Week 08 — Authentication (P1): MAC and HMAC Cryptographic Hash Functions (SHA-2 / SHA-3)

NT219 — Cryptography

October 29, 2025

- Course goals touched today: Integrity, Authentication, Non-repudiation (via DS), and Keyed MACs.

Thuật ngữ: toàn vẹn dữ liệu; xác thực; không chối bỏ; mã xác thực thông điệp

Outline

1. Motivation & Security Notions (vs CRC)
2. Merkle–Damgård (MD) framework and pitfalls
3. SHA-2 family — structure, padding, schedule, rounds
4. Length-Extension attacks & why HMAC fixes them
5. SHA-3/Keccak — sponge construction, security bounds
6. Engineering guidance, side-channels, and pitfalls

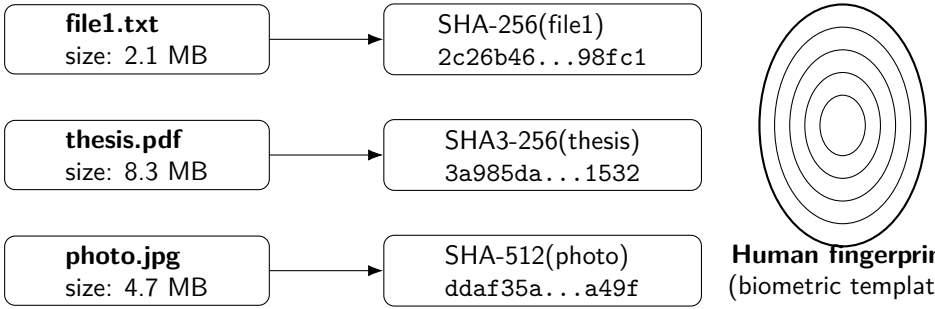
Thuật ngữ: khung MD; phần đệm (padding); lịch thông điệp; tấn công mở rộng độ dài; bọt biển (sponge)

Why Hash Functions?

- **Hash Function.** A public, deterministic function $H : \{0,1\}^{\to\{0,1\}^{\ell}}$ that maps any-length input to a fixed-length digest of ℓ bits (e.g., $\ell \in \{224, 256, 384, 512\}$).
- Compact, fixed-length *digest* of arbitrary-length input M.
- **Security notions (for an ℓ-bit hash H):**
  - **Preimage:** given random $y \leftarrow \{0,1\}^{\ell}$, find x with $H(x)=y$ (work $\approx 2^{\ell}$).
  - **Second-preimage:** given x, find $x' \neq x$ (often with $|x'|=|x|$) s.t. $H(x')=H(x)$ (work $\approx 2^{\ell}$).
  - **Collision:** find any $x \neq x'$ with the same digest (birthday bound $\approx 2^{\ell/2}$).
- **Sizing rule of thumb:** to target λ-bit collision security pick $\ell \geq 2\lambda$ (e.g., $\ell=256 \Rightarrow$ collisions $\approx 2^{128}$); preimage security $\approx 2^{\ell}$.
- CRCs detect random errors but are *linear* and *not* cryptographically secure.

Thuật ngữ: tham số an toàn λ

Digital "Fingerprint" vs Human Fingerprint (Analogy)



- **Files (similarity = equality):** decide match by digest equality $H(A) = H(B)$. Any bit flip almost surely changes the digest (avalanche).
- **Persons (similarity = score):** a matcher outputs a score $s \in [0, 1]$; accept if $s \geq \tau$ (threshold). Tune τ to trade off FAR/FRR.
- **Security note (hash size):** with an ℓ-bit hash, collisions need $\approx 2^{\ell/2}$ trials (birthday); preimages $\approx 2^{\ell}$.

