FPT Education
FPT UNIVERSITY

Session 4
Cryptography

Presenter: **Mr. Ngo Tung Son (Ph.D.)**

- Head of Information Assurance Department, FPT University, Hanoi, Vietnam
- Director of RISCS Laboratory, FPT University, Hanoi, Vietnam

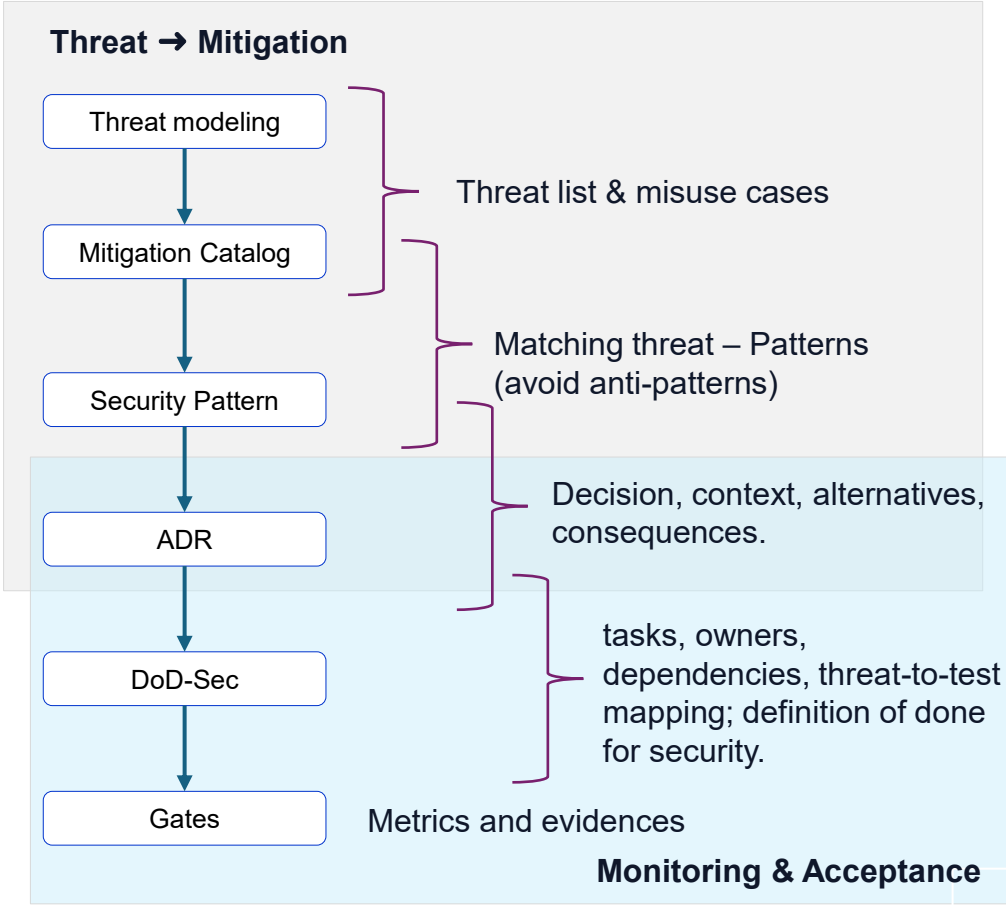
Recap: Mitigations & Security Design Patterns

SUMMARY

Key idea. Mitigations connect *Threat Modeling* with *Secure Design*: focus on structural levers (attack surface, windows of exposure, data exposure), implement via security *patterns*, avoid *anti-patterns*, and enforce through *guards/bottlenecks* to achieve Complete Mediation.
Ý chính. Mitigation nối *Threat Modeling (Ch.2)* với *Thiết kế bảo mật*: ưu tiên đòn bẩy cấu trúc (bề mặt tấn công, cửa sổ phơi bày, lộ dữ liệu), triển khai bằng *pattern*, tránh *anti-pattern*, và ràng buộc qua *guard/bottleneck* để đạt Complete Mediation.

Structural Mitigations to Prioritize

- **Attack surface** ↓: remove entry points, centralize via gateway/PEP, least privilege, network segmentation.
  Giảm bề mặt tấn công: loại điểm vào, gom qua gateway/PEP, tối thiểu đặc quyền, phân đoạn mạng.
- **Windows of exposure** ↓: patch cadence, staged rollout, feature flags/kill-switch, rapid revoke.
  Thu hẹp cửa sổ phơi bày: nhịp vá, phát hành từng bước, cờ tính năng/tắt khẩn, thu hồi nhanh.
- **Data exposure** ↓: data minimization, field-level encryption/tokenization, mTLS, privacy budgets.
  Giảm lộ dữ liệu: tối thiểu dữ liệu, mã hoá/đổi token theo trường, mTLS, ngân sách quyền riêng tư.
- **Patterns**: Defense in Depth, Complete Mediation, Fail Securely, Secure by Default, Reluctance to Trust.
  Pattern: Phòng thủ theo lớp, Kiểm soát toàn diện, Fail Securely, Secure by Default, Thận trọng với tin cậy.
- **Avoid anti-patterns**: Backflow of Trust, Third-Party Hooks, Unpatchable Components, Confused Deputy.
  Tránh anti-pattern: Dòng tin cậy chảy ngược, Móc nối bên thứ ba, Thành phần không vá được, Phó nhầm việc.
- **Guard/Bottleneck**: API Gateway + centralized AuthZ (PDP/PEP), service mesh policy, egress filters.
  Điểm chặn/điểm cổ chai: Gateway + uỷ quyền tập trung (PDP/PEP), policy mesh, lọc egress.



Objectives

- **Understand the role of Cryptography** in Secure-by-Design: protect data *in transit* & *at rest*.

  Hiểu vai trò của Mật mã học trong Secure-by-Design: bảo vệ dữ liệu khi truyền và khi lưu trữ.

- **Review core primitives**: Random Number, Digest/Hash, MAC/HMAC, Encryption (symmetric, asymmetric), Digital Signature, Certificate/PKI, Key Exchange.

  Nắm các công cụ cốt lõi: CSPRNG, hash, MAC/HMAC, mã hoá đối xứng, bất đối xứng, chữ ký số, chứng thư số/PKI, trao đổi khoá.

Cryptography — Overview

"Cryptography is typically bypassed, not penetrated." — Adi Shamir
Mật mã thường bị lách qua, không phải bị phá vỡ.

GROUP — RATIONALE & LIMITS

Pragmatic scope

- Like driver's ed: we learn to **use** the toolkit safely (APIs, modes, keys), not rebuild engines. Math is minimized, with one irresistible exception.
  Giống môn tập lái: học cách vận hành công cụ an toàn (API, mode, key), không phải tự chế động cơ. Toán học được tối giản, chỉ một ngoại lệ thú vị.

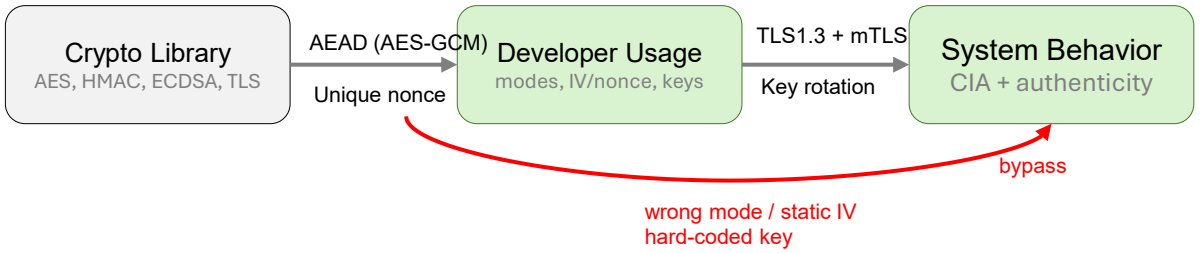
Developer barrier

- Libraries already provide robust crypto; the risk is *misuse*: wrong mode, static IV/nonce, home-rolled protocols, key mishandling.
  Thư viện đã có sẵn; rủi ro là *dùng sai*: chọn sai mode, IV/nonce cố định, tự chế giao thức, quản lý khoá kém.

