Chuẩn an toàn thông tin là : ISO 27001

Để nâng cao việc phát triển các giải pháp an toàn cho một hệ thống CNTT, người ta tập trung đầu tư vào 3 vấn đề chính là : Con người, quy trình, công nghệ

| **Mục đích** | **Khóa dùng để mã hóa/ký** | **Khóa dùng để giải mã/xác minh** |
| --- | --- | --- |
| **Mã hóa bảo mật (Confidentiality)** | ✅ **Public key của người nhận** | ✅ **Private key của người nhận** |
| **Chữ ký số (Authenticity)** | ✅ **Private key của người gửi** | ✅ **Public key của người gửi** |

Charlie mã hóa thông điệp bằng private key của mình và gửi cho David. Ai có thể đọc được?

✅ **Trả lời:**  
Bất kỳ ai có **public key của Charlie** đều có thể giải mã.  
→ Không phải để bảo mật, mà để **chứng minh thông điệp đến từ Charlie** (xác thực, không chối bỏ).

Giải pháp **Stackshield** giúp phòng chống tấn công tràn bộ đệm trên stack thực hiện như sau: Lưu trữ giá trị Return Address ở một nơi khác và sử dụng nó để kiểm tra xem giá trị ở Return Address có bị sửa đổi hay không

Giải pháp **StackGuard** giúp phòng chống tấn công tràn bộ đệm trên stack thực hiện như sau: Sử dụng một vùng nhớ đệm an toàn giữa Return Address và Buffer. Sử dụng vùng nhớ đệm an toàn này để kiểm tra xem Return Address có bị sửa đổi hay không

Với S-box :

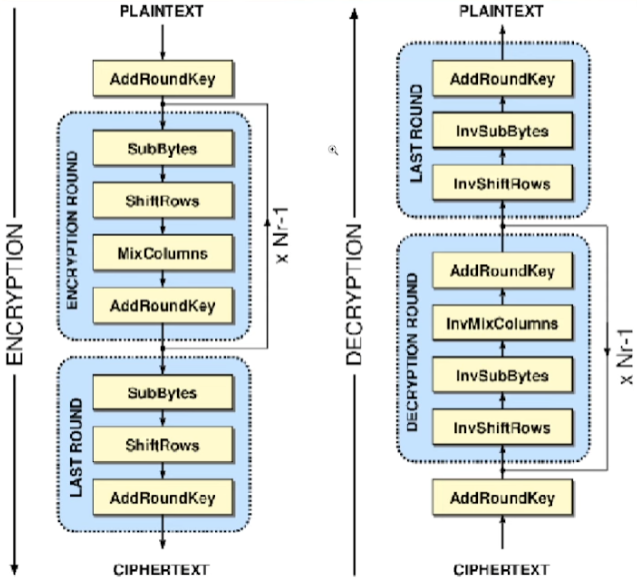
Với S-box nhận 6 bit đầu vào và trả về 4 bit đầu ra (như trong DES):

* **Hai bit ngoài cùng** (bit đầu và bit cuối) xác định **hàng** trong bảng S-box.
* **Bốn bit ở giữa** xác định **cột** trong bảng S-box.

Ví dụ :

Hai bit ngoài cùng: 1 (bit đầu) và 1 (bit cuối) → 11 (nhị phân) = 3 (thập phân) → Hàng 3.

Bốn bit ở giữa: 0010 (nhị phân) = 2 (thập phân) → Cột 2.



Bài tập về mô hình MAC : Bell – Lapadula

* **Mức độ của nhãn** được sắp xếp: Unclassified < Confidential < Secret < Top Secret.
* **Simple Security Property (đọc)**: Chủ thể (L\_s, C\_s) có thể đọc đối tượng (L\_o, C\_o) nếu:
  + **L\_s ≥ L\_o** (mức độ nhạy cảm của chủ thể cao hơn hoặc bằng đối tượng).
  + **C\_s ⊇ C\_o** (danh mục của chủ thể bao gồm tất cả danh mục của đối tượng).
* **Star Property (ghi)**: Chủ thể (L\_s, C\_s) có thể ghi vào đối tượng (L\_o, C\_o) nếu:
  + **L\_o ≥ L\_s** (mức độ nhạy cảm của đối tượng cao hơn hoặc bằng chủ thể).
  + **C\_o ⊇ C\_s** (danh mục của đối tượng bao gồm tất cả danh mục của chủ thể).

**Ví dụ : Alice đọc File1**

* **Alice**: (Top Secret, {Bộ binh, không quân})
* **File1**: (Secret, {bộ binh })
* **Kiểm tra Simple Security Property**:
  + **Mức độ nhạy cảm**: Top Secret > Secret → Đáp ứng.
  + **Danh mục**: {bộ binh, không quân } ⊇ {bộ binh} → Đáp ứng.
* **Kết luận**: Alice **được phép đọc** File1.

## 1. Làm cứng (Hardening) hệ điều hành

* **Tắt dịch vụ không cần thiết**: Ví dụ, tắt Telnet hoặc FTP nếu không dùng.
* **Cấu hình quyền**: Chỉ cấp quyền tối thiểu (Least Privilege).
* **Thêm công cụ bảo mật**: Antivirus, tường lửa, IDS.

## 2. Đánh giá rủi ro (Risk Assessment)

Risk = Xác suất × Mức ảnh hưởng

**Risk > 1**

* Chỉ xử lý nếu **nguy cơ đủ lớn**.