Thuật ngữ: exact match — khớp chính xác; similarity score — điểm tương tự; threshold — ngưỡng; FAR/FRR — tỉ lệ nhận sai/từ chối sai; avalanche effect — hiệu ứng thác lũ;

CRC and modulo 2^{32} checksum

Concrete collisions: CRC and modulo 2^{32} checksum

(A) CRC (tiny, explicit)
Use CRC-4-ITU (poly $x^4 + x + 1$), init= 0, xorout= 0, no reflection.

$$\underbrace{\texttt{0x03}}_{M_1=00000011} \implies \mathrm{CRC4}(M_1) = \texttt{0x5}, \qquad \underbrace{\texttt{0x10}}_{M_2=00010000} \implies \mathrm{CRC4}(M_2) = \texttt{0x5}.$$

$M_1 \neq M_2$ but the CRC remainders are **equal** (0x5) $\Rightarrow$ a collision.

(B) Modulo-2^{32} additive checksum (explicit)
Let the checksum be the sum of 32-bit words mod 2^{32}. Consider:

$$\begin{aligned} M &= (\texttt{0x00000001},\ \texttt{0x00000002},\ \texttt{0x00000003}) && \Rightarrow \sum M \equiv \texttt{0x00000006} \pmod{2^{32}}, \\ M' &= (\texttt{0x40000001},\ \texttt{0x00000002},\ \texttt{0xC0000003}) && \Rightarrow \sum M' = (\texttt{0x40000001} + \texttt{0x00000002} + \texttt{0xC0000} \\ & && = \texttt{0x100000000} \pmod{2^{32}}. \\ & && \equiv \texttt{0x00000006} \pmod{2^{32}}. \end{aligned}$$

Thus $M \neq M'$ but the modulo-2^{32} checksums are **identical** $\Rightarrow$ a collision.

Hash vs CRC

CRC

Linear over $\mathrm{GF}(2)$; easy to forge with chosen differences; great for noise, not for attackers.

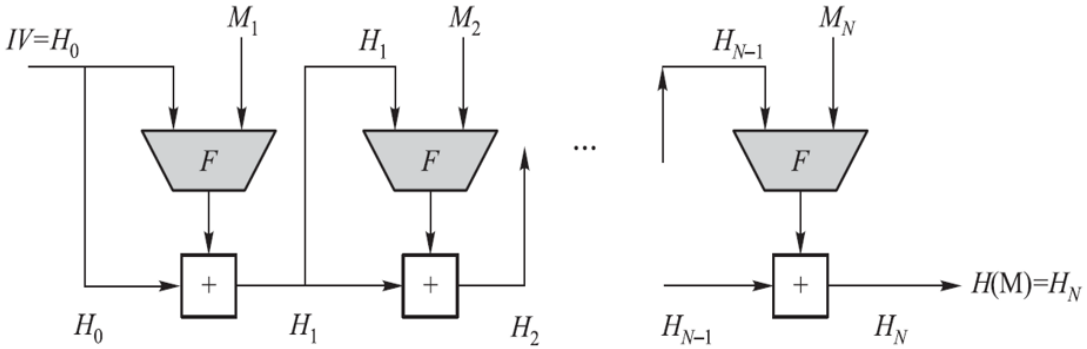
Cryptographic Hash

Nonlinear mixing; avalanche; large state; designed against adversaries.

- **Use-case split:** CRC for channel error detection; Hash/HMAC for authenticity and integrity.

Thuật ngữ: collision (CRC) — va chạm (CRC); fixup bytes — các byte bù; no key required — không cần khóa; authenticity — tính xác thực

Merkle–Damgård (MD) Framework



$$H_0 = \mathrm{IV}, \quad H_{i+1} = F(H_i, M_{i+1}),\ i = 0 \ldots k-1, \quad H(M) = H_k$$

Merkle–Damgård (MD) Framework

- Parse M into k blocks after MD-strengthening padding; iterate compression F:

$$H_0 = \mathrm{IV}, \quad H_{i+1} = F(H_i, M_{i+1}),\ i = 0 \ldots k-1, \quad H(M) = H_k$$

- MD-strengthening: append a '1' bit, then zeros, then encoded length $|M|$; prevents trivial length-collisions.
- **Pitfall:** Plain MD is vulnerable to *length-extension* if F is Merkle–Damgård compatible.