Analogy: driving stick

- Hands-on practice creates intuition: feel the clutch, hear the engine. Likewise, labs in this chapter build instinct for correct crypto usage.
  Thực hành tạo trực giác: cảm nhận côn, nghe tiếng máy. Tương tự, các lab tạo "cảm giác tay" cho cách dùng mật mã đúng.

ILLUSTRATION —BYPASS VS CORRECT USAGE



EXAMPLE

Minimal envelope encryption

Store files at rest using **AES-GCM** with a random nonce and a **DEK** (data-encryption key) that is itself wrapped by a **KMS** master key (KEK).

1. Generate DEK via CSPRNG; for each file, generate a unique nonce.
   Tạo DEK và nonce(duy nhất cho mỗi lần mã hóa/ký) ngẫu nhiên cho mỗi file (dùng CSPRNG).
2. Encrypt file → {ciphertext, tag} = AES-GCM(DEK, nonce, AAD=metadata).
   Mã hoá file bằng AES-GCM (DEK + nonce) → ra ciphertext và tag; có thể kèm metadata trong AAD.
3. Ask KMS to Wrap(DEK); store wrappedDEK + nonce alongside the file.
   Nhờ KMS "bọc" DEK (cất khoá nhỏ trong két an toàn).
   Lưu wrappedDEK và nonce cạnh file
4. For read: DEK = Unwrap(wrappedDEK), then AES-GCM decrypt and verify tag.
   KMS unwrap DEK → giải mã AES-GCM và kiểm tra tag. Sai tag → từ chối.
5. Rotate: generate new DEK; re-encrypt per policy; keep old DEK until migration completes.
   Xoay khoá: tạo DEK mới, mã hoá lại theo chính sách; giữ DEK cũ cho tới khi chuyển đổi xong.

Cryptography — Tools

CONCEPT & GUIDANCE

Math roots, practical use

- Modern crypto stems from deep mathematics. When implemented correctly, it is highly reliable — not because it's unbreakable, but because breaking it would demand major math breakthroughs.
  Mật mã hiện đại bắt nguồn từ toán học sâu. Dùng đúng cách thì rất tin cậy — không phải vì "bất khả xâm phạm", mà vì cần đột phá toán học lớn mới phá được.

Library-first

- Use high-quality libraries that offer complete solutions at the right abstraction level — so you understand what the API does and avoid home-rolled crypto.
  Dùng thư viện chất lượng, trọn gói, đúng mức trừu tượng — để hiểu API làm gì và tránh tự chế mật mã.

The modern toolbox

- For secure software, a modest set of basic tools suffices. We focus on how to use them properly, not on the underlying math.
  Cho phần mềm an toàn, chỉ cần một bộ công cụ cơ bản. Trọng tâm là cách dùng đúng, không đi sâu toán học nền tảng.

TOOLBOX AT A GLANCE

Basic crypto functions and what their security depends o

Random Numbers (CSPRNG)

Produce unpredictable values for keys, nonces, tokens.

Depends on: OS entropy / secure RNG.

Tạo giá trị khó đoán cho khoá, nonce, token; phụ thuộc nguồn ngẫu nhiên an toàn.

Message Digests (Hash)

Fingerprint of data; used for integrity and signing inputs.

Safe if: collision/second-preimage resistant.

Dấu vân tay dữ liệu; an toàn khi chống va chạm/tìm trước.

Symmetric Encryption

Conceals data with a shared secret (prefer AEAD modes).

Depends on: key secrecy, correct mode, unique nonce.

Che giấu bằng khoá chung; phụ thuộc bí mật khoá, mode đúng, nonce duy nhất.

Asymmetric Encryption

Conceals data using receiver's public key.

Depends on: secure padding/KEM, private-key secrecy.

Che giấu bằng khoá công khai của người nhận; cần padding/KEM an toàn.

Digital Signatures

Authenticate origin; support non-repudiation.

Depends on: private-key secrecy, secure hash/alg.

Xác thực nguồn gốc, không chối bỏ; phụ thuộc bí mật khoá riêng & thuật toán.

Digital Certificates (PKI)

Bind identity to public keys; enable trust chains.

Depends on: CA trust, chain/hostname checks, revocation.

Gắn danh tính với khoá công khai; cần tin cậy CA, kiểm tra chuỗi/tên miền, thu hồi.

Key Exchange (ECDHE)

Establish shared secret over an open channel (PFS).

Depends on: sound groups/curves, ephemeral keys.

Thiết lập bí mật chung qua kênh mở (PFS); dựa vào nhóm/đường cong và khoá tạm thời.

Why Cryptography?

GROUP — RATIONALE & LIMITS

Objective Model

- Reduce likelihood, increase attacker cost, limit impact, and improve detection (Reduce • Resist • Recover)

  GIẢI THÍCH: Giảm xác suất, tăng chi phí tấn công, giảm thiệt hại, và tăng khả năng phát hiện.

Cryto Properties

- **Encryption + MAC/Signatures** provide Confidentiality, Integrity, Authenticity and Non-repudiation over untrusted channels.

  Mã hoá + MAC/Chữ ký số đảm bảo bí mật, toàn vẹn, xác thực, và không thể chối bỏ trên kênh không tin cậy.

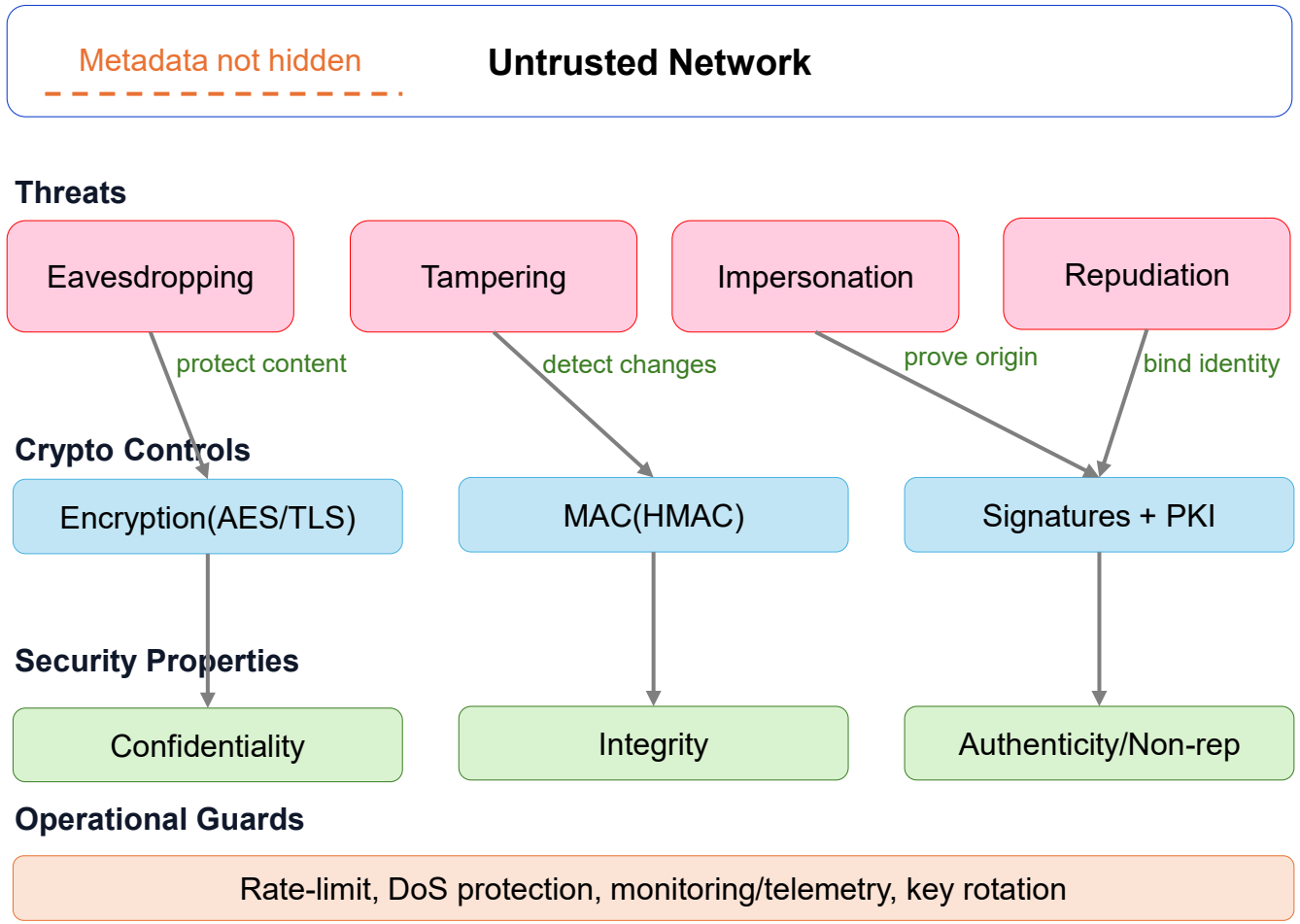
Limitations & ops guards

- Encryption does *not* hide metadata (who talks to whom, when). You still need rate-limit/DoS guards and monitoring.