Rủi ro chính từ việc sử dụng phần mềm lỗi thời (outdated software) là : Nó có thể không còn được hỗ trợ bởi các nhà cung cấp

## 3. Mã hoá đối xứng và bất đối xứng. Ví dụ + bài tập

**a) Mã hoá đối xứng :**

Là phương pháp mã hóa sử dụng **cùng một khóa (key)** cho cả quá trình mã hóa và giải mã.

| **Đặc điểm** | **Mô tả** |
| --- | --- |
| Khóa duy nhất | Cần **chia sẻ khóa bí mật** giữa người gửi và người nhận |
| Nhanh | Phù hợp **mã hóa lượng lớn dữ liệu** |
| Độ an toàn | Rất an toàn nếu khóa đủ mạnh và được bảo vệ tốt |

Ví dụ :

**🔹 Alice muốn gửi tài liệu mật cho Bob:**

1. Alice và Bob **thoả thuận trước một khóa** bí mật: "KEY123"
2. Alice dùng "KEY123" để mã hóa file PDF → ciphertext
3. Bob nhận được ciphertext và dùng "KEY123" để giải mã

📌 Nếu hacker chặn được dữ liệu nhưng **không có khóa**, họ không giải mã được.

**Các thuật toán tiêu biểu :**

| **Thuật toán** | **Khối** | **Tình trạng** |
| --- | --- | --- |
| **AES** | 128-bit block | 🔥 Chuẩn mạnh nhất hiện nay |
| **DES** | 64-bit block | ❌ Đã lỗi thời |
| **3DES** | 64-bit block | ⏳ Vẫn dùng, nhưng chậm |
| **RC4** | Dòng | ❌ Không còn khuyến nghị |

**b) Mã hoá bất đối xứng**

Là kỹ thuật mã hóa dùng **hai khóa khác nhau**:

* **Public key (khóa công khai)**: để mã hóa
* **Private key (khóa bí mật)**: để giải mã

📌 Ai cũng **nhìn thấy public key**, nhưng chỉ người giữ private key mới giải mã được!

Ví dụ :

**Alice muốn gửi thông tin bí mật cho Bob:**

1. Bob công khai **public key** của mình cho mọi người
2. Alice dùng **public key của Bob** để mã hóa nội dung
3. Chỉ có Bob – với **private key** – mới có thể giải mã được nội dung

📌 Nếu hacker bắt được ciphertext → **vô dụng vì không có private key**

**Ứng dụng đặc biệt – Chữ ký số**

* Người gửi dùng **private key để ký dữ liệu**
* Bên nhận dùng **public key của người gửi** để kiểm tra chữ ký

✅ Đảm bảo:

* Tính **xác thực** (người gửi thật sự là ai)
* Tính **toàn vẹn** (dữ liệu không bị sửa đổi)

| **Thuật toán** | **Dùng cho** | **Độ dài khóa** |
| --- | --- | --- |
| **RSA** | Mã hóa, chữ ký | 2048–4096 bit |
| **ECC** | Chữ ký, trao đổi khóa | 256–521 bit |
| **ElGamal** | Mã hóa | Cao |
| **Diffie-Hellman** | Trao đổi khóa | Không dùng để mã hóa trực tiếp |

Tấn công vào **mã hóa đối xứng**

| **Kỹ thuật** | **Mô tả** |
| --- | --- |
| **Brute-force** | Thử tất cả khóa → nếu key ngắn sẽ dễ bị dò |
| **Differential Cryptanalysis** | Phân tích sự thay đổi đầu vào & đầu ra |
| **Linear Cryptanalysis** | Dùng phương trình tuyến tính gần đúng để suy key |
| **Statistical Attacks** | Dò mật mã dựa vào tần suất xuất hiện |
| **Key Reuse** | Dùng lại key → làm lộ mối liên hệ P ⇔ C |

RSA: Dựa trên khó khăn của việc phân tích một số lớn thành hai số nguyên tố

Diffie-Hellman: Dựa trên bài toán logarit rời rạc, tức là khó tìm số mũ trong một phép lũy thừa modulo

ElGamal: Cũng dựa trên logarit rời rạc, nhưng có thể dùng cho cả mã hóa và chữ ký số

ECC: Dựa trên bài toán logarit rời rạc trên đường cong elliptic.

## 4. Mô hình bộ nhớ chương trình

A diagram of a software structure

AI-generated content may be incorrect.

Các vùng trong bộ nhớ:

| **Vùng** | **Chức năng** |
| --- | --- |
| **Text** | Mã thực thi |
| **Data segment** | Biến toàn cục đã khởi tạo |
| **BSS segment** | Biến toàn cục chưa khởi tạo |
| **Heap** | Cấp phát động (malloc, new) |
| **Stack** | Lưu biến cục bộ, return address |

Cụ thể :

**Data segment** : Lưu các biến static/global đã được khởi tạo trong chương trình

## 5. ACL ( access control list ) và CL ( Capability list ) + bài tập

So sánh :

| **Tiêu chí** | **ACL (Access Control List)** | **CL (Capability List)** |
| --- | --- | --- |
| Gắn với | **Đối tượng (Object)** | **Chủ thể (Subject)** |
| Dạng câu hỏi | “Ai có thể truy cập cái này?” | “Người này có thể truy cập gì?” |
| Phổ biến ở | Hệ điều hành, filesystem | Mạng phân tán, OS nghiên cứu |
| Kiểm tra quyền | Truy cập tài nguyên → xem ACL | Người dùng → xem capability |
| Bảo mật | Có thể bị tấn công nếu không kiểm tra kỹ | Phải bảo vệ capability token |

**Thêm quyền cho user:**

setfacl -m u:alice:rw file.txt

* Cho user alice quyền đọc (r) và ghi (w) vào file.txt

**Thêm quyền cho group:**

setfacl -m g:students:r file.txt

**Xóa quyền:**

setfacl -x u:alice file.txt

## 6. Ma trận điều khiển truy cập

**Ma trận điều khiển truy cập (Access Control Matrix)** là một **mô hình lý thuyết** dùng để **xác định quyền truy cập** của người dùng (subject) đối với tài nguyên (object).