Thuật ngữ: tăng cường MD; véc-tơ khởi tạo; hàm nén; mở rộng độ dài

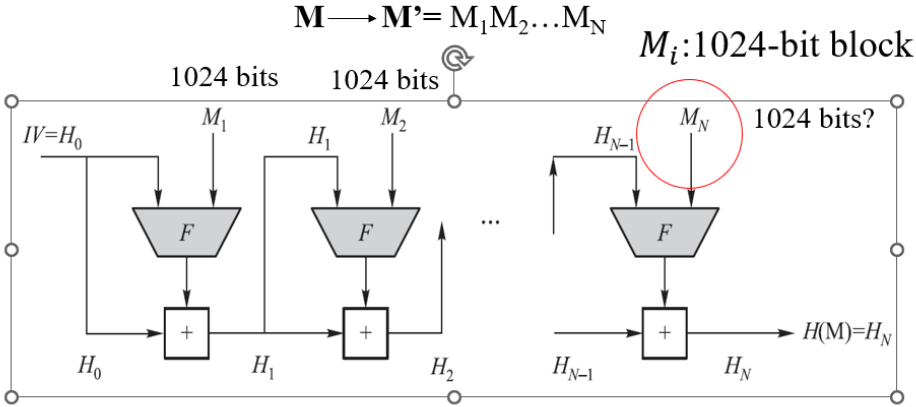
SHA-2 Family Overview

- Variants: SHA-224/256 (32-bit words); SHA-384/512/512/224/512/256 (64-bit words).
- Block size: 512 bits (SHA-256) or 1024 bits (SHA-512). Output: 224, 256, 384, or 512 bits.
- Constants K_t: first bits of fractional parts of cube roots of primes; IV from square roots of primes.

Thuật ngữ: từ 32/64-bit; hằng số vòng; véc-tơ khởi tạo

Example: SHA-512

Padding process



- **Length(M)=L**
- **$M' = M \| 1(0^{\ell}) \| b_{128}(L)$, where $\ell \geq 0$**

Figure: SHA-512 algorithm

SHA-512 Padding — Correct Formula & Example

Let $L = |M|$ (in bits). Padding forms $M' = M \,\|\, 1 \,\|\, 0^{\ell} \,\|\, \mathrm{len}_{128}(L)$ where ℓ is the smallest ≥ 0 such that:

$$L + 1 + \ell + 128 \equiv 0 \pmod{1024} \quad \Longleftrightarrow \quad \ell = (896 - (L+1)) \bmod 1024.$$

Example $M =$ `abc`**:** $L = 24$. Then $\ell = 896 - 25 = 871$. Total becomes 1024 bits (one block). *SHA-256* uses the same pattern but with 64-bit length field and modulus 512.

Thuật ngữ: độ dài tính theo bit; trường hợp `abc`; phần đệm đúng

SHA-2 Round Functions (64-bit variants)

$$
\begin{aligned}
\mathrm{Ch}(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\
\mathrm{Maj}(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\
\Sigma_0(x) &= \mathrm{ROTR}^{28} x \oplus \mathrm{ROTR}^{34} x \oplus \mathrm{ROTR}^{39} x \\
\Sigma_1(x) &= \mathrm{ROTR}^{14} x \oplus \mathrm{ROTR}^{18} x \oplus \mathrm{ROTR}^{41} x \\
\sigma_0(x) &= \mathrm{ROTR}^{1} x \oplus \mathrm{ROTR}^{8} x \oplus (x \gg 7) \\
\sigma_1(x) &= \mathrm{ROTR}^{19} x \oplus \mathrm{ROTR}^{61} x \oplus (x \gg 6)
\end{aligned}
$$

Message schedule W_t ($t = 0..79$) and working variables ($a..h$) follow the FIPS-180-4 specification.