  GIẢI THÍCH: Mã hoá không che metadata (ai nói với ai, khi nào); vẫn cần giới hạn tốc độ/chống DoS và giám sát.

ILLUSTRATION

Threats → Crypto Controls → Security Properties



Random Numbers (PRNG vs CSPRNG)

CONCEPT

PRNG – Pseudo Random Number Generator

- Algorithmically generated sequence that *appears* random but is **predictable** if the seed/state is known or recovered.
  Chuỗi được tạo theo thuật toán, có vẻ ngẫu nhiên nhưng có thể dự đoán được nếu hạt giống/trạng thái được biết hoặc khôi phục.
- Suitable for simulation, sampling, gaming; **not for cryptographic secrets** (keys, nonces, tokens).
  Thích hợp cho mô phỏng, lấy mẫu, chơi game; không dành cho bí mật mật mã (khóa, nonce, token).
- Examples: LCG, Mersenne Twister, Xorshift, PCG (non-crypto variants).

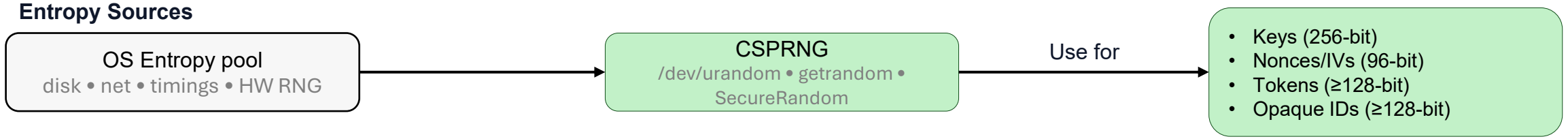
CSPRNG — Cryptographically Secure PRNG

- Outputs are **unpredictable** to any polynomial-time adversary without internal state; designed to resist state recovery.
  Đầu ra không thể dự đoán được đối với bất kỳ kẻ tấn công nào theo thời gian đa thức mà không có trạng thái nội bộ; được thiết kế để chống lại việc khôi phục trạng thái.
- Provides *prediction resistance* and *backtracking resistance* (with proper reseeding after compromise).
  Cung cấp khả năng chống dự đoán và chống quay lui (với khả năng gieo lại thích hợp sau khi xâm phạm).
- Use for **keys**, **nonces/IVs**, **tokens**, **opaque IDs**; target **≥128-bit entropy** for public IDs.
  Sử dụng cho khóa, nonce/IV, mã thông báo, ID mờ; mục tiêu entropy ≥ 128 bit cho ID công khai.
- APIs: OS RNG — Linux getrandom//dev/urandom, macOS/iOS SecRandomCopyBytes, Windows BCryptGenRandom; Lang APIs — Python secrets, Node crypto.randomBytes, Java SecureRandom.

Unpredictability vs Speed

CSPRNG — high unpredictability | PRNG — higher speed

ILLUSTRATION – FLOW : ENTROPY → CSPRNG → SECURE OUTPUTS



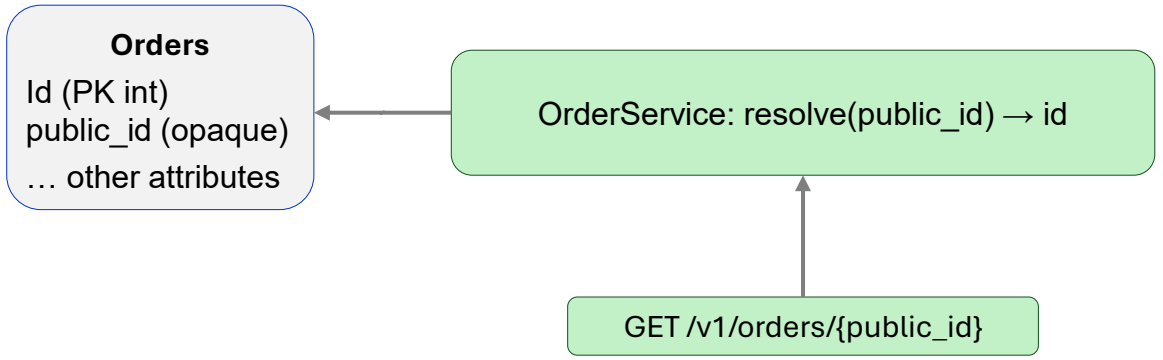
Design Playbook — IDs & Tokens

PLAYBOOK

- **Opaque public IDs:** Generate public-facing IDs with a **CSPRNG** (≥128-bit entropy). No semantics, no PII. Examples: customer, order, password_reset tokens.
  Tạo ID công khai bằng CSPRNG (≥128-bit). Không chứa ý nghĩa/PII. Ví dụ: customer, order, password_reset.
- **Separate internal PK:** Keep an internal primary key (auto-increment/UUIDv7) and a separate **public_id**. Never expose the internal PK in URLs or logs shared externally.
  Duy trì PK nội bộ (tự tăng/UUIDv7) và **public_id** riêng. Không lộ PK nội bộ trên URL/log chia sẻ.
- **Anti-enumeration UX:** Return uniform responses ("*Not found*") for both nonexistent and unauthorized IDs; add **rate-limit**, jitter, and monitoring. Avoid revealing existence via timing or error text.
  Phản hồi đồng nhất cho ID không tồn tại/không có quyền; thêm **rate-limit**, trễ ngẫu nhiên và giám sát. Tránh lộ thông tin qua thời gian/thông điệp lỗi.
- **Format & encoding:** Use a domain prefix + 128-bit random body encoded as **Base32** (~26 chars) or **Base58** (~22 chars). Example: ord_5T8K9... (base32), cus_7Yz3N... (base58).
  Dùng tiền tố theo domain + 128-bit ngẫu nhiên (Base32 ~26 ký tự hoặc Base58 ~22 ký tự).
- **Bug Bar checks:** (1) Public IDs ≥128-bit; (2) CSPRNG only; (3) No PII/semantics; (4) No sequential patterns; (5) Uniform 404/403; (6) Strong rate-limit; (7) Rotate/reset tokens with TTL.

ILLUSTRATION OF OPAQUE

Internal PK ↔ Public ID mapping + anti-enumeration.



EXAMPLE – CREATE PUBLIC ID IN SQL SERVER

```
CREATE OR ALTER FUNCTION dbo.make_public_id (
    @prefix NVARCHAR(16) = N'ord_',
    @bits   INT          = 128      -- 128/192/256 ok; 128-bit enough for public IDs
)
RETURNS NVARCHAR(128)
AS
BEGIN
    DECLARE @len   INT = (@bits + 7) / 8;
    DECLARE @bytes VARBINARY(32) = CRYPT_GEN_RANDOM(@len);
    RETURN @prefix + dbo.base64url_from_varbinary(@bytes);  -- ~22 chars with
                                                                    128-bit
END;
GO
```

Digests / Hash & Collisions

CONCEPT & TERMS

What is a message digest?