* Hàng: đại diện cho **subject** (user, process)
* Cột: đại diện cho **object** (file, database, printer,...)
* Mỗi ô chứa **quyền truy cập** của subject với object đó

Các thành phần gồm : **Subject, Object** và **Acess Right ( AC )**

Mô hình này **lý tưởng nhưng không thực tế để lưu trữ** → hệ thống thực tế triển khai thành 2 dạng chính:

* **Access control lists :** Gắn danh sách quyền theo từng object

🡪 Dễ cho kiểm soát theo **đối tượng**

* **Capability lists :** Gắn danh sách quyền theo từng subject

🡪 Dễ cho quản lý theo **người dùng**

Từ ma trận điều khiển truy cập, ta có thể suy ra các thông tin nào :

* **Access control lists**
* **Capability lists**

Liên hệ với mô hình phân quyền :

| **Mô hình** | **Triển khai dựa trên ACM?** | **Ghi chú** |
| --- | --- | --- |
| **DAC** | ✅ | Chủ sở hữu có quyền sửa dòng trong ma trận |
| **MAC** | ✅ | Hệ thống kiểm tra nhãn bảo mật khi tra cứu ma trận |
| **RBAC** | ✅ | Role là “Subject”, Object là “Tài nguyên”, quyền xác định qua vai trò |
| **ABAC** | ❌ (mở rộng ACM) | Quyền không cố định mà **tính toán tại thời điểm truy cập** dựa trên thuộc tính |

## 7. Phân loại tấn công( passvive và active )+ liệt kê + đặc điểm + ví dụ + xảy ra ở tầng nào

| **Loại tấn công** | **Mục tiêu** | **Đặc điểm chính** |
| --- | --- | --- |
| **Passive** (Thụ động) | **Nghe lén, quan sát** thông tin | Không thay đổi dữ liệu, khó phát hiện |
| **Active** (Chủ động) | **Can thiệp, thay đổi** hoặc phá hoại dữ liệu | Làm thay đổi nội dung, dễ phát hiện |

**Tấn công thụ động ( Passive )**

| **Hình thức** | **Mô tả** | **Ví dụ thực tế** | **Tầng OSI** |
| --- | --- | --- | --- |
| Eavesdropping | Nghe lén dữ liệu đang truyền | Dùng Wireshark nghe trộm gói tin Wi-Fi | Tầng 1–3 |
| Traffic analysis | Phân tích số lượng và thời gian gửi dữ liệu (dù không đọc được nội dung) | Phân tích luồng truy cập đến website chính phủ | Tầng 3–4 |
| Sniffing | Bắt gói tin trên mạng LAN | ARP spoof + Wireshark để lấy mật khẩu | Tầng 2–3 |

**Biện pháp phòng chống:**

* **Mã hóa dữ liệu** (HTTPS, TLS)
* Dùng mạng ảo riêng (**VPN**)
* Tách VLAN, bật **port security**

**Tấn công chủ động ( Active )**

**Mục tiêu:**

* **Thay đổi, chặn, giả mạo hoặc phá hoại** dữ liệu
* Ảnh hưởng trực tiếp đến tính **toàn vẹn, sẵn sàng**

| **Hình thức** | **Mô tả** | **Ví dụ thực tế** | **Tầng OSI** |
| --- | --- | --- | --- |
| **Masquerade** (Giả mạo) | Hacker giả danh người khác | Hacker giả email của sếp → lừa chuyển tiền | Tầng 4–7 |
| **Replay** | Ghi lại gói tin → gửi lại sau để lừa hệ thống | Gửi lại lệnh chuyển tiền đã ghi trước đó | Tầng 3–7 |
| **Modification** | Thay đổi nội dung gói tin | Sửa điểm số sinh viên trong hệ thống | Tầng 5–7 |
| **Denial of Service (DoS/DDoS)** | Làm nghẽn, tê liệt hệ thống | Gửi hàng triệu truy cập giả vào website | Tầng 3–4 |
| **Man-in-the-Middle (MITM)** | Hacker chen giữa 2 bên, đọc và sửa dữ liệu | Giả làm Wi-Fi công cộng để nghe lén và can thiệp | Tầng 2–4 |

## 8. Hàm băm, MAC, HMAC và Chữ ký số (Digital Signature) ( khái quát )

### a) Hàm Băm ( hash function )

**Mục đích:**

* Tạo **dấu vân tay số** cho dữ liệu → dùng để **kiểm tra toàn vẹn**
* Không thể dùng để giải ngược lại → đây là **hàm một chiều**

**Đặc điểm kỹ thuật:**

* Đầu vào: dữ liệu bất kỳ
* Đầu ra: chuỗi bit cố định (128, 160, 256, 512 bit…)
* Thuộc tính an toàn:
  1. **Pre-image resistance** (khó tìm input từ hash)
  2. **Second pre-image resistance** (khó tìm input thứ 2 có cùng hash)
  3. **Collision resistance** (khó tìm 2 input khác nhau cho ra cùng hash)

**Thuật toán hàm Băm**

| **Thuật toán** | **Độ dài** | **Ghi chú** |
| --- | --- | --- |
| **MD5** | 128 bit | Cũ, dễ bị tấn công, không còn an toàn |
| **SHA-1** | 160 bit | Bị collision bởi Google (dự án SHAttered) |
| **SHA-2** | 256/384/512 bit | Hiện vẫn an toàn và dùng phổ biến |
| **SHA-3** | Tùy chọn | Mới, dùng kiến trúc sponge (Khác SHA-2) |
| **RIPEMD** | 160 bit | Tùy chọn thay thế SHA |

### b) MAC – MESSAGE AUTHENTICATION CODE

**Mục đích:**

* **Xác minh tính toàn vẹn** và **người gửi hợp lệ**
* Dữ liệu không bị sửa trong quá trình truyền

**Thuật toán MAC**

| **Thuật toán** | **Ghi chú** |
| --- | --- |
| **CBC-MAC** | Dùng thuật toán mã hóa khối (AES, DES) với chế độ CBC để tạo MAC |
| **CMAC** | Phiên bản cải tiến của CBC-MAC, an toàn hơn |
| **UMAC, VMAC** | MAC hiệu suất cao, dùng trong môi trường mạng tốc độ cao |

