を組み合わせるAEADと、nonceの規則です。
- restic：参考資料 — 実際のバックアップツールが、パスワードから鍵を導出し、データを暗号化、認証する方法です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「暗号化とは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/encryption/