- A **hash/digest** maps an arbitrarily long message to a fixed-size output (e.g., 256-bit). It is designed to be **one-way** and to act as a compact *fingerprint* for integrity checks.
  Hash/digest biến thông điệp dài thành đầu ra cố định (vd 256-bit). Mục tiêu là một chiều và là "*dấu vân tay*" gọn để kiểm tra toàn vẹn.

Collision & (Second-) Preimage

- **Collision**: two distinct messages m1 ≠ m2 with the same digest H(m1)=H(m2) (should be infeasible).
  Va chạm: hai thông điệp khác tạo cùng hash (phải bất khả thi).
- **Preimage**: given y, find m with H(m)=y (should be infeasible).
  Tiền-ảnh: tìm thông điệp khớp hash đã cho.
- **Second-preimage** given m1, find m2≠m1 with same digest (should be infeasible).
  Ảnh thứ hai: tìm thông điệp thứ hai trùng hash với thông điệp đã biết.
- They are called collision/preimage attack when they are feasible

Practical Choice

- Use **SHA-256** or **SHA-512** (per your org's standard). Avoid broken/legacy hashes (MD5, SHA-1) for security decisions.
  Dùng SHA-256 hoặc SHA-512 (tuỳ chuẩn tổ chức). Tránh MD5/SHA-1 trong quyết định an toàn.
- don't assume hash = identity; don't truncate excessively without analysis.
  không coi hash là danh tính; không cắt ngắn tuỳ tiện.

Rule of Thumb

- If two digests match, we treat the messages as identical *under the assumption* that finding collisions is infeasible. For **authenticity**, use HMAC/signatures — a hash alone does not prove who wrote it.
  VI: Hash trùng → xem như thông điệp trùng *giả định va chạm khó*. Cần **HMAC/chữ ký** để xác thực nguồn gốc; hash thuần không chứng minh người gửi.

Message Authentication Codes (MAC)

CONCEPT & GUIDANCE

From digests to keyed digests

- A digest is a fixed-size *fingerprint* used for integrity. Add a secret key to the computation → a **MAC**. With the key secret, attackers cannot forge a valid tag.
  Hash là "dấu vân tay" cố định để kiểm toàn vẹn. Thêm **khoá bí mật** vào phép tính → **MAC**. Khi khoá giữ kín, kẻ tấn công không thể giả mạo thẻ.
- A keyed hash (e.g., **HMAC**) effectively yields a *family* of functions (e.g., 2^{256} for a 256-bit key). Without the key, the digest cannot be reproduced forgeries.
  Hash có khoá (vd HMAC) tạo ra *họ* hàm khác nhau (2^{256} với khoá 256-bit). Không có khoá thì không thể tái tạo thẻ hợp lệ.

Using MAC to prevent Tampering

1. both parties share a secret key K in advance.
   hai bên chia sẻ trước một khóa bí mật K.
2. sender calculates $tag = HMAC(K, \text{"hello"})$ and sends it with: message="hello", tag.
   bên gửi tính tag = HMAC(K, "hello") rồi gửi kèm: message="hello", tag.
3. receiver recalculates $tag' = HMAC(K, \text{"hello"})$ and compares constant-time with tag. if someone (Mallory) changes tag or message, for example "hello" → "hallo" but still uses the old tag, then $HMAC(K, \text{"hallo"}) \neq tag$ ⇒ is rejected.
   bên nhận tự tính lại tag' = HMAC(K, "hello") và so sánh constant-time với tag. — nếu ai đó (mallory) sửa 1 byte, ví dụ "hello" → "hallo" nhưng vẫn dùng tag cũ, thì HMAC(K, "hallo") ≠ tag ⇒ bị từ chối.

Design tips: MAC the *exact* canonical form (e.g., method|path|ts|nonce|body); normalize JSON; protect key in KMS/HSM; consider AEAD (AES-GCM) when confidentiality is also needed.
Hãy MAC đúng dạng chuẩn hoá, bảo vệ khoá trong KMS/HSM, cần bí mật thì dùng AEAD.

RELAY RISK AND DEFENSE

Relay Risk

- **What it is**: Attacker resends an old (message, tag) pair that verified before (e.g., re-orders "3 widgets").
  Kẻ tấn công gửi lại một cặp (tin nhắn, thẻ) cũ đã được xác minh trước đó (ví dụ: sắp xếp lại "3 tiện ích").
- **Why MAC alone is not enough**: The pair is still *valid* for that exact old message; integrity ≠ freshness.
  Tại sao chỉ MAC là không đủ: Cặp này vẫn hợp lệ cho chính tin nhắn cũ đó; tính toàn vẹn ≠ tính mới.

Defenses

- **Timestamp window** (e.g., ≤ 300s) and reject stale messages.
  Cửa sổ dấu thời gian (ví dụ: ≤ 300 giây) và từ chối các tin nhắn cũ.
- **Receiver nonce** (challenge) or **sequence number** stored per sender to ensure uniqueness.
  Người nhận lưu trữ nonce (thử thách) hoặc số thứ tự cho mỗi người gửi để đảm bảo tính duy nhất.
- **Idempotency keys** for APIs that may be retried legitimately.
  Khóa bất biến cho các API có thể được thử lại một cách hợp lệ.

- **Implementation Sketch:** Verify timestamp window → check and store nonce/seq (no reuse) → recompute MAC and constant-time compare → proceed.
  Xác minh cửa sổ dấu thời gian → kiểm tra và lưu trữ nonce/seq (không sử dụng lại) → tính toán lại MAC và so sánh thời gian không đổi → tiến hành.

Symmetric Encryption — OTP, AES

CORE IDEAL

What symmetric encryption is and why correct key/mode handling matters

- **Plaintext → Ciphertext** via an encryption algorithm and a secret key; decryption uses the *same* key ⇒ **symmetric cryptography**.
  Văn bản gốc → văn bản mã hoá bằng thuật toán và khoá bí mật; giải mã dùng cùng khoá ⇒ mã hoá đối xứng.
- Security hinges on: choosing a sound algorithm, the right **mode of operation**, correct handling of **keys**, **IV/nonce**, and **padding**, plus **integrity**.
  An toàn phụ thuộc vào: thuật toán đúng, chế độ phù hợp, xử lý đúng khoá, IV/nonce, padding, và tính toàn vẹn.

ONE-TIME PAD (OTP)

- Perfect secrecy via XOR; impractical due to key distribution and reuse risks
  Bí mật tuyệt đối nhờ XOR; thiếu thực tiễn do phân phối khoá và rủi ro tái sử dụng

Encryption: $M \oplus K = C$ and Decryption: $M = C \oplus K$

Example:
M = 'Hi' = 01001000 01101001 ; K = 11010010 01011010
→ $M \oplus K = C$ = 10011010 00110011

SECURITY CHARACTERISTICS AND LIMITATIONS

- **Perfect secrecy** if K is truly random, as long as M, and never reused.
  Bí mật tuyệt đối nếu K ngẫu nhiên thật sự, dài bằng M, không tái sử dụng.
- **Do not reuse the key:** $(M_1 \oplus K) \oplus (M_2 \oplus K) = M_1 \oplus M_2$ leaks structure.
  Cấm dùng lại khoá: công thức trên làm lộ cấu trúc hai thông điệp.
- Impractical: distributing long random keys and managing one-time use.
  Thiếu thực tiễn: phân phối khoá dài và quản trị việc dùng một lần.

ADVANCED ENCRYPTION STANDARD (AES)

- 128-bit block cipher; choose modes to hide patterns and support integrity
  Mã khối 128-bit; chọn mode để ẩn mẫu và hỗ trợ toàn vẹn
- Messages are split into 128-bit blocks; last block is padded (e.g., PKCS#7).
  Chia dữ liệu thành các khối 128-bit; khối cuối được đệm (VD PKCS#7).
- Same symmetric key is used for both encryption and decryption.
  Dùng cùng một khoá đối xứng cho mã hoá và giải mã.