Thuật ngữ: xoay phải (ROTR); dịch phải; chọn (Ch); số đông (Maj)

SHA-512 Core (one round, schematic pseudocode)

```
for t = 0..79:
  T1 = h + 1(e) + Ch(e,f,g) + K[t] + W[t]
  T2 = 0(a) + Maj(a,b,c)
  h = g; g = f; f = e;
  e = d + T1;
  d = c; c = b; b = a;
  a = T1 + T2;
```

- Initialization from IV; after 80 rounds add back to state ($a..h$).
- Distinct constants and rotation offsets for 32-bit (SHA-256) vs 64-bit (SHA-512).

Thuật ngữ: khởi tạo; cộng vòng; hằng số K; lịch thông điệp

Known Test Digests (sanity checks)

- SHA-256("abc") =
  `ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad`
- SHA-512("abc") =
  `ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9eeee64b55d39a`
  `2192992a274fc1a836ba3c23a3feebbd454d4423643ce80e2a9ac94fa54ca49f`

Thuật ngữ: chuẩn kiểm thử; băm mẫu abc

Length-Extension (informal intuition)

- For many MD hashes (e.g., SHA-1/SHA-256/SHA-512), given $H(M)$ and $|M|$, attacker can compute $H(M \| \text{pad}(M) \| M')$ *without* knowing M.
- Breaks naive "MAC $= H(K \| M)$" constructions for authenticity.
- Defenses: HMAC; prefix-free encodings; domain separation; MACs from block ciphers (e.g., CMAC).

Thuật ngữ: đệm; tiền tố; phân tách miền; CMAC

Why "MAC = H(K ‖ M)" Fails

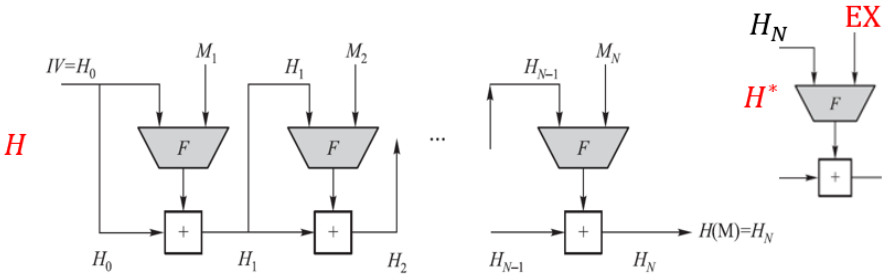
- Attacker with $\tau = H(K\|M)$ can compute $\tau' = H(\underbrace{K\|M\|\text{pad}(K\|M)}_{\text{internal state known}}\|M')$ for chosen suffix M'.
- Leads to forgeries if protocol authenticates $\langle M, \tau \rangle$ without contextual checks.

Fixes:

1. **HMAC:** $\text{HMAC}_H(K, M) = H((K \oplus \text{opad}) \| H((K \oplus \text{ipad}) \| M))$.
2. Prefix-free encodings: include *length of K* or *type/domain tag* inside the inner hash input.

Thuật ngữ: MAC khóa tiền tố; giả mạo; HMAC; mã hoá miền

Length-Extension attacks on SHA2 Family



$$\mathrm{H}(K||M) = \mathrm{H}(K||M||padded)$$

$$\Rightarrow \mathrm{H}(K||M||padded||\mathrm{EX}) = H^*(EX)$$

can compute $\mathrm{H}(K|M||padded||\mathrm{EX})$ **without knowing the input** K

Figure: Length-Extensio on SHA2 Family

HMAC Security Sketch & Parameters

- Security reduces to PRF/PRP properties of H's compression under standard assumptions (tight bounds depend on model).
- Truncation: publishing only the first t bits of HMAC (e.g., t=128) is acceptable with security $\approx 2^t$.
- Key management: normalize K to block size (hash if longer; zero-pad if shorter) before mixing with ipad/opad.