### c) HMAC – HASH-BASED MAC

**Mục đích:** Kết hợp giữa **hash function** và **khóa bí mật** để tạo MAC

**Ưu điểm:**

* Chống lại tấn công mở rộng chiều dài (length extension attack)
* Dễ triển khai, dùng rộng rãi

**Các thuật toán của HMAC**

| **Thuật toán** | **Ghi chú** |
| --- | --- |
| **HMAC-SHA1** | Kết hợp SHA-1 (ít dùng vì yếu) |
| **HMAC-SHA256** | Phổ biến trong web/API |
| **HMAC-SHA512** | Bảo mật mạnh hơn, chậm hơn |
| **HMAC-MD5** | Không còn an toàn, nhưng đôi khi vẫn dùng trong hệ cũ |

### d) DIGITAL SIGNATURE – CHỮ KÝ SỐ

**Mục đích:** Xác nhận **ai gửi**, dữ liệu có **bị thay đổi không**, và **người gửi không thể chối bỏ**

**Hoạt động :**

| **Bên gửi (ký)** | **Bên nhận (xác minh)** |
| --- | --- |
| Tính hash của thông điệp | Nhận thông điệp và chữ ký |
| Dùng private key để mã hóa hash → thành chữ ký số | Dùng public key giải mã chữ ký |
| Gửi dữ liệu + chữ ký | Tự tính lại hash, so sánh |

**Các thuật toán chữ ký số**

| **Thuật toán** | **Dựa trên** | **Ghi chú** |
| --- | --- | --- |
| RSA Signature | RSA | Cổ điển, dễ hiểu, dùng nhiều |
| DSA (Digital Signature Algorithm) | Logarit rời rạc | Chuẩn FIPS 186 (Mỹ) |
| ECDSA (Elliptic Curve DSA) | Elliptic Curve | Mạnh, khóa nhỏ, nhanh |
| EdDSA (Ed25519) | Elliptic Curve | Rất nhanh, hiện đại |

e) So sánh tổng quát

| **Phương thức** | **Có mã hóa?** | **Dùng khóa gì?** | **Mục tiêu chính** | **Thuật toán phổ biến** |
| --- | --- | --- | --- | --- |
| **Hash** | ❌ Không | Không cần | Toàn vẹn | MD5, SHA-1, SHA-2 |
| **MAC** | ❌ Không | Khóa đối xứng | Toàn vẹn + xác thực | CBC-MAC, CMAC |
| **HMAC** | ❌ Không | Khóa đối xứng | An toàn hơn MAC | HMAC-SHA256 |
| **Chữ ký số** | ✅ Có | Khóa bất đối xứng | Toàn vẹn + xác thực + chống chối bỏ | RSA, DSA, ECDSA, EdDSA |

## 9. Mã hoá theo khối ( Block Cipher ) và mã hoá theo dòng ( Stream Cipher )

**a) Block cipher :**

Là kỹ thuật chia dữ liệu thành **các khối có độ dài cố định** (ví dụ: 64 bit, 128 bit), và **mỗi khối được mã hóa riêng biệt** bằng thuật toán mã hóa khối.

Đặc điểm :

* Làm việc trên khối (block) – thường là 64 hoặc 128 bit
* Dùng một khóa bí mật và thực hiện nhiều **vòng biến đổi**
* Các thuật toán: **DES, 3DES, AES**

Cách hoạt động của AES ( advanced …. )

* **Chia dữ liệu đầu vào thành các khối 128 bit**
* Mỗi khối được đưa vào thuật toán mã hóa
* Áp dụng **nhiều vòng biến đổi (rounds)** như:
  + **SubBytes** – Thay byte bằng S-box
  + **ShiftRows** – Xoay dòng
  + **MixColumns** – Trộn cột
  + **AddRoundKey** – XOR với khóa
* Kết quả là một khối mã hóa 128 bit

Tính an toàn nằm ở:

* **Đa tầng** (multiple rounds)
* **Ẩn mối quan hệ giữa dữ liệu – khóa – kết quả**

**b) Stream cipher**

Là kỹ thuật tạo ra một **luồng khóa giả (keystream)** rồi **XOR từng bit hoặc byte với dữ liệu gốc** để tạo ciphertext.

Đặc điểm :

* Mã hóa **theo từng bit hoặc byte liên tiếp**, không chia khối
* Phù hợp dữ liệu thời gian thực (âm thanh, video, IoT)
* Nhanh hơn block cipher
* Nhạy cảm với **việc tái sử dụng khóa** (key reuse → thảm họa)

Cách hoạt động ( RC4 ) :

* Sinh ra luồng khóa (keystream) từ seed và khóa ban đầu
* XOR từng byte của plaintext với keystream
* Kết quả là ciphertext

**c) So sánh**

Mã hoá theo khối và mã hoá theo dòng là 2 dạng của mã hoá đối xứng :

| **Tiêu chí** | **Block Cipher** | **Stream Cipher** |
| --- | --- | --- |
| Đơn vị xử lý | Khối cố định (ex: 64, 128 bit) | Bit hoặc byte đơn lẻ |
| Ví dụ | DES, AES | RC4, A5/1 |
| Tốc độ | Chậm hơn (do cần xử lý cả khối) | Nhanh hơn (bit-by-bit) |
| Cơ chế | Dựa vào S-box, P-box, các vòng lặp | XOR với key stream |
| Độ an toàn | Cao hơn (nếu dùng đúng chế độ) | Dễ bị tấn công nếu tái sử dụng key |

**d) Chế độ**

Các chế độ ( mode ) của block cipher :

| **Chế độ** | **Đặc điểm chính** | **Ưu/Nhược** |
| --- | --- | --- |
| **ECB** | Mỗi block mã hóa độc lập | ❌ Lộ mẫu dữ liệu |
| **CBC** | Block sau phụ thuộc block trước + IV | ✅ Bảo mật cao, ❌ không song song |
| **CFB** | XOR từng bit hoặc byte với output trước đó | Dùng như stream cipher |
| **OFB** | Giống CFB nhưng input không phụ thuộc cipher | Không lan truyền lỗi |
| **CTR** | Dùng counter thay vì đầu ra trước | ✅ Nhanh, song song, được dùng nhiều |