Identical plaintext blocks → identical ciphertext in naive mode (ECB) ⇒ pattern leak.
Khối plaintext giống nhau → ciphertext giống nhau nếu dùng mode đơn giản (ECB) ⇒ lộ mẫu.

Symmetric Encryption — Using Symmetric Cryptography

WHY SYMMETRIC CRYPTO?

The workhorse of modern encryption when applied properly

"Xương sống" của mã hoá hiện đại khi dùng đúng cách

- Protects confidentiality for communication over insecure channels and for data at rest.
  Bảo vệ tính bí mật khi truyền trên kênh không an toàn và khi dữ liệu lưu trữ.
- High performance and widely supported in hardware and software (e.g., AES).
  Hiệu năng cao, hỗ trợ rộng rãi ở phần cứng và phần mềm (VD AES).

KEY ESTABLISHMENT

Algorithms require pre-shared secrets but do not define how to share them

Thuật toán cần khoá dùng chung nhưng không chỉ cách chia sẻ

- Symmetric schemes **assume** Alice & Bob already share a secret key.
  Các sơ đồ đối xứng **giả định** Alice & Bob đã có khoá chung.
- Real-world options: offline exchange, secure channels, or **use asymmetric cryptography** (key agreement/transport) to bootstrap the shared key.
  Tuỳ chọn thực tế: trao đổi ngoại tuyến, kênh an toàn, hoặc **dùng mã hoá bất đối xứng** (thoả thuận/vận chuyển khoá) để khởi tạo khoá chung.

KEY SECRECY

Encryption only works if the secret stays secret — yet available on demand

Mã hoá chỉ hiệu quả khi khoá giữ bí mật — nhưng vẫn sẵn sàng khi cần

- Protect keys at rest and in use: secure storage (HSM/KMS), access control, auditing, rotation.
  Bảo vệ khoá khi lưu và khi dùng: kho lưu trữ an toàn (HSM/KMS), kiểm soát truy cập, audit, luân chuyển.
- Minimize exposure: derive sub-keys (e.g., HKDF), avoid long-lived broad-scope keys.
  Giảm phơi lộ: sinh khoá con (VD HKDF), tránh khoá sống lâu/phạm vi rộng.
- Ensure availability with disaster recovery plans; design for least privilege.
  Đảm bảo sẵn sàng với kế hoạch DR; thiết kế theo nguyên tắc ít đặc quyền.

KEY SIZE

Bigger keys resist brute force but are harder to manage

Khoá lớn khó bị vét cạn nhưng khó quản trị

Mã hoá chỉ hiệu quả khi khoá giữ bí mật — nhưng vẫn sẵn sàng khi cần

- Larger keys → higher computational cost to guess; one-time pad is theoretically ideal but impractical.
  Khoá lớn → chi phí tính toán vét cạn cao hơn; one-time pad lý tưởng về lý thuyết nhưng thiếu thực tiễn.
- Choose modern defaults (e.g., AES-128/256) based on security requirements and performance.
  Chọn mặc định hiện đại (VD AES-128/256) tuỳ yêu cầu an toàn và hiệu năng.

Asymmetric Encryption

WHAT IS ASYMMETRIC ENCRYPTION?

A counter-intuitive one-way setup: anyone encrypts to you; only you decrypt

Một cơ chế "một chiều" ngược trực giác: ai cũng mã hoá gửi bạn; chỉ bạn mới giải mã.

- Uses a key pair: a public key (shared) for encryption and a private key (secret) for decryption.
  Dùng cặp khoá: khoá công khai (chia sẻ) để mã hoá và khoá bí mật (giữ kín) để giải mã.
- Knowing one key doesn't reveal the other; the two transformations are inverse functions.
  Biết một khoá không suy ra được khoá kia; hai phép biến đổi là nghịch đảo của nhau.

KEY ESTABLISHMENT

Algorithms require pre-shared secrets but do not define how to share them

Thuật toán cần khoá dùng chung nhưng không chỉ cách chia sẻ

- Symmetric schemes **assume** Alice & Bob already share a secret key.
  Các sơ đồ đối xứng **giả định** Alice & Bob đã có khoá chung.
- Real-world options: offline exchange, secure channels, or **use asymmetric cryptography** (key agreement/transport) to bootstrap the shared key.
  Tuỳ chọn thực tế: trao đổi ngoại tuyến, kênh an toàn, hoặc **dùng mã hoá bất đối xứng** (thoả thuận/vận chuyển khoá) để khởi tạo khoá chung.

RSA – CORE IDEA

Easy to multiply big primes; hard to factor the product (trapdoor function)

Nhân số nguyên tố lớn thì dễ; phân tích thừa số của tích đó thì cực khó (hàm bẫy)

- Pick random large primes p, q; compute N = p·q (keep p, q secret).
  Chọn các số nguyên tố lớn ngẫu nhiên p, q; tính N = p·q (giữ kín p, q).
- Derive key pair ⟨e, d⟩ from N and φ(N); publish (N, e); keep d secret.
  Suy ra cặp khoá ⟨e, d⟩ từ N và φ(N); công bố (N, e); giữ kín d.
- Inverse property: D(E(x)) = x and E(D(x)) = x for 0 < x < N.
  Tính nghịch đảo: D(E(x)) = x và E(D(x)) = x với 0 < x < N.

RSA Encrypt and Decrypt

Encryption (Bob → Alice): $C = E_A(M)$ using Alice's public key.
$C = E_A(M)$ dùng khoá công khai của Alice.

Decryption (Alice): $M = D_A(C)$ using Alice's private key.
Giải mã (Alice): $M = D_A(C)$ dùng khoá bí mật của Alice.

SECURITY NOTES — PADDING & RANDOMNESS

Public keys enable guessing attacks; randomized padding defeats them

Khoá công khai tạo điều kiện đoán mẫu; padding ngẫu nhiên sẽ hoá giải

- With public key known, attackers can test likely messages (chosen-plaintext). **Textbook RSA is deterministic.**
  Vì khoá công khai ai cũng biết, kẻ tấn công có thể thử các thông điệp đoán trước (chosen-plaintext). RSA "sách giáo khoa" là tất định.
- Mitigation: use randomized padding (e.g., **RSA-OAEP**) so the same message encrypts to different ciphertexts.
  Giảm thiểu: dùng padding ngẫu nhiên (VD RSA-OAEP) để cùng một thông điệp cho bản mã khác nhau.

Do not roll your own RSA. Use library primitives like RSA-OAEP for encryption and RSA-PSS for signatures.
Không tự cài RSA. Dùng primitive thư viện như RSA-OAEP (mã hoá) và RSA-PSS (chữ ký).

Digital Signatures (Authenticity / Non-Repudiation)

WHAT AND WHY?

Authenticity and non-repudiation — next to confidentiality in importance

Xác thực nguồn gốc và không thể chối bỏ — quan trọng chỉ sau tính bí mật

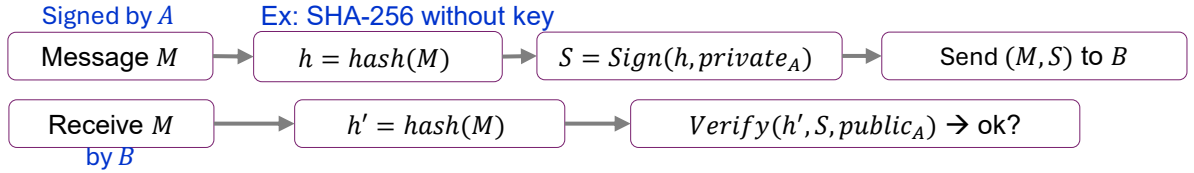
- Digital signatures give the receiver assurance a message is truly from the signer.
  Chữ ký số bảo đảm người nhận rằng thông điệp thực sự đến từ người ký.
- They also serve as **evidence** if the signer later denies having sent it.
  Đồng thời là bằng chứng nếu người ký về sau phủ nhận việc đã gửi.
- Signatures are **independent** of encryption: you may sign, encrypt, or both.
  Chữ ký độc lập với mã hoá: có thể chỉ ký, chỉ mã hoá, hoặc làm cả hai.