Thuật ngữ: cắt ngắn thẻ; khối băm; trộn khóa với ipad/opad

SHA-3 / Keccak: Sponge Construction

- State size b=1600 bits; split into *rate* r and *capacity* c ($b = r + c$).
- **Absorb:** XOR message blocks of size r into state, apply permutation f.
- **Squeeze:** Output r bits per round; for XOFs, continue permuting for more output.
- Security $\approx \min(2^{c/2}$ (collision), 2^c (preimage)).

Thuật ngữ: tốc độ (rate); dung lượng (capacity); hoán vị; XOF

SHA-3 / Keccak: Sponge Construction

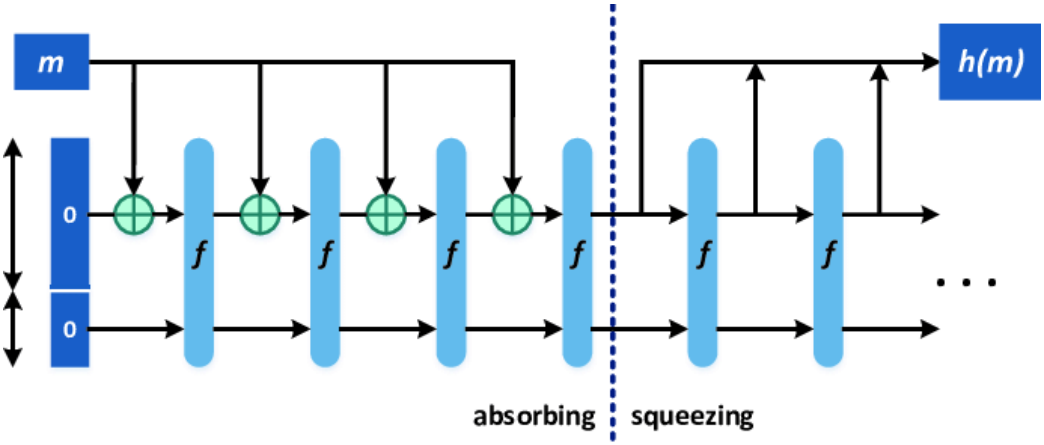


Figure: Sponge Construction

Keccak-f[1600] Permutation (24 rounds)

- Round steps: $\theta \to \rho \to \pi \to \chi \to \iota$ on a $5 \times 5 \times 64$ lane array.
- SHA3-256: r=1088, c=512; SHA3-512: r=576, c=1024.
- XOFs: SHAKE128/256; domain-separated variants: cSHAKE, KMAC (NIST SP 800-185).

Thuật ngữ: các bước vòng $\theta, \rho, \pi, \chi, \iota$; phân tách miền; họ SHAKE/KMAC

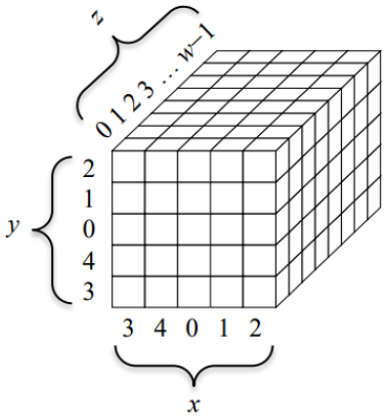


Figure: Keccak-f[1600] round structure

SHA-3: Domain Separation & Padding (FIPS 202)

- **Suffix bits before padding (domain separation):**
  - Hashes: SHA3-224/256/384/512$(M) = \mathrm{KECCAK}[c]\big(M \,\|\, 01, d\big)$.
  - XOFs: SHAKE128/256$(M, d) = \mathrm{KECCAK}[c]\big(M \,\|\, 1111, d\big)$.
  - RawSHAKE (alt. def.): $\mathrm{KECCAK}[c](M \,\|\, 11, d)$.
- **Multi-rate padding** pad10*1 with $j = (-m - 2) \bmod r$: append $1 \,\|\, 0^j \,\|\, 1$ so that total is a multiple of the rate r.
- **Capacity–security rule:** $c = 2d$ for hashes; collision $\approx 2^{c/2}$, (second-)preimage $\approx 2^c$ (generic bounds).