* **ECB - Electronic Codebook**
* **CBC – Cipher Block Chaining**
* **CFB – Cipher Feedback**
* **OFB – Output Feedback**
* **CTR – Counter Mode**

**e) Các thuật toán phổ biến**

**DES – Data Encryption Standard**

* 64-bit block, 56-bit key
* Dựa trên Feistel Network, có 16 vòng
* Dùng S-box, P-box, permutation, shift

**3DES – Triple DES**

* Mã hóa 3 lần: E-D-E
* Key dài: 112 hoặc 168 bit
* Bảo mật cao hơn DES nhưng rất chậm

**AES – Advanced Encryption Standard**

* Block 128-bit
* Key: 128, 192, 256 bit
* Dùng phép biến đổi: SubBytes, ShiftRows, MixColumns, AddRoundKey
* Không dựa trên Feistel
* Hiện là chuẩn mã hóa khối mạnh nhất và phổ biến nhất

**Các loại tấn công vào mã hóa đối xứng**

| **Tấn công** | **Giải thích** |
| --- | --- |
| Brute-force | Thử mọi key (DES dễ dính) |
| Differential | Phân tích thay đổi đầu vào → đầu ra |
| Linear | Dò quan hệ tuyến tính giữa input/output |
| Key Reuse | Lặp keystream hoặc IV → dễ dò key |
| Side-channel | Dựa vào thời gian, điện năng tiêu thụ để đoán key |

## 10. Thuật toán RSA ( chi tiết về cách hoạt động của RSA + bài toán )

Ví dụ bài toán : Cho hai số nguyên tố p=13, q=19, giá trị e nào sẽ được chọn trong thuật toán mã hóa RSA từ số các giá trị sau:

Đáp án là 47

**Cách giải :**

Để xây dựng khóa RSA, ta thực hiện các bước:

1. **Chọn hai số nguyên tố khác nhau**: p, q
2. Tính n = p × q (modulus)
3. Tính **φ(n) = (p – 1)(q – 1)** (Euler's totient)
4. **Chọn e**: 1 < e < φ(n), **thỏa mãn** gcd( e, φ(n) ) = 1
5. Tính **d**: là **nghịch đảo modular** của e theo mod φ(n) sao cho :

d × e ≡ 1 mod φ(n)  
🡪 d ≡ e⁻¹ mod φ(n)

1. Cặp khóa:
   * Public key: (e, n)
   * Private key: (d, n)

**Mã hóa và giải mã:**

* Mã hóa: C = M^e mod n
* Giải mã: M = C^d mod n

**1. RSA là gì?**

***RSA (Rivest–Shamir–Adleman)*** *là một thuật toán* ***mã hóa bất đối xứng****, được đặt tên theo ba nhà khoa học phát minh ra nó vào năm 1977.*

**2. Dựa trên nguyên lý toán học :**

*RSA dựa trên* ***độ khó của bài toán phân tích một số nguyên lớn thành tích của hai số nguyên tố****.*

**📌 Cụ thể:**

* Bạn chọn hai số nguyên tố lớn: p, q
* Tính n = p × q
* Public key gồm (n, e)
* Private key được tính dựa trên p, q, e

➡ Kẻ tấn công biết n, nhưng **không biết p và q**, nên **không thể tính private key** nếu không **phân tích được n thành thừa số nguyên tố**.

**🧨 Tại sao điều này an toàn?**

* Với số n có 2048 bit (khoảng 617 chữ số), **phân tích thừa số là bất khả thi bằng các thuật toán hiện tại**
* Không có **thuật toán khả thi trong thời gian đa thức (polynomial time)** để phân tích các số nguyên lớn nếu không có yếu tố lượng tử

**Đối với các thuật toán khác thì sao ?**

| **Thuật toán** | **Cơ sở toán học** |
| --- | --- |
| **RSA** | Phân tích số thành thừa số nguyên tố |
| **ElGamal, DSA** | Logarit rời rạc (Discrete Log Problem) |
| **ECC** | Logarit rời rạc trên đường cong elliptic |
| **Diffie-Hellman** | Logarit rời rạc (cho trao đổi khóa) |

## 11. Các dạng mã độc + các đặc điểm + so sánh

Hai dạng mã độc sau đây sống độc lập : Worm, Zombie

Metamorphic virus có đặc điểm : Lẩn tránh phát hiện thông qua việc ghi đè lên chính nó

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Tiêu chí** | **Virus** | **Worm** | **Trojan** | **Ransomware** | **Rootkit** | **Spyware** | | --- | --- | --- | --- | --- | --- | --- | | Tự lây lan | ❌ | ✅ | ❌ | ❌ | ❌ | ❌ | | Cần hành động người dùng | ✅ | ❌ | ✅ | ✅ | ✅ | ✅ | | Có thể phá hủy hệ thống | ✅ | ✅ | ⚠️ | ✅ | ✅ | ❌ | | Gây mã hóa – đòi tiền | ❌ | ❌ | ❌ | ✅ | ❌ | ❌ | | Mục tiêu gián điệp | ❌ | ❌ | ✅ | ❌ | ✅ | ✅ | | Khó phát hiện | ❌ | ❌ | ⚠️ | ✅ | ✅ | ✅ | |  |  |

## 12. Tấn công mạng + các dạng + đặc điểm + cách hoạt động + hậu quả/kết quả

| **Dạng tấn công** | **Cách hoạt động chính** | **Hậu quả phổ biến** |
| --- | --- | --- |
| **Passive** (Sniffing, MITM) | Nghe trộm, theo dõi | Lộ thông tin |
| **Active** (DDoS, Phishing, Malware) | Phá hoại, giả mạo | Mất dữ liệu, tài chính |
| **Mã độc** (Virus, Ransomware…) | Gắn vào file/hệ thống | Mã hóa, phá, chiếm quyền |
| **XSS / SQLi / CSRF** | Tấn công vào ứng dụng web | Mất quyền, mất dữ liệu |
| **Phishing** | Dùng web/email giả | Lừa người dùng |
| **Replay / Brute-force** | Lặp lại gói / đoán mật khẩu | Truy cập trái phép |
| **Privilege Escalation** | Leo lên quyền cao hơn | Hacker điều khiển hệ thống |
|  |  |  |