EXAMPLE – SIGN & VERIFY

Same key pair as encryption; only the signer can compute the signature function

Dùng cùng cặp khoá như mã hoá; chỉ người ký mới tính được hàm chữ ký

- Signing (Alice): S = S_A(M) using **Alice's private key**.
  Ký (Alice): S = S_A(M) dùng khoá bí mật của Alice.
- Verification (Bob): check whether M = V_A(S) using **Alice's public key** (and N).
  Kiểm (Bob): kiểm tra M = V_A(S) dùng khoá công khai của Alice (và N).



PUBLIC VERIFICATION & EVIDENCE

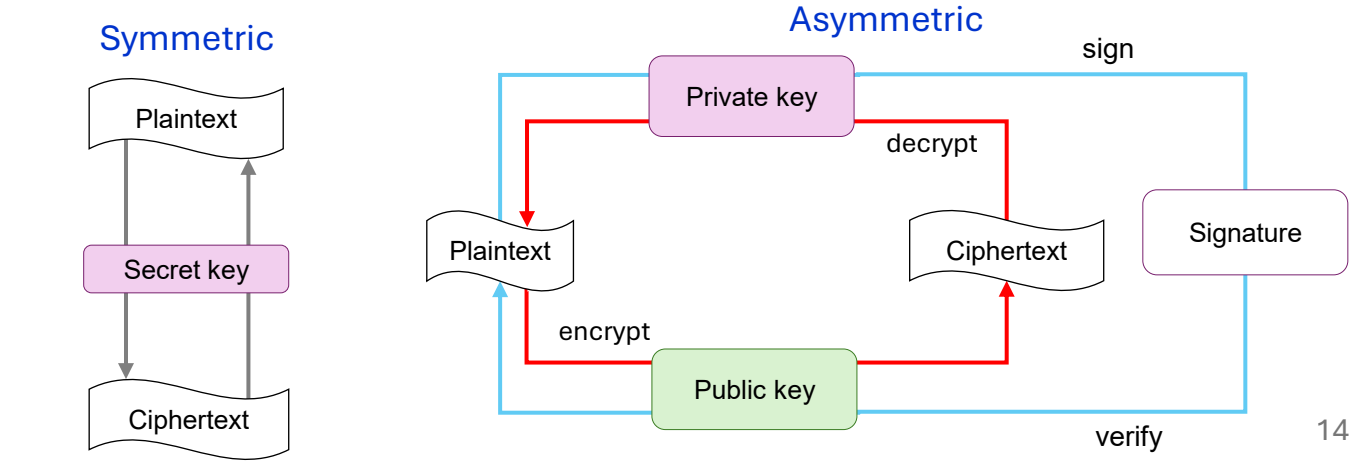
Anyone with the public key can verify without exposing any secret

Ai có khoá công khai cũng kiểm được mà không lộ bí mật

- Verification uses only the **public key**, so Bob can prove Alice signed a message to a third party without risking her private key.
  Việc kiểm chỉ dùng khoá công khai, nên Bob có thể chứng minh Alice đã ký với bên thứ ba mà không rủi ro lộ khoá bí mật.
- Contrast: symmetric keys are **shared** by both parties, so true signing is not possible.
  với khoá đối xứng là dùng chung, nên không thể có chữ ký đúng nghĩa.

SYMMETRIC VS ASYMMETRIC

Why only asymmetric enables signatures while revealing no secrets in verification

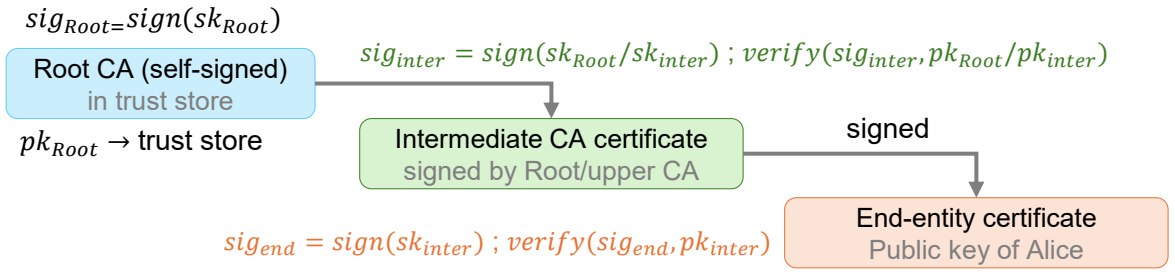


Digital Certificates

WHY DIGITAL CERTIFICATES?

Problem: how to **announce** and **trust** a public key at Internet scale?
Vấn đề: công bố và tin cậy khoá công khai ở quy mô Internet như thế nào?

- Publish one public key for everyone — but it must be trustworthy.
  Công bố một khoá công khai cho mọi người — nhưng phải đáng tin



WHAT INSIDE A CERTIFICATE?

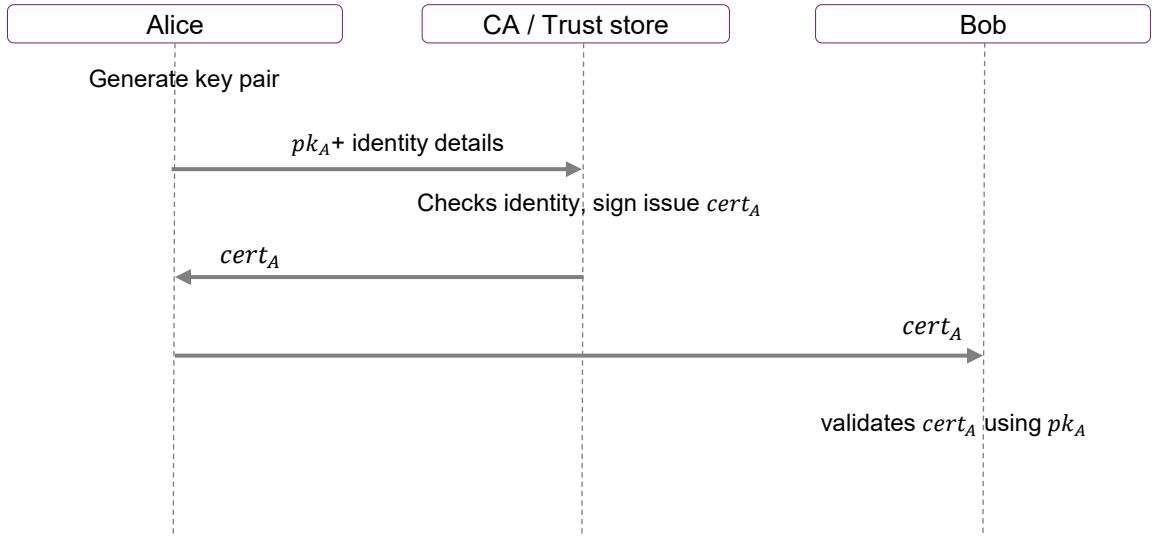
- **Subject** (e.g., "Alice", domain, email), **Subject Public Key Info (SPKI)**.
  Chủ thể (VD "Alice", tên miền, email), SPKI.
- **Issuer** (CA), **Validity** (notBefore/notAfter), **Serial**, **Extensions** (Key Usage, SAN…).
  Bên cấp (CA), Hiệu lực (notBefore/notAfter), Số sê-ri, Mở rộng (Key Usage, SAN…).
- CA signs the **digest** of the certificate → authenticity.
  CA ký digest của chứng thư → xác thực.

CERTIFICATE AUTHORITIES (CA) & ROOT STORES

CAs sign certificates; operating systems and browsers trust a set of root Cas.
CA ký chứng thư; hệ điều hành và trình duyệt tin một tập CA gốc.

- A CA publishes its own **root certificate** (self-signed) and is distributed in trust stores.
  CA công bố **chứng thư gốc** (tự ký) và được phân phối trong kho tin cậy.
- CAs collect applicants' public keys and issue **signed certificates** that bind identity → key.
  CA thu nhận khoá công khai của người đăng ký và phát hành chứng thư đã ký gắn danh tính → khoá.