Thuật ngữ: phân tách miền; hậu tố (suffix) 01, 1111; phần đệm đa tốc pad10*1

SHA-3 State Layout & Permutation Steps

State size b=1600 bits as a 5×5 array of w=64-bit lanes $A[x, y]$.

1. $\boldsymbol{\theta}$: column parity diffusion; $C[x] = \bigoplus_y A[x, y]$, $D[x] = C[x-1] \oplus \mathrm{ROTL}_1(C[x+1])$, $A[x, y] \oplus = D[x]$.
2. $\boldsymbol{\rho}$: intra-lane rotations by fixed offsets.
3. $\boldsymbol{\pi}$: lane permutation $(x, y) \mapsto (y, 2x+3y) \pmod 5$.
4. $\boldsymbol{\chi}$: non-linear step $A[x, y] \leftarrow A[x, y] \oplus ((\neg A[x+1, y]) \wedge A[x+2, y])$.
5. $\boldsymbol{\iota}$: add round constant to $A[0, 0]$.

Thuật ngữ: ma trận 5x5 lane 64-bit; các bước vòng: $\theta, \rho, \pi, \chi, \iota$

SHA-3 Security Strengths (per FIPS 202)

For hashes, $c{=}2d$; for XOFs, strength depends on requested output d.

- SHA3-256: collision 2^{128}, (preimage, 2nd-preimage) 2^{256}.
- SHA3-512: collision 2^{256}, (preimage, 2nd-preimage) 2^{512}.
- SHAKE128: collision $\min(2^{d/2}, 2^{128})$; preimage $\geq \min(2^{d}, 2^{128})$.
- SHAKE256: collision $\min(2^{d/2}, 2^{256})$; preimage $\geq \min(2^{d}, 2^{256})$.

Thuật ngữ: độ mạnh bảo mật: va chạm, tiền ảnh, ảnh thứ hai; XOF phụ thuộc độ dài đầu ra d

SHA-3 vs Keccak (competition submission)

- NIST SHA-3 adds *domain separation suffix* (`01` for hashes; `1111` for SHAKE) on top of Keccak and fixes parameters.
- Same core permutation Keccak-p[1600, 24] and multi-rate padding `pad10*1`.
- Practical note: SHA-3 hash functions are *fixed-output*; SHAKE are XOFs suitable for KDFs, masks, commitments.

Thuật ngữ: khác biệt tiêu chuẩn: hậu tố miền; Keccak-p[1600,24]; XOF dùng cho KDF

Keyed Use: HMAC with SHA-3 & KMAC (SP 800-185)

- **HMAC-SHA3**: same construction as HMAC-SHA2; block sizes B (bytes): 144 (224), 136 (256), 104 (384), 72 (512). Key pre-process to B.
- **KMAC128/256**: PRF/MAC over Keccak with customization strings; variable-length tag; avoids length-extension; native to sponge.
- **cSHAKE**: customizable SHAKE; N-name and S-customization provide robust domain separation.
- Use KMAC for new designs; HMAC-SHA3 for compatibility where HMAC interface is mandated.

Thuật ngữ: KMAC: MAC dựa trên Keccak; cSHAKE: SHAKE tuỳ biến; chuỗi tuỳ biến để tách miền

Sponge API Patterns (Absorb/Squeeze/Finalize)

```
SpongeInit(state);
for each r-bit block M_i: // absorb
  state[0..r-1] ^= M_i;
  state = KeccakF1600(state); // 24 rounds
// squeeze
out = empty;
while |out| < d:
  out += state[0..r-1];
  if |out| < d: state = KeccakF1600(state);
return Truncate(out, d);
```

- Duplex mode enables AEAD/KDF streams by alternating absorb/squeeze.