## 13. Các hệ thống phòng thủ, an ninh, bảo mật + đặc điểm + cách hoạt động

### **a) IDS**

**IDS (Intrusion Detection System)** là **hệ thống phát hiện xâm nhập**, có nhiệm vụ:

* Giám sát mạng hoặc hệ thống
* Phát hiện hành vi **bất thường hoặc nguy hiểm**
* **Cảnh báo quản trị viên**, không chủ động chặn (khác IPS)

👉 IDS là **tên gọi chung** cho một loại hệ thống – bên trong có **nhiều phương pháp phát hiện khác nhau**

1. ***Hệ thống phát hiện xâm nhập dựa vào dấu hiệu (Signature-based IDS)***

Là : hệ thống giám sát lưu lượng hoặc hành vi, sau đó so sánh với các mẫu tấn công đã biết (signature) để phát hiện hành vi xâm nhập.

**Signature là gì?**

Là một **mẫu (pattern)** đại diện cho một hành vi tấn công cụ thể, ví dụ:

* Dạng gói tin (TCP flag, chuỗi lệnh)
* Câu lệnh SQL đặc trưng cho SQL Injection
* Mẫu lưu lượng SYN liên tục (DDoS)
* Câu lệnh cmd.exe bất thường trong HTTP

📌 Signature giống như **"dấu vân tay" của một cuộc tấn công**.

**Cách hoạt động :**

* IDS nhận lưu lượng mạng hoặc nhật ký (log)
* So sánh với **cơ sở dữ liệu các mẫu tấn công**
* Nếu trùng khớp → Cảnh báo

***So sánh với Anomaly-based ( Phát hiện hành vi bất thường )***

| **Tiêu chí** | **Signature-based** | **Anomaly-based** |
| --- | --- | --- |
| Dựa vào | Dấu hiệu đã biết | Mô hình hành vi bình thường |
| Phát hiện zero-day | ❌ Không | ✅ Có thể |
| False positive | Rất thấp | Dễ bị cao |
| Triển khai | Dễ hơn | Cần huấn luyện |
| Ví dụ | Snort | OSSEC, ML-based IDS |

***Một số dạng khác của IDS :***

| **Phân loại theo cách phát hiện** | **Mô tả** |
| --- | --- |
| **Signature-based IDS** | So sánh với **mẫu tấn công đã biết** (rule) |
| **Anomaly-based IDS** | Phát hiện **hành vi bất thường** so với mẫu bình thường |
| **Heuristic-based IDS** | Dựa vào **kinh nghiệm, rule mở rộng** để phát hiện các mẫu mới |
| **Hybrid IDS** | Kết hợp nhiều phương pháp để tăng hiệu quả |

### **b) IPS**

**IPS (Intrusion Prevention System)** là hệ thống **giám sát lưu lượng mạng giống như IDS**, nhưng có thêm khả năng **tự động can thiệp và ngăn chặn** hành vi tấn công **ngay khi phát hiện**.

***So sánh với IDS :***

| **Tiêu chí** | **IDS** | **IPS** |
| --- | --- | --- |
| Viết tắt | Intrusion Detection System | Intrusion Prevention System |
| Mục tiêu | Phát hiện tấn công | Phát hiện + Ngăn chặn |
| Hành vi | Giám sát, ghi log, gửi cảnh báo | Ghi log + **drop/block gói tin** |
| Vị trí triển khai | Ngoài luồng (out-of-band) | **Inline (trong luồng dữ liệu)** |
| Tác động hiệu suất | Ít ảnh hưởng | Có thể ảnh hưởng nếu cấu hình sai |
| Rủi ro | Có thể bỏ sót tấn công (nếu không phản ứng kịp) | Có thể **block nhầm (false positive)** gây gián đoạn dịch vụ |

***Một số công cụ, phần mềm thực hiện :***

| **Hệ thống** | **IDS hay IPS** |
| --- | --- |
| **Snort** | Có thể cấu hình làm **IDS hoặc IPS** |
| **Suricata** | Hỗ trợ cả hai |
| **Cisco Firepower**, **Palo Alto**, **Fortinet** | IPS chuyên dụng |
| **Zeek (Bro)** | IDS chuyên sâu, không block |
| **WAF (Web Application Firewall)** | Một loại IPS chuyên biệt cho HTTP/Web |

### **c) SIEM**

**SIEM** là viết tắt của: **Security Information and Event Management** (Quản lý sự kiện và thông tin an ninh)

* *Đây là một nền tảng tổng hợp – phân tích – giám sát bảo mật tập trung, giúp phát hiện sớm và phản ứng với mối đe dọa mạng.*
* *SIEM tích hợp log từ những hệ thống trên để phân tích!*

***Hoạt động :***

| **Chức năng chính** | **Mô tả** |
| --- | --- |
| 🧲 **Thu thập log** | Tập trung log từ nhiều nguồn: firewall, IDS/IPS, máy chủ, ứng dụng |
| 🧹 **Chuẩn hóa & lưu trữ** | Biến log "thô" thành dữ liệu có thể phân tích |
| 🔍 **Phân tích & phát hiện** | Dùng rule, AI/ML, correlation để phát hiện tấn công, bất thường |
| 🚨 **Cảnh báo theo thời gian thực** | Gửi email, SMS, kích hoạt phản ứng tự động |
| 🕵️‍♂️ **Forensics & Audit** | Truy vết tấn công, điều tra sau sự cố |
| 📊 **Báo cáo & tuân thủ** | PCI-DSS, ISO 27001, GDPR, HIPAA... |

## 14. Máy chủ Proxy là gì? + Đặc điểm

Proxy Server (Máy chủ ủy quyền) là một máy chủ trung gian đứng giữa client (người dùng) và máy chủ đích (server), có nhiệm vụ đại diện thay người dùng để gửi và nhận dữ liệu.