ISSUANCE & VALIDATION FLOW



Key Exchange — Diffie and Hellman

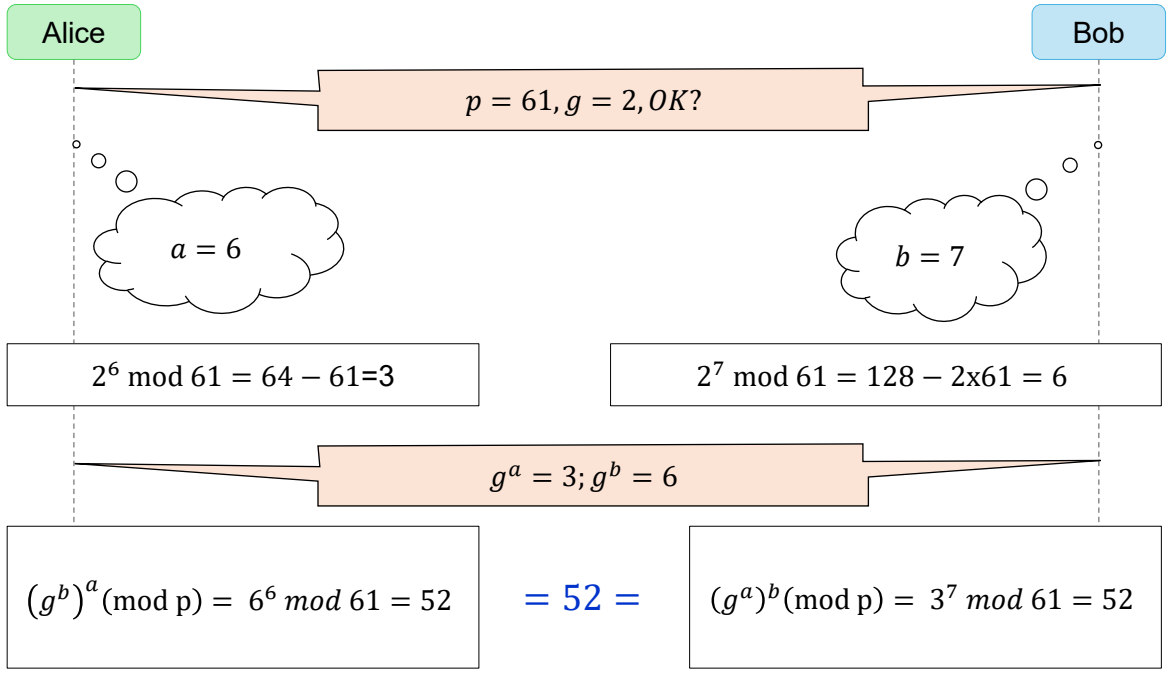
WHAT IS KEY EXCHANGE?

Share a secret while Mallory sees all the traffic

Chia sẻ bí mật trong khi Mallory nhìn thấy toàn bộ lưu lượng

- No pre-shared secret, no trusted CA required to start.
  Không cần khoá dùng chung từ trước, cũng không cần CA để bắt đầu.
- DH lets Alice & Bob derive the **same secret** from different one-time values they sent publicly.
  DH cho phép Alice & Bob suy ra **cùng một bí mật** từ các giá trị một lần gửi công khai.

WORKED EXAMPLE (SMALL NUMBERS)



DIFFIE AND HELLMAN KEY EXCHANGE ALGORITHM

Openly agree ⟨p, g⟩, exchange powers, derive the same secret

Công khai thống nhất ⟨p, g⟩, trao đổi lũy thừa, suy ra bí mật giống nhau

1. Agree publicly on a prime p and a base g where $1 < g < p$.
2. Alice picks random a, sends $A = g^a \pmod p$.
3. Bob picks random b, sends $B = g^b \pmod p$.
4. Alice computes $S = B^a \pmod p$; Bob computes $S = A^b \pmod p$.
5. Both get the same S (discrete-log trapdoor).

Use large safe primes and a generator g; choose a, b via a CSPRNG.

Dùng số nguyên tố lớn và g phù hợp; chọn a, b bằng CSPRNG.

MODERN PRACTICE – ECDH & TLS

Ephemeral ECDH, forward secrecy, and hybrid with symmetric crypto

ECDH tạm thời, bí mật chuyển tiếp, và kết hợp với mã hoá đối xứng.

- Most systems use **ECDH** for performance and smaller keys.
  Đa số hệ thống dùng ECDH vì hiệu năng và khoá nhỏ.
- **Ephemeral (DHE/ECDHE)** keys yield **forward secrecy** — past sessions stay safe if a long-term key leaks.
  Khoá tạm thời (DHE/ECDHE) đem lại forward secrecy — phiên cũ vẫn an toàn nếu khoá dài hạn rò rỉ.
- TLS: perform key exchange over TCP, derive a shared secret, then switch to symmetric ciphers (e.g., AES-GCM).
  TLS: trao đổi khoá qua TCP, suy ra bí mật chung, sau đó dùng mã đối xứng (VD AES-GCM).

Key Management Lifecycle

KMS OVERVIEW

CREATE – ENTROPY & STORAGE

Generate with CSPRNG/KMS; store in HSM/KMS/Vault; enforce separation of duties

Sinh bằng CSPRNG/KMS; lưu hardware security modules/KMS/Vault; tách nhiệm vụ

- Use **CSPRNG** or KMS to generate keys; label purpose/scope (CMK vs DEK).
  Dùng CSPRNG/KMS sinh khoá; gắn nhãn mục đích/phạm vi (CMK vs DEK)
- Protect material in **HSM** or managed **KMS**; access via roles/policies (SoD).
  Lưu trong HSM/KMS; truy cập qua vai trò/chính sách (tách nhiệm vụ)

ROTATE – POLICY & PROCESS

Periodic/triggered rotation; re-wrap vs re-encrypt; escrow & backups

Luân chuyển định kỳ/khi có sự cố; re-wrap hay re-encrypt; ký quỹ & sao lưu

- Policy e.g., DEK: 90 days, CMK: 12 months, or on compromise/role change.
  Chính sách: VD DEK 90 ngày, CMK 12 tháng, hoặc khi có sự cố/đổi vai trò.
- Prefer **re-wrap** DEKs with new CMK to avoid re-encrypting large data; re-encrypt on algorithm change.
  Ưu tiên re-wrap DEK; re-encrypt khi đổi thuật toán
- Maintain **escrow/backups** for disaster recovery with strict SoD and access logging.
  Có escrow/backup với SoD và log chặt.

USE – ACCESS & ENVELOPE

Least privilege for decrypt/encrypt; envelope encryption maps CMK↔DEK↔Data

Ít đặc quyền cho decrypt/encrypt; bao bọc khoá CMK↔DEK↔Dữ liệu

- App requests decrypt of wrapped DEK (never sees CMK); uses DEK in memory, then zeroizes.
  Ứng dụng yêu cầu giải bọc DEK (không thấy CMK); dùng DEK trong RAM rồi xoá sạch;
- Tag ciphertext with key-id, alg, creation-time.
  gắn nhãn key-id, alg, creation-time

RETIRE – REVOKE & SECURE DELETION

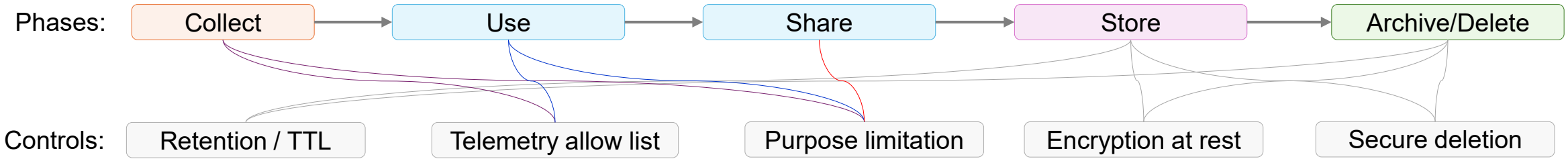
Disable, schedule deletion; crypto-shred by destroying key material; keep proofs

Vô hiệu, lên lịch xóa; "crypto-shred" bằng cách hủy vật liệu khoá; lưu bằng chứng

- Revoke old keys (disable for decrypt); set **deletion windows**; purge caches/replicas.
  Thu hồi khoá cũ; đặt cửa sổ xóa; dọn cache/bản sao
- Produce **deletion certificates**; keep audit trail of who approved, when, and what was destroyed.
  Tạo chứng chỉ xóa; lưu vết kiểm toán.