Thuật ngữ: API bọt biển; chế độ duplex cho AEAD/KDF

Bit/Byte Ordering Pitfalls

- Lanes are 64-bit little-endian; byte-to-bit mapping differs from MD hashes; test with known vectors ("abc").
- Rotation offsets and index mapping must match spec; ensure portable ROTL/ROTR on 64-bit words.
- Beware of padding: suffix bits are appended *before* `pad10*1`.

Thuật ngữ: thứ tự bit/byte; little-endian theo lane; xoay vòng; chèn hậu tố trước khi đệm

Design Guidance (when to choose what)

- **Hashing:** SHA-256 (ubiquity) or SHA3-256 (sponge diversity, domain separation with SHAKE ecosystem).
- **MAC:** KMAC256 (new designs) or HMAC-SHA-256/SHA3-256 (compatibility).
- **KDF/XOF:** SHAKE128/256 or cSHAKE for app-specific separation.
- **Parallel hashing:** ParallelHash/Tree modes (SP 800-185) or Sakura-tagged constructions.

Thuật ngữ: lựa chọn thuật toán theo mục đích; tách miền; băm song song

Hands-on: Test Vectors

- SHA3-256("abc") = 3a985da74fe225b2045c172d6bd390bd855f086e3e9d525b46bfe24511431532
- SHAKE128("abc", 256 bits) = 5881092dd818bf5cf8a3ddb793fbcba7...

Use these to sanity-check your implementation and endianness.

Thuật ngữ: chuẩn kiểm thử; kiểm chứng triển khai

Engineering Guidance & Pitfalls

- Prefer HMAC-SHA-256 or HMAC-SHA-512 for MACs; SHA-256/512 for *unkeyed* digests.
- Do *not* use raw SHA for passwords; use Argon2id/scrypt with salt & memory hardness.
- Be wary of length-extension in protocols; authenticate *exactly* what is parsed/used (AD of AEAD if possible).
- Constant-time implementations; avoid secret-dependent branches/table indices.
- Domain separation: tag different contexts (e.g., “KDF”, “MAC”, “TREE”) to avoid cross-protocol collisions.

Thuật ngữ: Argon2id; độ cứng bộ nhớ; thời gian hằng; tách miền

Tree Hashing & Parallelism (brief)

- Merkle trees enable parallel hashing of large inputs at fixed depth; many modern hashes offer tree/XOF modes.
- Use domain tags for leaf/internal/root nodes to preserve collision resistance.

Thuật ngữ: cây Merkle; băm song song; thẻ miền

Hands-on Checks (for class)

1. Verify ISO images: compute SHA-256 locally; compare to vendor-supplied digest.
2. Demonstrate length-extension with a lab tool; then show HMAC preventing it.
3. Implement SHA-256 round functions and test vectors; compare against reference.

Thuật ngữ: bài thực hành; so khớp băm; công cụ minh hoạ

Summary

- SHA-2 (MD-based) and SHA-3 (sponge) differ structurally but both are robust when used correctly.
- Length-extension affects MD-based hashes; HMAC neutralizes it by re-keying the outer hash.
- Engineering choices: HMAC for MAC; Argon2id for passwords; strong domain separation across contexts.

Thuật ngữ: tóm tắt; lựa chọn kỹ thuật; dùng đúng ngữ cảnh

References (selected)

- NIST FIPS 180-4: *Secure Hash Standard* (SHA-1, SHA-224/256/384/512).
- NIST FIPS 202: *SHA-3 Standard: Permutation-Based Hash and XOFs*.
- NIST SP 800-185: cSHAKE, KMAC, TupleHash, ParallelHash.
- Bellare, Canetti, Krawczyk (1996): HMAC: Keyed-Hashing for Message Authentication.