**Mục tiêu chính:**

* Ẩn danh người dùng
* Kiểm soát và lọc lưu lượng
* Tăng hiệu suất nhờ cache
* Ghi log, giám sát truy cập
* Vượt tường lửa, truy cập bị chặn

**Cách hoạt động :**

1. Người dùng truy cập website → Gửi yêu cầu đến **proxy**
2. Proxy nhận yêu cầu, gửi đến website đích
3. Proxy nhận phản hồi, rồi chuyển lại cho người dùng

📌 Người dùng **không giao tiếp trực tiếp** với server đích → **IP thật bị ẩn**

| **Đặc điểm** | **Mô tả** |
| --- | --- |
| 🎭 **Ẩn IP thật của client** | Giúp tăng tính riêng tư hoặc vượt kiểm duyệt |
| 🧹 **Lọc nội dung** | Chặn website, từ khóa, file tải về |
| 🧾 **Ghi log truy cập** | Giám sát hoạt động người dùng, phân tích bảo mật |
| 🚀 **Tăng tốc độ truy cập** | Dùng bộ nhớ đệm (cache) để tải lại nhanh nội dung phổ biến |
| 🔒 **Tường lửa lớp ứng dụng nhẹ** | Có thể giới hạn port, giao thức, truy cập theo thời gian |
| ❌ **Không mã hóa (nếu không dùng HTTPS Proxy)** | Không an toàn bằng VPN nếu dùng proxy HTTP thông thường |

## 15. Defense in depth ( mô hình bảo mật theo chiều sâu )

Layer 1 → Data security,

Layer 2 → Application security,

Layer 3 → Host security,

Layer 4 → LAN security,

Layer 5 → Perimeter security

Layer 6 → Physical security,

Layer 7 → Policies, procedures, awareness,

Cách nhớ nhanh : "**DAHPNU**" → (Dữ liệu – Ứng dụng – Host – Phần mạng – Ngoại vi – User)

## 16. Buffer Overflow

**Cách phòng tránh:**

* Sử dụng thư viện, hàm an toàn như fgets, strncpy
* Áp dụng kỹ thuật như:
  + **Stack canaries**
  + **ASLR (Address Space Layout Randomization)**
  + **DEP/NX (Data Execution Prevention)**
* Kiểm tra đầu vào cẩn thận (input validation)

Với ASLR :

Trong HĐH Linux, để tắt chức năng phát sinh địa chỉ bộ nhớ ngẫu nhiên, sử dụng lệnh nào?

***$sudo sysctl –w  kernel.randomize\_va\_space=0***

## 17. Dạng bài True Possitive, False Negative,…

| **Kết quả hệ thống** | **Thực tế là…** | **Phát hiện là…** | **Gọi là** |
| --- | --- | --- | --- |
| Có tấn công | ✅ Có | ✅ Có | ✅ **True Positive (TP)** |
| Có tấn công | ✅ Có | ❌ Không | ❌ **False Negative (FN)** |
| Không tấn công | ❌ Không | ✅ Có | ❌ **False Positive (FP)** |
| Không tấn công | ❌ Không | ❌ Không | ✅ **True Negative (TN)** |

Giải thích đơn giản

| **Loại** | **Ý nghĩa** |
| --- | --- |
| ✅ **True Positive (TP)** | Hệ thống **phát hiện đúng** một mối đe dọa |
| ❌ **False Positive (FP)** | **Báo động nhầm** – tưởng có tấn công nhưng không có |
| ❌ **False Negative (FN)** | **Bỏ sót tấn công** – có mối đe dọa nhưng không phát hiện |
| ✅ **True Negative (TN)** | Không có tấn công, hệ thống cũng không báo gì – đúng |

## 18. Các dịch vụ và cổng

| **Dịch vụ** | **Cổng** |
| --- | --- |
| HTTP / HTTPS  FTP – dữ liệu / FTP điều khiển ( thường đi chung ) | 80 / 443  20 / 21 |
| SFTP | 22 |
| SSH | 22 |
| DNS | 53 |
| SMTP / POP3 / IMAP | 25 / 110 / 143 |
| SMB | 445 |
| RDP | 3389 |
| VPN (OpenVPN) | 1194 |
| MySQL | 3306 |

Công cụ nào dùng để quét cổng của máy tính là **nmap**

**HTTPS = HTTP + TLS**

**Cổng mặc định: 443 (TLS), 80 (HTTP thường)**

## 19. Thuật toán Diffie-Hellman ( chi tiết về cách hoạt động của Diffie-Hellman + bài toán )

**Diffie–Hellman (DH)** là giao thức thỏa thuận khóa, cho phép hai bên (ví dụ: Alice và Bob) tạo ra cùng một khóa bí mật chung mà không cần gửi khóa trực tiếp qua mạng.

🎯 Mục tiêu: tạo được shared secret key dùng để mã hóa, nhưng không lộ khóa thật cho kẻ tấn công.

**Ứng dụng của DH**

* Dùng để **tạo khóa bí mật chung** trong các hệ thống:
  + Mã hóa đối xứng
  + VPN (IPSec), SSL/TLS
  + Giao tiếp bảo mật giữa 2 thiết bị

**2. Cơ chế hoạt động (giải thích dễ hiểu)**

**💡 Giống như pha màu sơn:**

1. Alice và Bob cùng **thỏa thuận công khai**:
   * Một màu nền g (ví dụ: vàng)
   * Một quy tắc trộn màu p (ví dụ: modulo 23)
2. Mỗi người **chọn màu riêng bí mật** (khóa riêng):
   * Alice: a = 4
   * Bob: b = 3
3. Họ **trộn màu riêng với màu chung** (g^a mod p → gửi đi):
   * Alice gửi: A = g^a mod p
   * Bob gửi: B = g^b mod p
4. Mỗi bên **trộn màu nhận được với màu riêng của mình**:
   * Alice: K = B^a mod p
   * Bob: K = A^b mod p

➡️ Cả hai đều thu được **chung một khóa bí mật K**, mặc dù không ai gửi a hoặc b qua mạng.