Data Lifecycle & Policy Alignment

DATA LIFECYCLE - OVERVIEW



Retention / TTL

- Define policy per data class (e.g., logs: 30–90d, PII: 12–24m with justification).
- Implement TTL/retention in DB/object store; track created_at, ttl_expires_at.

Chính sách theo lớp dữ liệu (VD log: 30–90 ngày, PII: 12–24 tháng có lý do). Áp dụng TTL trong DB/kho đối tượng.

Secure Deletion

- Crypto-shred: destroy keys (DEK/segment keys) → ciphertext unusable.
- For plaintext replicas/indexes/caches: schedule purge; verify and log evidence.

Crypto-shred: huỷ khoá (DEK/segment) → bản mã vô dụng; dọn bản sao/chỉ mục/cache, có log bằng chứng.

Encryption at Rest

- Use **DEK** for content; wrap with **CMK**; store key-id and alg alongside ciphertext.
- Protect backups/archives with same policy; test restores respect encryption.

Dùng DEK mã hoá dữ liệu; bọc bằng CMK; lưu key-id/alg; áp dụng cho backup/archiving.

Purpose Limitation

- Tag data with purpose and lawful_basis; block queries outside allow-list.
- Break glass flow with approvals & audit for exceptional access.

Gắn nhãn purpose/lawful_basis; chặn truy vấn ngoài danh trắng; quy trình "break glass" có duyệt & audit.

Telemetry Allow list

- Define allowed events & fields (e.g., counters, performance) — **no raw identifiers**.
- Hash/tokenize if needed; strip payloads; review changes via privacy change control.

Định nghĩa sự kiện/trường được phép (đếm, hiệu năng) — không định danh thô; băm/ẩn danh nếu cần; kiểm soát thay đổi.

Crypto Anti-Patterns

ANTI PATTERNS TO AVOID

Common crypto mistakes that create exploitable weaknesses

- ECB mode, nonce/IV reuse:
  - ✓ ECB encrypts each block independently → identical plaintext blocks ⇒ identical ciphertext; leaks patterns, enables replay/chosen-block tricks.
  - ✓ Reusing IV/nonce with the same key in stream/AEAD modes (GCM/ChaCha20-Poly1305) breaks confidentiality and can allow tag forgery.

    ECB lộ mẫu; tái sử dụng IV/nonce với cùng khoá có thể làm lộ dữ liệu và giả mạo tag.
- Weak Randomness :
  - ✓ Time-seeded PRNGs are **predictable** → attackers can guess keys/nonces/tokens.
  - ✓ Identifiers like UUIDv1 embed time/MAC; not suitable for secrets.

    PRNG dựa thời gian dễ đoán; UUIDv1 chứa timestamp/MAC, không dùng cho bí mật.
- Keys — Hard-coding, Missing Rotation/Revocation:
  - ✓ Hard-coded or shared keys leak via repo history, logs, or build artifacts.
  - ✓ No rotation/revocation ⇒ compromised keys remain valid indefinitely.

    Khoá hard-code dễ rò rỉ; không xoay/thu hồi khiến khoá lộ vẫn dùng được.
- Signatures & JWT — Verification Pitfalls
  - ✓ Algorithm confusion: accepting token-declared alg (e.g., RS↔HS) or alg=none.
  - ✓ Unpinned JWKS: trusting attacker-controlled jku; skipping iss/aud, exp/nbf, jti.

    Nhầm thuật toán, tin JWKS tuỳ ý, bỏ qua kiểm tra thời gian/audience/jti.
- Error Handling:
  - ✓ Uniform client errors; detailed reasons only in server logs

    Thông điệp lỗi đồng nhất cho client; chi tiết chỉ ghi log máy chủ
  - ✓ Different messages reveal state (user exists? key valid?) → enumeration/oracles.
  - ✓ Timing/format differences can leak verification results.

    Thông điệp/độ trễ khác nhau làm lộ thông tin.
- Predictable IDs & Timestamp Tokens
  - ✓ Sequential IDs enable scraping/inference of activity volume and relationships.
  - ✓ Timestamps used as tokens/nonces are guessable and may collide.

    ID tuần tự dễ dò; timestamp làm token/nonce có thể đoán được.

Using Crypto(Session Wrap-Up)

CRYPTO TOOLBOX

What each tool give you

- **CSPRNG** — unpredictable bits.
- **Digests / MAC** — integrity & equality tests.
- **Symmetric crypto** — fast confidentiality.
- **Asymmetric crypto** — public/private key tricks (encrypt for, or verify from).
- **Digital signatures** — authenticity + non-repudiation.
- **Certificates (PKI/CA)** — share public keys safely.
- **Key exchange (DH/ECDH)** — agree secrets over open networks.

PRACTICAL RECIPES — IN TRANSIT & INTEGRITY

Confidentiality + integrity with existing services

Bảo mật & toàn vẹn bằng dịch vụ sẵn có

- Internet link: use TLS 1.3 (ECDHE + AEAD). kênh internet dùng TLS 1.3.
- Message integrity: sign or use HMAC. toàn vẹn: chữ ký/HMAC.
- Endpoint auth: certs/JWKS + iss/aud/exp/nbf/jti. xác thực đầu cuối.

LIMITS — METADATA & TRAFFIC ANALYSIS

Even encrypted systems leak patterns (size, timing, frequency)

Ngay cả khi mã hoá, vẫn có thể lộ mẫu (kích thước, thời gian, tần suất)

- Example: motion-trigger camera → low traffic when nobody's home.
  Ví dụ: camera phát hiện chuyển động → lưu lượng giảm khi vắng nhà.
- Mitigate: batching, padding, cover traffic, and access patterns minimization.

FORWARD LOOK — MATH & MACHINES EVOLVE

Algorithms may be broken; quantum is a looming factor

Thuật toán/hardware tiến hoá; nguy cơ lượng tử trong tương lai

- Track deprecations, rotate to modern curves/AEAD; avoid DIY crypto.
  Theo dõi thuật toán bị loại; xoay sang giải pháp hiện đại; tránh tự chế.
- Post-quantum: plan crypto-agility; inventory keys/certs/usages.

SECURITY VS THE "5$ WRENCH"

Strong crypto shifts attackers to *circumvention*

Mật mã mạnh khiến kẻ tấn công vòng tránh qua con người/quy trình

- **Takeaway:** Use crypto *correctly*, and harden the surrounding people/process.
  Thông điệp: Dùng crypto *đúng cách*, gia cố con người/quy trình.

PATTERN — ENCRYPT EARLY, DECRYPT LATE

Frontend encrypts with public key; only a hardened backend holds the private key

Mã hoá sớm ở frontend (khóa công khai); giải mã muộn ở backend (khóa bí mật)

- **Flow**: FE receives credit card → encrypt with **gateway public key** → pass opaque blob through services → decrypt only inside **PCI-zone** host with private key.
  FE nhận thẻ → mã hoá bằng public key → truyền như blob → chỉ giải mã trong vùng bảo vệ.
- **Benefit**: Most code never sees plaintext; limits blast radius.
  đa số code không thấy dữ liệu rõ; giảm rủi ro.

PATTERN — SPLIT STORAGE (DATA VS KEY)

Encrypted backups with provider; keep keys in your own vault

Gửi bản mã cho bên lưu trữ; giữ khoá ở két của bạn

- **Trusts**: provider ↔ integrity/availability; you ↔ confidentiality via key custody.
  nhà cung cấp chịu trách nhiệm toàn vẹn/sẵn sàng; bạn giữ bí mật nhờ giữ khoá.
- **Practice**: envelope encryption (DEK↔CMK via KMS/HSM); test restore w/ encryption intact.