**Ví dụ bài tập số (tay tính)**

**🎯 Cho:**

* p = 23 (số nguyên tố)
* g = 5 (cơ sở sinh)

**🔒 Alice chọn:**

* a = 6 → Tính A = g^a mod p = 5^6 mod 23 = 15625 mod 23 = 8

**🔒 Bob chọn:**

* b = 15 → Tính B = 5^15 mod 23 = 30517578125 mod 23 = 2

→ Alice gửi A = 8, Bob gửi B = 2

**🧠 Tính khóa chung:**

* Alice: K = B^a mod p = 2^6 mod 23 = 64 mod 23 = 18
* Bob: K = A^b mod p = 8^15 mod 23

→ Tính nhanh bằng Python: 8^15 mod 23 = 18

✅ **Cả hai đều thu được khóa chung là 18**, mặc dù chưa từng truyền khóa riêng!

## 20. Một số thuật toán cần biết

Các thuật toán mã hoá :

* SHA-512
* DES
* 3DES
* AES
* MD5

Sắp xếp các thông tin cho đúng về độ dài đầu ra của các thuật toán mã hóa trên :

* SHA-512 → 512bits,
* DES → 64bits,
* 3DES → 64bits,
* AES → 128bits,
* MD5 → 128bits

Các giải pháp mã hóa KHÔNG đảm bảo được tính chất nào? : **Tính sẵn sàng**

**So sánh tổng quan**

| **Thuật toán** | **Là loại gì?** | **Mục tiêu chính** | **Có mã hóa dữ liệu không?** | **Bảo mật hiện tại** |
| --- | --- | --- | --- | --- |
| SHA-512 | Hàm băm (Hash Function) | Tạo dấu vân tay số (kiểm tra toàn vẹn) | ❌ Không mã hóa | ✅ An toàn |
| MD5 | Hàm băm (Hash Function) | Như SHA, nhưng yếu hơn | ❌ Không mã hóa | ❌ Đã bị phá |
| DES | Thuật toán mã hóa đối xứng (block cipher) | Mã hóa dữ liệu | ✅ Có | ❌ Quá yếu |
| 3DES | Mã hóa đối xứng (DES x 3 lần) | Mã hóa an toàn hơn DES | ✅ Có | ⚠️ Cũ, đang loại bỏ |
| AES | Mã hóa đối xứng hiện đại | Mã hóa dữ liệu mạnh và nhanh | ✅ Có | ✅ Rất an toàn |

**SHA-512**

* Là một hàm băm trong họ SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512…).
* Dùng để tạo ra chuỗi 512-bit đại diện cho dữ liệu.
* Không thể dùng để giải ngược, không dùng để mã hóa.
* Ứng dụng:
  + Kiểm tra toàn vẹn file (checksum)
  + Làm thành phần trong HMAC và chữ ký số
* ✅ Hiện vẫn an toàn và được khuyến nghị sử dụng.

**MD5**

* Cũng là một hàm băm, cho ra 128-bit hash.
* Rất phổ biến trước năm 2010.
* ❌ Hiện nay không còn an toàn: dễ tạo ra collision (2 dữ liệu khác → cùng hash).
* Không nên dùng trong bất kỳ ứng dụng bảo mật nghiêm túc nào.

**DES (Data Encryption Standard)**

* Là một thuật toán mã hóa đối xứng cổ điển, ra đời từ 1970s.
* Mã hóa 64-bit block với 56-bit key (thật ra là 64 bit nhưng 8 bit dùng kiểm tra).
* Dễ bị brute-force vì độ dài khóa ngắn.
* ❌ Không còn được dùng trong môi trường bảo mật.

**3DES (Triple DES)**

* Dùng DES 3 lần để tăng cường độ bảo mật.
* Gồm các phiên bản:
  + DES–Encrypt → DES–Decrypt → DES–Encrypt
* Khóa dài hơn (112 hoặc 168 bit).
* ⚠️ Vẫn chậm và bị coi là cũ – NIST đã ngừng khuyến nghị sử dụng sau năm 2023.

**AES (Advanced Encryption Standard)**

* Là thuật toán mã hóa đối xứng hiện đại và mạnh nhất hiện nay.
* Hỗ trợ key 128, 192 hoặc 256 bit.
* Mã hóa từng khối 128 bit, nhanh, hiệu quả và bảo mật cao.
* Được dùng trong:
  + HTTPS (SSL/TLS)
  + Wi-Fi (WPA2/WPA3)
  + VPN, ổ cứng mã hóa…

**So sánh theo nhóm hàm băm**

| **SHA-512** | **MD5** |
| --- | --- |
| Rất mạnh | Đã lỗi thời |
| 512-bit | 128-bit |
| Dùng trong chữ ký số, HMAC | Chỉ dùng để kiểm tra đơn giản (không bảo mật) |

**So sánh theo nhóm mã hoá đối xứng :**

| **DES** | **3DES** | **AES** |
| --- | --- | --- |
| Cũ, yếu | Tốt hơn DES nhưng chậm | Nhanh, an toàn, dùng phổ biến |
| 56-bit key | 112–168 bit | 128/192/256 bit |
| Không nên dùng | Sắp ngưng dùng | Nên dùng |

| **Thuật toán** | **Độ dài băm** | **Bảo mật** | **Ứng dụng hiện tại** |
| --- | --- | --- | --- |
| MD5 | 128 bit | Không an toàn | Kiểm tra toàn vẹn không nhạy cảm |
| SHA-1 | 160 bit | Không an toàn | Hạn chế sử dụng |
| SHA-256 | 256 bit | An toàn | Chữ ký số, xác thực |
| SHA-512 | 512 bit | An toàn | Các ứng dụng bảo mật cao |